OGC Testbed-16

Federated Security

Publication Date: 2021-01-13

Approval Date: 2020-12-15

Submission Date: 2020-11-19

Reference number of this document: OGC 20-027

Reference URL for this document: http://www.opengis.net/doc/PER/t16-D019

Category: OGC Public Engineering Report

Editor: Craig A. Lee

Title: OGC Testbed-16: Federated Security

OGC Public Engineering Report

COPYRIGHT

WARNING

Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Table of Contents

1. Subject . . . 6
2. Executive Summary . . . 7
    2.1. Document contributor contact points . . . 9
    2.2. Foreword . . . 9
3. References . . . 11
4. Terms and Definitions . . . 12
    4.1. Abbreviated Terms . . . 12
    4.2. Definitions . . . 13
5. Overview . . . 17
6. Introduction . . . 18
7. A Quick Review of Security and Trust . . . 19
    7.1. Authentication . . . 19
    7.2. Authorization . . . 19
    7.3. Privacy . . . 19
    7.4. Integrity . . . 20
    7.5. Non-repudiation . . . 20
    7.6. Trust . . . 21
8. Federation Security and Trust Requirements: A Threat Analysis . . . 22
    8.1. Threat Modeling Methods, Tools, and Mitigations . . . 22
    8.2. A Federation Service Threat Model and Analyses . . . 25
        8.2.1. A Threat Analysis of Authentication to a Federation Service . . . 28
        8.2.2. A Threat Analysis of FedAdmin and ResOwner Interactions . . . 30
        8.2.3. A Threat Analysis of User Interactions . . . 31
        8.2.4. A Threat Analysis of FS-FS Interactions . . . 32
    8.3. Threat Analysis Observations . . . 33
9. Emerging Security Technologies . . . 35
    9.1. Identity and Credential Management . . . 35
        9.1.1. Identity Proofing . . . 35
        9.1.2. Authentication . . . 35
        9.1.3. Federated Credential Management . . . 36
        9.1.4. Attribute Release and Verifiable Credentials . . . 38
    9.2. Trust . . . 40
        9.2.1. Defining Trust . . . 40
        9.2.2. Trust Frameworks . . . 41
        9.2.3. Trust Framework Approaches . . . 46
        9.2.4. The OIDC Federation Specification . . . 51
        9.2.5. Hardware Roots of Trust . . . 53
        9.2.6. Blockchain in Federated Environments . . . 56

9.3. Zero Trust . . . 58
9.3.1. The Zero Trust Architectural Concepts . . . 58
9.3.2. The Microservice Mesh ZTA Approach . . . 61
9.3.3. Zero Trust Federations . . . 63
9.4. Data-Centric Security . . . 66
9.5. Trusted Data Format . . . 68
9.6. Attribute-Based Encryption . . . 69
9.7. Trusted Execution Environments . . . 69
9.8. Homomorphic Computing . . . 71
9.9. Quantum Key Distribution . . . 71
10. Possible Development and Adoption Strategies . . . 73
10.1. An Assessment for Viability, Relevance, and Maturity . . . 73
10.2. A Federation Development Roadmap . . . 75
10.2.1. Roadmap Phase 1: Near-term Tasks . . . 75
10.2.2. Roadmap Phase 2: Subsequent Tasks . . . 76
10.2.3. Roadmap Phase 3: Longer-term Tasks . . . 76
11. Final Comments . . . 78
Appendix A: Threat Dragon Report . . . 79
Appendix B: Revision History . . . 114
Appendix C: Bibliography . . . 115

Chapter 1. Subject

This OGC Testbed 16 Engineering Report (ER) examines all aspects of security and trust in federated computing environments as defined in the NIST Cloud Federation Reference Architecture cite:[CFRA2020]. The security and trust requirements are identified. Then possible approaches for achieving security and trust are examined. These approaches range from traditional methods for securing just the basic communications among federated entities to the use of emerging security technologies including federated roots of trust, trust frameworks, blockchain, data-centric security, and zero trust architectures.

Chapter 2. Executive Summary

A *federation* is a set of organizations that wish to collaborate at a level beyond simply using document sharing web sites. A federation enables stakeholders to identify which services and resources are to be shared, who can access them, and the joint policies governing their use. A federation can be considered a *virtual administrative domain* that can span multiple physical organizations.

As defined in the NIST Cloud Federation Reference Model (CFRA) cite:[CFRA2020], federated systems can be deployed using one or more *Federation Managers*. For consistency with the current work of the IEEE P2302 WG cite:[P2302], this ER uses the term *Federation Hosting Services* (FHS) as a synonym to Federation Manager. For brevity, Federation Hosting Service will be shortened to just *Federation Service*.

These Federation Services can host multiple *federation instances*, i.e., virtual administrative domains. To accomplish this, these FHSs must exchange resource information, user credentials, and enable access decisions to be made across the stakeholder organizations. Hence, if a FHS was compromised in a security breech, the security of resources across multiple federations could be put in jeopardy. **The business value of this Engineering Report is to identify the vulnerabilities of possible federation implementations, suggest mitigations, and examine emerging security approaches that can be integrated to achieve the highest possible levels of trust and security.**

After a brief review of basic security and trust terminology, a threat analysis is done for a specific CFRA-based federation deployment. As noted in the CFRA, many deployment and governance models are possible. Hence, this analysis is against one specific deployment that is expected to be widely used. This deployment model uses OpenID Connect (OIDC) as the identity providers. The threat analysis is done using *Threat Dragon*, an open source threat analysis tool wherein the major system components are defined in a graphical format, trust boundaries are identified, and interactions catalogued. Threat Dragon does not identify specific threats nor does it recommend remedies. However, Threat Dragon provides a framework for reasoning about possible threats, where to look for them, and how they could be mitigated. Four interaction scenarios were examined and the following number of threats and mitigations were identified:

Table 1. Identified Threats.

Number of Identified Threats	Interaction Scenario
29	Authentication to a Federation
10	FedAdmin/ResOwner Interactions
16	User Interactions
18	FS-FS Interactions

The ER notes that many of these identified threats were multiple instances of the same kind of threat. For example, all communication links were vulnerable to *man-in-the-middle* attacks that are mitigated by the use of TLS.

The take-away message from this threat analysis is that the vulnerabilities of a federation are derived from the vulnerabilities of the tools they are built on. In addition to using TLS on every communication link, federations built on OIDC inherit the vulnerabilities of OIDC. In turn, OIDC vulnerabilities arise from the specific implementation approach used. For example, OIDC uses two HTTP redirections during the authentication process. To prevent various attacks, the redirection URIs must be full with no parameters and registered with the OIDC server. These URIs must checked on every redirection to prevent Cross-Site Request Forgery attacks. Hence, any federation threat analysis must do a threat analysis of every tool used in a specific implementation. What is different about federations are the possible impacts of a compromise. If a compromise does occur, then federation tooling must be designed to contain and limit any adverse impacts.

The need for enhanced security is certainly not specific to federations. There are many emerging security technologies that could be integrated and utilized in federation systems Many emerging technologies are reviewed in this ER.The Executive Summary only reviews the technologies that are recommended for further investigation and investment. These emerging security technologies can be broadly categorized as follows:

- **Identity and Credential Management:**
  Not only is OpenID Connect (OIDC) gaining in popularity, the OpenID Foundation is also building out support for federations. Specifically the OIDC Federation Specification enables OIDC installations to establish trust and secure communications. This is done using a protocol to find *common trust anchors* and subsequently exchange JSON Web Key Sets whereby all communication can be encrypted and signed. Work is also being done on *shared signaling* between OIDCs. These tools can be directly used in CFRA-based Federation Hosting Services.
- **Trust and Zero Trust:**
  All federation tooling will directly depend on establishing shared data that is *trusted*. Hence, *blockchain* could be employed to establish trust for service catalogs, roles and attributes, policies, etc. This would entail evaluating different consensus algorithms, block sizes, and scalability in federated environments. As an alternative approach, the *Zero Trust Architecture (ZTA)* concept is based on defining *zones of implicit trust* that are protected by *Policy Enforcement Points*. These zones of implicit trust are also made as small as possible. The ZTA concept has strong parallels with the CFRA-based federation approach which can use Web Service API Gateways that provide Policy Enforcement Points at the RESTful service level. Hence, the ZTA concept can be directly extended to that of *Zero Trust Federations*.
- **Data Security Methods:**
  When sharing data, *data leakage* is always a concern. After data is shared with a partner, how can the partner be prevented from sharing the data with a third-party that should not have the data? The *Next Generation Access Control (NGAC)* paradigm supports dynamic policy management. As an example, this means that if User X is given permission to read Data Q, User X can no longer write Data Q, i.e., copy it to any third-party. The *Data-Centric Security (DCS)* model being pursued in Testbed-16 is based on using a *DCS Service* and a separate *Key Management Service (KMS)*. The DCS only serves encrypted data. Any data recipient must also contact the KMS to retrieve the necessary decryption keys. These services can be managed in a federation — as any other kind of services. This dovetails with the ZTA and ZTF concepts of encrypting all data between trust zones. The *Trusted Data Format (TDF)* approach takes this a step farther. A Trusted Data Object actually incorporates a Policy Enforcement Point that can contact a remote authorization service to authorize a user's access. Finally, *Trusted Execution Environments (TEEs)* actually provide hardware support whereby data is only decrypted when it is copied into special memory regions that can only be accessed by authorized processes. TEE

use cases include cloud computing where a remote cloud user can ensure their data is encrypted until it gets installed in the TEE, and that no one else can see it — not even the hypervisor or host OS. While more investigation needs to be done, it might be possible to utilize TEEs are part of federated environments.

All of these topics are organized in a simple, three-phase, roadmap.

- Phase 1 consists of near-term tasks.
- Phase 2 consists of subsequent tasks that build on Phase 1.
- Phase 3 consists of tasks that will have a longer development cycle.

Specifically, Phase 3 includes a review of the NIST SP 800-53 catalog of security controls to identify controls directly relevant to federations, and to identify any possible federation-specific controls that should be added. (This could ultimately be part of any FedRAMP process granting federated systems Authorization to Operate.) A formal method for defining federations is also a longer-term project. Formal definitions for federation requirements and semantics would enable more aspects of federation management to be automated. This could include compliance assessment for security purposes in Federation Trustmark Frameworks.

A final take-home message is that the governance of a CFRA-based federation is *purpose-defined*. It does not federate existing environments that were not intended nor designed with federation in mind. These could be called *purpose-built* federations. This approach makes the actual federation governance much easier. However, requiring organizations to agree on how to run their joint federations may force them to face and resolve significant organizational differences.

Since federation is inherently *cross-organizational*, some type of *umbrella organization* could greatly facilitate the development and adoption of federations, and the resolution of cross-organizational issues. By analogy with the Internet, in the 1970s the Advanced Research Projects Agency (ARPA) created *ARPANET*. ARPANET was essentially an umbrella environment whereby many universities, corporations and government agencies could bring their resources to the table and work together in the nascent field of computer networking. We argue that a similar type of umbrella organization would greatly benefit all stakeholders in the development of federation.

2.1. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Contacts

Name	Organization	Role
Craig A. Lee	Federation Partners LLC	Editor

2.2. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 3. References

The following normative documents are referenced in this document.

- IETF: Hypertext Transfer Protocol Version 3 (HTTP/3).
- IETF: RFC 2818, HTTP Over TLS.
- IETF: RFC 8259, The JavaScript Object Notation (JSON) Data Interchange Format.
- IETF: RFC 7517, JSON Web Key (JWK).
- IETF: RFC 7519, JSON Web Token (JWT).
- IETF: RFC 6749, The OAuth 2.0 Authorization Framework.
- OpenID Foundation: OpenID Connect Core 1.0.
- OpenID Foundation: OpenID Connect Federation 1.0 - draft 12.
- OpenID Foundation: OpenID Authentication 2.0 - Final.
- OASIS: PKI (Public Key Infrastructure) Technical Standards.
- OASIS: Security Assertion Markup Language (SAML) V2.0 Technical Overview.
- DNI: XML Data Encoding Specification for Trust Data Format, Version 3.
- IETF: RFC 8446, The Transport Layer Security (TLS) Protocol Version 1.3.
- IETF: RFC 8705, OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens.
- W3C: Verifiable Credentials Data Model 1.0.
- OASIS: eXtensible Access Control Markup Language (XACML) Version 3.0.

Chapter 4. Terms and Definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard OGC 06-121r9 shall apply. In addition, the following terms and definitions apply.

4.1. Abbreviated Terms

- ABAC: Attribute-Based Access Control
- ABE: Attribute-Based Encryption
- CA: Certificate Authority
- CFRA: Cloud Federation Reference Architecture
- CSP: Credential Service Provider
- DCS: Data-Centric Security
- DoS: Denial of Service
- DFD: Data Flow Diagram
- DDoS: Distributed Denial of Service
- FHS: Federation Hosting Service
- FS: Federation Service
- IdAM: Identity and Access Management
- IdP: Identity Provider
- JWKS: JSON Web Key Set
- JWT: JSON Web Token
- KMS: Key Management Service
- NIST: National Institute of Standards and Technology
- OIDC: OpenID Connect
- OP: OIDC Provider
- MFA: Multi-Factor Authentication
- mTLS: Mutual Transport Layer Security
- ODNI: Office of the Director of National Intelligence
- OWASP: Open Web Application Security Project
- P2P: Peer-to-Peer
- PAP: Policy Administration Point
- PDP: Policy Decision Point
- PEP: Policy Enforcement Point
- PKI: Public Key Infrastructure

- QKD: Quantum Key Distribution
- RBAC: Role-Based Access Control
- REST: REpresentational State Transfer
- RP: Relying Party
- RP: Resource Provider
- SAML: Security Assertion Markup Language
- SEAL: Simple Encrypted Arithmetic Library
- SP: Service Provider
- STRIDE: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege
- TDF: Trusted Data Format
- TDO: Trusted Data Object
- TEE: Trusted Execution Environment
- TLS: Transport Layer Security
- TPM: Trusted Platform Module
- XACML: eXtensible Access Control Markup Language
- ZTA: Zero Trust Architecture
- ZTF: Zero Trust Federation

4.2. Definitions

- **API Gateway**

  A centralized point for exposing APIs to backend resources while abstracting the architectural details from the client. An API Gateway maintains a consistent interface to the requesting clients. It may also support middleware functionality such as security, governance, and monitoring.

- **Data-Centric Security | DCS**

  Data-centric security is an approach that underlines the security of the data itself rather than the security of communication infrastructure such as networks, servers, or applications. See Data Centric Security Engineering Report, OGC 20-021, cite:[DCS-TB16].

- **Federation Administrator | FedAdmin**

  The entity that has the authorization to configure and operate a Federation Instance. This entity may be distributed depending on the governance model. See NIST Cloud Federation Reference Architecture, Appendix A cite:[CFRA2020].

- **Federated Identity Management**

  The process of asserting an identity across different systems or organizations. This is a key enabler of Single Sign On and managing IdAM in cloud computing. See NIST Cloud Federation Reference Architecture, Appendix A cite:[CFRA2020].

- **Federation**

  An organization of self-governing entities that have common policies, administrative controls, and enforcement abilities governing the use of shared resources among members. A virtual administrative domain wherein multiple participating organizations/sites can define, agree upon and enforce resource discovery, access and usage policies for the sharing of a subset of their resources. See NIST Cloud Federation Reference Architecture, Appendix A cite:[CFRA2020].

- **Federation Hosting Service|Federation Service**

  The entity that provides the essential federation management functions described in the CFRA for potentially multiple federations over their lifespans. In the CFRA, this entity is called the *Federation Manager*. For consistency with the current IEEE P2302 WG, this engineering report uses the term *Federation Hosting Service* as a synonym for Federation Manager. *Federation Service* is used as a short form of Federation Hosting Service. See NIST Cloud Federation Reference Architecture, Appendix A cite:[CFRA2020].

- **Federation Operator**

  The entity that deploys, configures and maintains one or more Federation Hosting Services. See NIST Cloud Federation Reference Architecture, Appendix A cite:[CFRA2020].

- **Federation Service**

  See *Federation Hosting Service*.

- **Gateway**

  See *API Gateway*.

- **Identity Proofing**

  The process by which a CSP collects, validates, and verifies information about a person. See NIST Digital Identity Guidelines cite:[NIST800-63-3].

- **IEEE P2302 WG**

  The Standard for Intercloud Interoperability and Federation (SIIF) Working Group cite:[P2302].

- **InCommon**

  A federation-based collaboration system cite:[InCommon] that is run as part of Internet2 cite:[Internet2].

- **Microservice**

  An application functional component that is implemented in its own container. A set of microservice containers can work together to compose an application. See NIST Application Container Security Guide cite:[NIST800-190].

- **Microservice Mesh Architecture**

  An architectural approach where multiple microservices are managed by a *control plane*. The control plane can associated *sidecars* with each microservice to enable fine-grained control. The control plane also controls all ingress/egress to and from the mesh of microservices.

- **P2302 WG**

  See *IEEE P2302 WG*.

- **Resource Owner | ResOwner**

  The entity that is accountable and authorizes use and governance of a resource. See NIST Cloud Federation Reference Architecture, Appendix A citey:[CFRA2020].

- **Root of Trust**

  Roots of trust are highly reliable hardware, firmware, and software components that perform specific, critical security functions. Because roots of trust are inherently trusted, they must be secure by design. As such, many roots of trust are implemented in hardware so that malware cannot tamper with the functions they provide. Roots of trust provide a firm foundation from which to build security and trust cite:[NIST_RoT].

- **Threat Dragon**

  An open-source threat analysis tool from OWASP cite:[OWASP].

- **Trust Anchor**

  An authoritative entity for which trust is assumed. In a PKI, a trust anchor is a certification authority, which is represented by a certificate that is used to verify the signature on a certificate issued by that trust-anchor. The security of the validation process depends upon the authenticity and integrity of the trust anchor's certificate. Trust anchor certificates are often distributed as self-signed certificates. See Recommendation for Key Management cite:[NIST800-57].

- **Trust Boundary**

  See *Trust Domain*.

- **Trust Domain**

  The context in which trust has been established. This context can be established by enterprise boundaries. These boundaries can be hardware-defined or software-defined, i.e., virtual. See NIST Cloud Federation Reference Architecture, Appendix A cite:[CFRA2020].

- **Trust Framework**

  The "rules" underpinning federated identity management, typically consisting of: system, legal, conformance, and recognition. See Developing Trust Frameworks to Support Identity Federations cite:[NIST_Trust_Frameworks].

- **Trusted Platform Module**

  A hardware *root of trust* as defined by the Trusted Computing Group. See Trusted Platform Module cite:[TPM2020].

- **Trustmark**

  A cryptographically signed document that signifies compliance to a desired requirement or property. Trustmarks can be linked to a *root of trust*, such as a PKI Certificate Authority. See Developing Trust Frameworks to Support Identity Federations cite:[NIST_Trust_Frameworks].

- **Verifiable Credentials**

  A credential management system from W3C that allows users to directly manage their own identity credentials. See Verifiable Credentials Data Model 1.0 cite:[W3C_VCs].

- **X.509**

  The X.509 public-key certificate or the X.509 attribute certificate, as defined by the ISO/ITU-T X.509 standard. An X.509 certificate refers to the X.509 public-key certificate. See Recommendation for Key Management cite:[NIST800-57].

- **Zero Trust Architecture**

  An architectural approach where there is no implicit trust granted to systems based on their physical or network location (i.e., local area networks v. the Internet). The ZTA design philosophy is to make trust zones as small as possible, i.e., to make protection as *fine-grained* as possible. This design philosophy is embodied in a set of ZTA design tenets cite:[Rose2019].

- **Zero Trust Federation**

  Federation deployments that satisfy the tenets of Zero Trust Architecture design.

Chapter 5. Overview

Section 6 provides an introduction to the topic of security and trust in federated computing environments as defined by the NIST Cloud Federation Reference Architecture (CFRA) cite:[CFRA2020].

Section 7 provides a quick overview of the main security and trust concepts. This is a quick "level set" on the terminology that is used throughout this ER.

Section 8 reviews the NIST CFRA to establish the necessary security and trust requirements. The CFRA is composed of a number of conceptual actors that have various interactions. By considering these actors and interactions, and how they may be mapped into concrete implementations, the security and trust requirements can be identified.

Once the requirements are established, **Section 9** examines a number of approaches for satisfying those requirements. These range from common methods for securing the communication among federated entities, to the use of Zero Trust Architectures.

While a number of security and trust approaches exist for federations, actually realizing them will require *socialization* and the investment of organizational resource. Hence, **Section 10** discusses a number of development and adoption strategies.

Section 11 provides final comments concerning the field of federation security and trust.

Chapter 6. Introduction

A *federation* is a set of organizations that wish to collaborate through their IT systems. The desired level of collaboration, however, goes beyond simply making web sites available to each other, or sharing documents on a third-party, such as Google Docs. In a federation, the stakeholders can identify which services or resources are to be shared, who from the organizations can access them, and what the policies are for the resource's joint use. A federation can be thought of as a *virtual administrative domain* that can span multiple physical organizations. In the cloud computing era, processors, storage, networks, and many other resource types have all been virtualized. Virtualizing the administrative domains for sets of these resources is being recognized as the natural next step in this progression.

To this point, NIST published the NIST Cloud Federation Reference Architecture cite:[CFRA2020] in February 2020. As a reference architecture, the CFRA is inherently conceptual as it organizes the entire federation design space, yet it also presents several implementation approaches. Federation can be explained in a nutshell by Figure 1 from the CFRA.

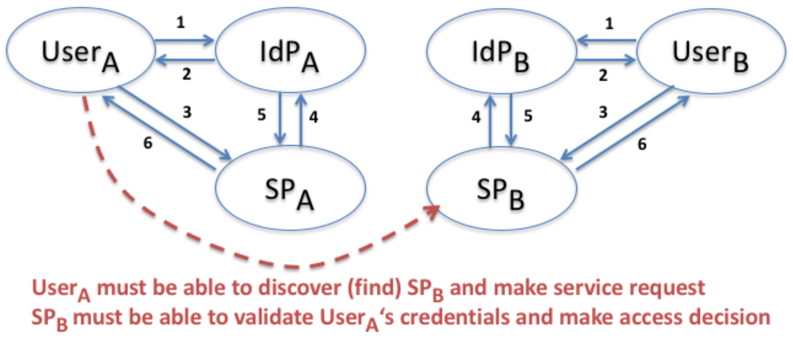

User$_A$
IdP$_A$
SP$_A$
IdP$_B$
User$_B$
SP$_B$
1
2
3
4
5
6
User$_A$ must be able to discover (find) SP$_B$ and make service request
SP$_B$ must be able to validate User$_A$'s credentials and make access decision

Figure 1. Federation in a Nutshell.

Modern administrative domains are all based on the notion of having an independent *Identity Provider (IdP)* that issue identity credentials to *Users*. When requesting service from a *Service Provider (SP)*, the SP validates the User's credentials with the issuing IdP. Different Identity and Access Management (IdAM) systems may have a slightly different sequence of operations, and may involve HTTP redirections, but having an independent, third-party IdP is fundamental.

Now consider two organizations, *A* and *B*, that are each in their own "identity silo". A federated environment will enable a *User A* to able to discover (find) a *SP B* at another site, and enable SP B to make a proper access decision based on User A's credentials. The are many implementation approaches that can realize these fundamental operations.

Any implementation approach, however, has serious security responsibilities. Since a federation system is necessarily spanning organizational trust boundaries, and is managing identity credentials and resource access, any compromise could have serious consequences. If a Federation Hosting Service was compromised, resources at multiple sites could be jeopardized. Hence, the purpose of this Engineering Report is to examine the threats and vulnerabilities of federation systems, and how emerging security approaches can be integrated to achieve the highest level of trust and security possible.

Chapter 7. A Quick Review of Security and Trust

Security and trust are broad terms that cover a number of constituent concepts. These concepts are briefly reviewed here to provide a "level set" for their meanings.

7.1. Authentication

```
Authentication: Verifying the identity of a user, process, or device, often as a
prerequisite to allowing access to resources in an information system.①
```

① NIST Security and Privacy Controls for Federal Information Systems and Organizations cite:[NIST800-53]

Informally, *authentication* is the process of verifying that a person or entity is who they say they are. In IT systems, this means validating a user's *identity credentials* with the *Identity Provider (IdP)* that issued them. Identity credentials can be represented in many different formats and the protocols for verifying them can also vary. Nonetheless, identity credentials will carry a number of *identity attributes* about the credential holder. These can be simple items, such as first and last name or more organization-specific items, such as one's role or position in an organization chart.

7.2. Authorization

```
Authorization:  The process of verifying that a requested action or service is
approved for a specific entity.①
```

① Guide to Industrial Control Systems (ICS) Security cite:[NIST800-82r2]

Once a person or non-person entity has been authenticated, and their identity credentials have been validated, any requests that are being made to access resources can be validated as well, i.e., they can be *authorized* to access the resource. This is typically done by evaluating a person's identity attributes against the resource's *access policy* and making an access decision.

Note that authorization does not have to be a yes/no binary decision. Authorization can involve how a service or resource can be used and also the context in which the resource is being used. For example, some users may get access to high-resolution imagery, while others may only get low-resolution imagery. Similarly, a user may have full access while at work in the office, but the same user may be denied access after-hours when using a Starbuck's Wi-Fi.

7.3. Privacy

```
Privacy:  Preserving authorized restrictions on information access and disclosure,
including means for protecting personal privacy and proprietary information.①
```

① NIST Security and Privacy Controls for Federal Information Systems and Organizations cite:[NIST800-53]

Privacy entails both making proper access decisions as well as preventing unwanted and unintended disclosure while data is at rest or in-transit. This can mean preventing cyberattacks that compromise a system, and also encrypting data when in transmission. These data do not have to be just personal or proprietary, but could be any information that an organization wishes to control the dissemination of.

Privacy is often equated with *confidentiality* but a distinction can be drawn between the two terms. Privacy typically means controlling the disclosure of any information. Confidentiality can mean the willingness to disclose information to a second party with the expectation that the information will not be disclosed to a third party. A typical example is a client disclosing information to an attorney. In both cases the goal is to control the disclosure of information.

7.4. Integrity

```
Integrity:  Guarding against improper information modification or destruction, and
includes ensuring information non-repudiation and authenticity.①
```

① NIST Security and Privacy Controls for Federal Information Systems and Organizations cite:[NIST800-53]

In some cases, an organization may not care who sees a particular data set. At the same time knowing that the data is correct and that the data has not be corrupted, changed or tampered with in any way is critical. Data could be corrupted due to non-malicious events, such as a software or hardware failure. Data can also be corrupted due to malicious events, such as a cyberattack. In operational environments, integrity can mean preventing any corruption from happening for any reason. This could also mean simply mean being able to *detect* that data has been corrupted.

A simple example of data that is public but requires integrity is government-produced weather data. Such data is often produced for the personal use of citizens, but is also produced to promote the economic activity of organizations. Therefore, "monkey-wrenching" weather data could have serious impact on organizations that use the weather data to make business decisions.

7.5. Non-repudiation

```
Non-repudiation:  Protection against an individual falsely denying having performed a
particular action. Provides the capability to determine whether a given individual
took a particular action such as creating information, sending a message, approving
information, and receiving a message.①
```

① NIST Security and Privacy Controls for Federal Information Systems and Organizations cite:[NIST800-53]

Non-repudiation may be less familiar to many people as a security term, but is actually quite commonplace. Non-repudiation means that two parties to a transaction of some sort (e.g., exchanging information) cannot later deny that it took place. A simple example is that of going to a grocery store and getting a receipt. If you meant to buy a can of baked beans but bought refried beans my mistake, the grocery store will probably want to see the receipt before allowing an exchange. The receipt makes disputing the transaction difficult. In IT systems, enforcing non-repudiation is somewhat more complicated, but can nonetheless be critical.

7.6. Trust

```
Trust:  The willingness to take actions expecting beneficial outcomes, based on
assertions by other parties.①
```

① Guide to Secure Web Services cite:[NIST800-95]

The definitions provided in this section of the ER are intended to be as concise and to-the-point as much as possible. While the above definition captures the essence of what trust is, it nonetheless has immense implications for what trust means in the context of IT systems. The first step in establishing any trust relationship is establishing the identity of the entity with whom you are communicating. The second step could simply be ensuring that the communication between the two entities is secure. While these aspects of trust are necessary, they are by no means sufficient. Even after identity and communication integrity have been established, trust will also depend on some type of assurances that the other entity will act in a trustworthy manner.

Many tools and security controls to achieve these requirements are already cataloged in NIST Security and Privacy Controls for Federal Information Systems and Organizations cite:[NIST800-53]. In the following pages, additional methods to achieve these properties in federated environments will be discussed. This suggests that the security controls in cite:[NIST800-53] should ultimately be reviewed and revised to include any federation-specific controls that are needed.

Chapter 8. Federation Security and Trust Requirements: A Threat Analysis

With a basic understanding of security and trust concepts, understanding how they are needed in federated environments and how they can be addressed is the next step. At the conceptual level, CFRA-based federations will be hosted by one or more *Federation Managers* that interact with Users and other Federation Managers. At the implementation level, a Federation Manager will be implemented as a *Federation Service*. A Federation Service (FS) will be functionally similar to any web server. A web server can host multiple web sites and serve all manner of web content. A FS will be able to host multiple federations and serve all manner of federation content.

By standing up a FS that is listening on one or more ports, a commensurate number of *attack surfaces* will be opened up. The attack surface of a software environment is the sum of the different points where an unauthorized user can try to enter data to or extract data from an environment (Wikipedia). These federation attack surfaces will share many vulnerabilities present in other types of servers, but will also present new, federation-specific vulnerabilities.

To hopefully identify these shared and new vulnerabilities, this section will develop a federation *threat model* on which a *threat analysis* will be done. The large body of work has been researched and published in threat modeling and analysis is leveraged in the following discussion. A number of threat modeling tools exist that provide a modeling structure in which to do an analysis. These tools and evaluation paradigms are briefly reviewed here.

Note that any threat analysis is only as good as the threat model on which it is based. That is to say, threat models should be based — *as much as possible* — on a specific, concrete implementation. The less specific or concrete that a model is, the harder it becomes for a threat analysis to identify specific or concrete threats.

This makes doing "a" federation threat analysis problematic. There are clearly multiple ways a CFRA-based Federation Service could be implemented. The CFRA also identifies a range of deployment and governance models that could be used. To do any analyses, a number of implementation assumptions and deployment scenarios have to be made. As such, an exhaustive analysis is not be possible. Furthermore, an authoritative threat analysis can be a sizeable effort. This leads us to a note of caution:

```
This threat analysis should not be considered complete, definitive,
nor authoritative.  This analysis is just an initial threat model
and analysis that can point the way to more thorough efforts that
are properly funded and resourced.
```

Given this caution, available modeling tools are now discussed.

8.1. Threat Modeling Methods, Tools, and Mitigations

Each threat modeling tool is based on some *threat modeling method*. A useful review of twelve methods is presented in Threat Modeling: A Summary of Available Methods cite:[Shevchenko18].

The authors emphasize that no one method is inherently better than the others. Some methods focus on risk and privacy concerns, while others focus on people issues. Some methods operate on high-level abstractions, while others strive for finer granularity. For the threat analysis in this engineering report, the *STRIDE* model cite:[MS-STRIDE] will be used. STRIDE was developed over twenty years ago and has been adopted by Microsoft. STRIDE is an acronym based on the following threat categories:

Table 2. STRIDE Categories

Category	**Description**
Spoofing	The deliberate deceiving of a user and tricking them into disclosing sensitive information, such as username and password(s), that is then used the user's knowledge or consent. Spoofing can be done through using bogus email addresses, websites, or even IP addresses.
Tampering	The malicious modification of data, either when at-rest, such as in a database, or in-transit over a network.
Repudiation	When a party that observed an event or engaged in a transaction later denies that the event or transaction took place. What is desired is *non-repudiation*: no one can deny that the event or transaction took place.
Information Disclosure	The exposure of information to individuals who are not supposed to have access to it, either when at-rest (e.g., reading a file) or when in-transit (e.g., snooping on the network).
Denial of Service	Denial of Service (DoS) attacks deny service to valid users by overloading the service with unnecessary requests for service. Distributed Denial of Service (DDoS) attacks can be especially difficult to deal with.
Elevation of Privilege	An unprivileged user gains privileged access and thereby has sufficient access to compromise the entire system.

STRIDE and other threat modeling methods have been built into a variety of *threat modeling tools* cite:[ThreatTools]. These tools likewise have different scopes, ranging from security requirements management to identifying risk patterns throughout the software development lifecycle. Three identified tools are based on STRIDE:

- The Microsoft Threat Modeling Tool cite:[MTMT]
- Open Weakness and Vulnerability Modeller cite:[Schaad2019OpenWA,OVVL], and
- Open Web Application Security Project (OWASP) Threat Dragon cite:[OWASP,ThreatDragon]

The threat analysis in this engineering report will use *OWASP Threat Dragon*. Threat Dragon is an open source tool based on STRIDE. Threat Dragon can be downloaded as a stand-alone, desktop application, or run through an on-line application that is integrated with GitHub. This integration with GitHub means that Threat Dragon can be used to do threat analyses on code as it is being built in a GitHub repository. Since a CFRA-based Federation Service is not actually being built (yet), the desktop version of Threat Dragon (version 1.2.0) will be used.

Mitigations for the STRIDE threats cite:[ThreatMigs] can be categorized as follows:

Table 3. Mitigation Categories

Category	**Description**
Auditing and Logging	Who did what and when? Application logging of security-related events enables subsequent auditing to possibly identify malicious events.
Authentication	Who are you? Unequivocally establishing the identity of an entity is critical for establishing who did what when.
Authorization	What can you do? Authorization is determined by evaluating identity credentials against the discovery and access policies for specific resources.
Communication Security	The privacy and integrity of data as it is in-transit must be ensured.
Configuration Management	All systems are composed of multiple components that must be deployed and managed as a whole. Properly managing the configuration of these components is central to reducing threat vulnerabilities, such as making sure software versions are current, ensuring that credentials are properly handled, ensuring that components only communicate with other authorized components, and so forth.
Cryptography	Cryptographic methods to encrypt/decrypt and sign data are commonly used to provide confidentiality and integrity. Is cryptography being properly used in the system under review?
Exception Management	When something in a system fails, and an exception is raised, is the failure handled properly? First, does the system fail gracefully? Second, does the exception capture essential information about the failure? Third, is the information passed back to the user both useful but at the same time restricted to a need-to-know?
Input Validation	Validating input data when it is received by a system component can prevent many security issues. Is the data from a trusted source? Has the data been corrupted or modified in-transit? Is the data in the right format and within expected bounds? Does the input data contain any extraneous information?
Sensitive Data	Systems may have data that is considered to be more sensitive and must be appropriately protected. This includes when data is in memory, at-rest in a data store, or in-transit over a network.
Session Management	A session refers to a series of related interactions between a user and an application. Sessions are often used to establish a security environment that is used for the duration of the session. This can provide performance benefits, yet must be properly managed to avoid security vulnerabilities.

These STRIDE categories and mitigation approaches will be used in a threat analysis of a CFRA-based Federation Service using Threat Dragon. Threat Dragon works by providing a palette on which *Data Flow Diagrams (DFDs)* can be built describing how a system works. These DFDs are based on *Processes, Stores (Storage), Actors, Data Flows* and *Trust Boundaries*. For each of the elements, threats can be identified and added to the overall model. Threat Dragon also provides a rule engine that can auto-generate basic threats for each element.

What the Threat Dragon tool actually provides is a way to (a) describe how a system is built and how it works, and (b) catalog possible threats. As such, Threat Dragon does not tell the user exactly what the specific system vulnerabilities are, nor how to mitigate them. Rather it gives the knowledgeable system builder a systematic organization and structure in which to do such an analysis.

8.2. A Federation Service Threat Model and Analyses

With an understanding of threat modeling and analysis, a likely Federation Service implementation, deployment and governance model are chosen to analyze. As stated above, specific threats and mitigations can only be identified for specific system implementations. Given the range of possible implementations, and the range of deployment and governance models, it is simply not be possible to exhaustively analyze all of them in this document. Hence, a few strategic examples that are considered to be attractive and likely are chosen for analysis.

This approach requires that a number of specific architectural assumptions be made. First the case of a single Federation Service being used in a Third-Party deployment is considered. As only one Federation Service is involved, this is the simplest deployment identified in the NIST CFRA. Based on established software engineering practices, we will assume that the FS implementation will be:

- Based on RESTful services,
- Integrated with Open ID Connect, and
- Based on a Web Service API Gateway approach.

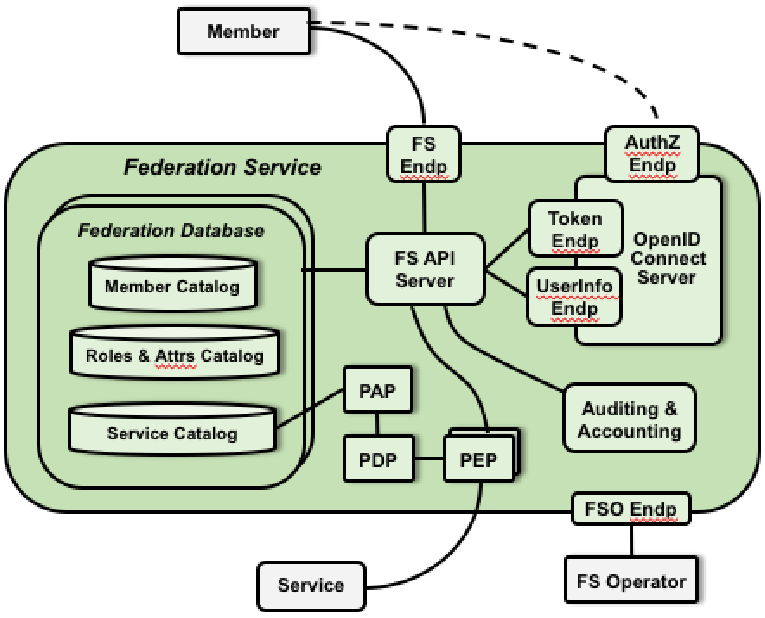


Figure 2. An Object Model for a single CFRA-based Federation Service.

This overall implementation is illustrated in Figure 2. Several other very important architectural features are identified. RESTful tools are often built with multiple *endpoints* that serve different purposes. This Federation Service has two endpoints: the *FS Endp* and the *FSO Endp*.

The FSO Endp is used exclusively by the *Federation Service (FS) Operator*. The FS Operator is the entity that "spins-up" the Federation Service — just like any other service. The FS Operator is primarily concerned with the general health and status of the server's operation, and does not get involved with the operation of any federation instances that the server is hosting. As such, the FSO Endp can be assigned to an internal subnet that is not accessible to the "outside world". This is an important security feature that eliminates many security threats.

The *FS Endp* is the endpoint exposed to all federation Members that wish to use a federation being hosted on this server. This includes all types of Members that have different roles, such as *FedAdmin*, *ResOwner*, ordinary *Users*, or any other federation-specific roles or authorizations that may be defined. With regards to this threat analysis, however, it must be assumed that this endpoint is accessible by malicious actors.

Likewise, as defined by the OIDC standard, the OIDC service has three endpoints. These endpoints are used in different ways, according to several different defined *flows* that are defined. This analysis is done using the *Authorization Code Grant Flow*, which is the typical and most secure way that OIDC can be used. In this flow, when a Member makes a request to the FS, the Member must first be authenticated. This is done by a *redirect* to the *AuthZ Endp* on the OIDC Service. Since this endpoint must be accessible to any federation members, it must also be assumed that this endpoint is accessible by malicious actors.

The OIDC service also uses the *Token Endp* and the *UserInfo Endp*. These endpoints, however, are only used by the *FS API Server*. For this reason, these endpoints could also be assigned to an internal subnet where they are not accessible by the outside world.

The Federation Service has two other critical components that need to be mentioned. The *Federation Database* contains the state for all federations that currently being hosted. For each federation, this includes:

- Who is a member,
- What roles and authorization attributes have been defined,
- The assignment of these roles and attributes to members,
- Which services have been registered for use, along with their metadata, discovery policies, and access policies.

This database function for a Fed Service could be provided by an Oracle database or a simple MongoDB server. In any case, this database function could be on an internal subnet such that it is only accessible by the FS API Server.

Auditing and Accounting is another very important function. This is primarily implemented as a *logging service*. Multiple commercial logging tools are available that manage log formats, timestamping, and the disk space used by log files. Such a service can likewise be on an internal subnet.

Finally, we note that this Federation Service incorporates the basic OASIS eXtensible Access Control

Markup Language (XACML) security architecture through the use of a *Policy Administration Point (PAP), Policy Decision Point (PDP)*, and one or more *Policy Enforcement Points (PEP)*. After authenticated, when a Member makes a service request, the FS API Server forwards the request to a PEP, which asks the PDP to make an access decision based on the access policy for that service. If allowed, the request is finally forwarded to the service. Clearly the PAP/PDP/PEP machinery can be internal to the Federation Service. However, when a PEP forwards a request to an arbitrary federation service, it must be assumed that the request can be threatened by malicious actors.

While a single Federation Service is completely sufficient for running multiple federations, larger scale federations will require *multiple Federation Services* that are peers. This will be necessary not only for scalability reasons, but also due to possible desirable deployment models. It is anticipated that many organizations will want to deploy their own Federation Service that "peers" to the Federation Services in other organizations. This is illustrated in Figure 3.

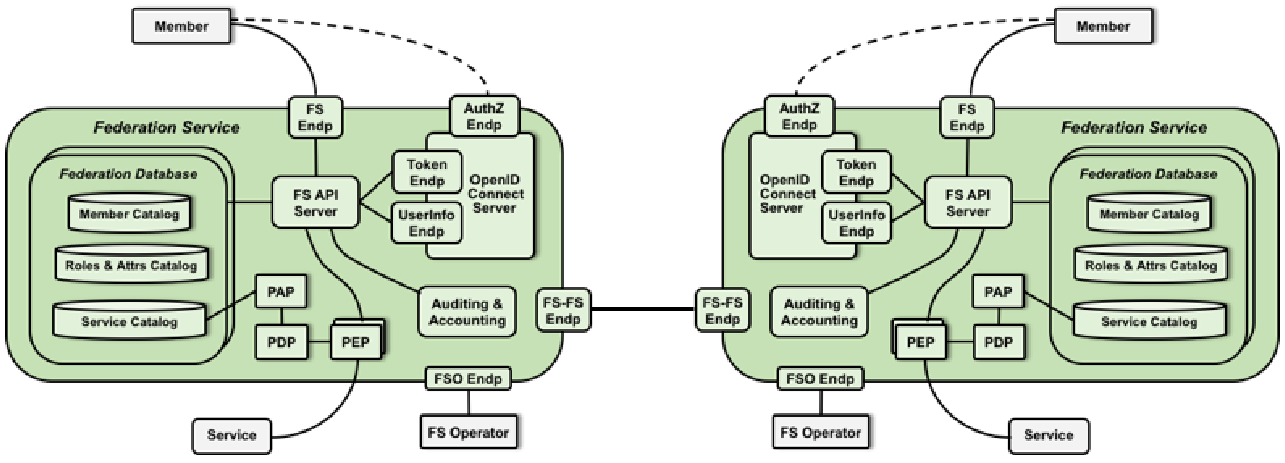


Figure 3. A FS-FS Interaction Threat Model.

The only addition to this model are the *FS-FS Endpoints* on the two Federation Services, and the link connecting them. As discussed in the CFRA, when two or more Federation Services peer, a major design decision is how this link is used: Which federation information is replicated among the Federation Services, which information is partitioned, and what queries must be passed among them. For the analysis described in this ER, this issue is out of scope. What data flow over this link would change what impacts a compromise might have, but the potential threat methods are the same.

These *static* architectural concerns must be considered in conjunction with the *dynamic lifecycle* of a federation and how those architectural components are used. To this end, the following interaction scenarios are examined:

- Authentication to a Federation Service
- FedAdmin/ResOwner Interactions
- User Interactions, and
- FS-FS Interactions.

All federation members must first authenticate to a Federation Service. This scenario exercises the entire OIDC Authorization Code Flow. As defined in the NIST CFRA, there are three fundamental, pre-defined roles that any federation member may have: *FedAdmin, ResOwner*, and *User*. The

FedAdmin can administer a federation, i.e., grant/revoke membership, roles, or attributes. A *ResOwner* can register services to be available in a federation, along with defining their discovery and access policies. For the purposes of this analysis, the FedAdmin and ResOwner threat analyses are combined since the threat models are structurally the same. Finally, a *User* is something of a default role that is able to invoke services, but not to administer the federation or register services. Hence, each of these fundamental roles exercises different functional capabilities in a Federation Service.

Based on the object model given in Figure 2 and Figure 3, separate Threat Dragon analyses can be performed for the four scenarios identified above. These four analyses that can be performed are part of one ThreatDragon analysis project. After completing an analysis, ThreatDragon can write the entire project results to a pdf report file. That final report file is given in Appendix A: Threat Dragon Report. A summary and discussion of each scenario analysis is presented here.

8.2.1. A Threat Analysis of Authentication to a Federation Service

Figure 4 shows the architecture defined in Figure 2 cast into a data flow diagram. This diagram identifies all actors and interactions when a Federation Member authenticates to a Federation Service using OIDC.

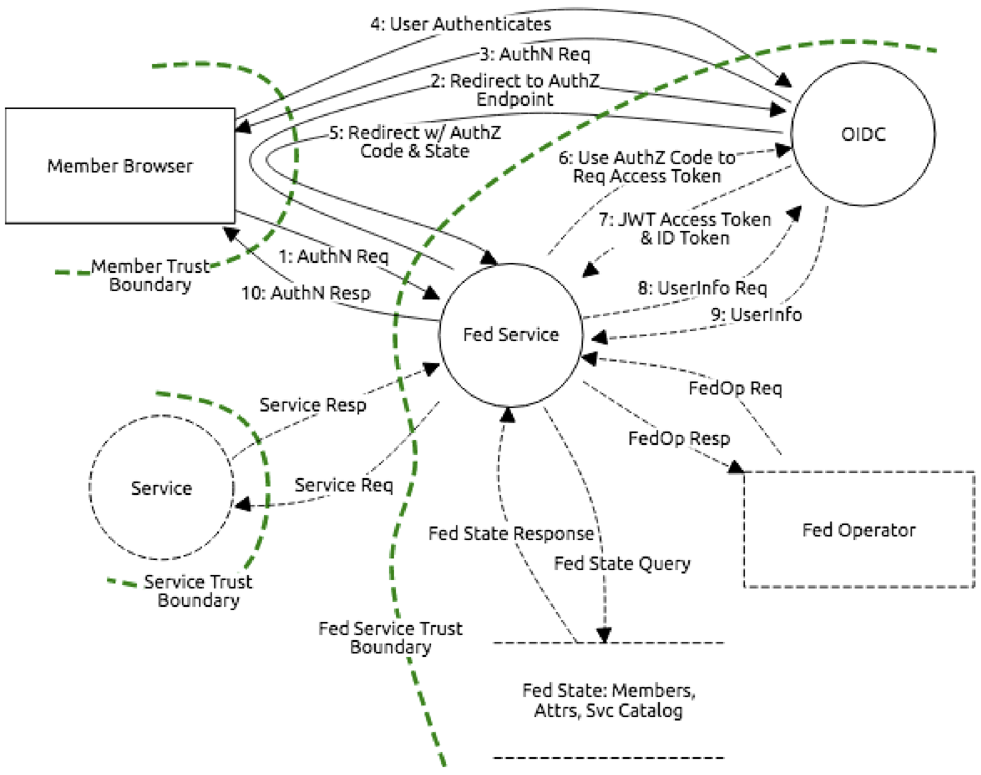


Figure 4. A Fed Service OIDC Authentication Threat Model.

Following the configuration shown in Figure 2, the Federation Service is shown behind the *Fed Service Trust Boundary*, along with the *OIDC Server*, *Fed State*, and the *Fed Operator*. A *Member* (and their browser) and a *Service* are shown behind their own Trust Boundaries. All dashed items are considered to be out-of-scope for the analysis documented in this ER. The *Service*, the *Service Request* and *Response* arrows are shown as out-of-scope since they are not involved in the authentication process. All other dashed actors and interactions are considered to be secure because they are behind the *Fed Service Trust Boundary*.

While this assumption may be flawed, it is based on currently established and accepted security practices, such as "securing" service endpoints by operating them on a different, internal subnet. Newer architectural approaches that do not require such assumptions are reviewed in Emerging Security Technologies.

As per the OIDC specification, the numbered arrows indicate the *OIDC Authorization Grant Flow*, the primary flow used by most applications to achieve authentication. (While this flow is called *authorization*, the first step is to establish identity, i.e., authentication.) When a Member asks to be authenticated (1), the Fed Service redirects the request (2) to the OIDC Server. The OIDC Server challenges the Member to authenticate (3) which the Member responds to (4). On success, the OIDC Server redirects back to the Fed Service (5) with a short-lived *Authorization Code* and a "state" value. The Fed Service uses this code to request an *Access Token* from the OIDC Server (6). The *Access Token* and *ID Token* are returned (7). These are *JSON Web Tokens (JWTs)*. JWT is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object (jwt.io). The *Access Token* can be used to request further information about the Member (8 and 9). Finally, the results of the authentication are returned to the Member (10).

There are two very important security aspects to this protocol. First, the Member does not divulge any secrets to the Fed Service. That is to say, the Member authenticates directly with OIDC. This could be simply by using account name and password, multi-factor authentication, biometrics, or other methods. Second, no tokens or other credentials are divulged to the Member. The Access and ID Tokens are exchanged directly between OIDC and the Fed Service. By having OIDC act as a trusted, third-party, the Member and Fed Service do not have to directly exchange any secret information.

A threat analysis of this protocol is not trivial. However, an extensive threat analysis has been performed for OAuth 2.0 cite:[OAuthThreatModel]. Since OIDC is essentially an integration of OpenID and OAuth 2.0, this threat analysis can be applied here. Threats and mitigations identified in cite:[OAuthThreatModel] are cited as such in the Threat Dragon report.

Threat Analysis Summary and Discussion:

The analysis described focuses solely on those actors that have *externally accessible endpoints* and the *interactions between them*. That is to say, this analysis focuses on interactions that *cross trust boundaries*. For the purposes of this analysis, the endpoints and interactions that are completely behind a trust boundary, i.e. within a *trust zone*, are considered to be secure.

Also, while some *Denial of Service (DoS)* attacks are considered here, DoS attacks against the communication bandwidth between a Member and any part of the Fed Service are considered out-of-scope. There is nothing federation-specific about such attacks. Such attacks can be mitigated without requiring any structural or behavioral changes to a Fed Service.

With these ground rules in mind, the threat analysis recommendations can be summarized as follows:

- All redirection URIs should be completely defined with no parameters and registered with the OIDC Server.
- Registered URIs should be checked to avoid Cross-Site Request Forgery (CSRF). CSRF is an attack

that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.

- Use stronger authentication methods, such as multi-factor authentication (MFA), random questions, CAPTCHAs etc.
- Only browsers that support the X-FRAME_OPTION should be allowed such that click-jacking using transparent iFrames can be avoided.
- The binding of an entity's externally accessible API endpoint to the entity's domain name should be validated, such as the Fed Service.
- Use TLS on all communication links.
- Rate-limit requests to avoid DoS attacks to the Fed Service or OIDC Server.

While very insightful, these threats and mitigations emphasize the fact that such analyses must be done by individuals that have a first-hand knowledge and understanding of how RESTful web services are built and how they work. For example, understanding the Cross Site Request Forgery vulnerability means knowing exactly how the redirection protocol works, and more importantly, how the protocol could be misused. Also, a threat analysis for any system should be done against specific implementations. Different implementations of the same standard could have different vulnerabilities. For example, under extreme malicious loading conditions, a service may commit a synchronization error that allows a response to be returned to the wrong requestor. For all these reasons, any serious analysis to identify threats and effective mitigations must be done for a specific implementation and deployment, staffed by knowledgeable persons, and properly funded.

8.2.2. A Threat Analysis of FedAdmin and ResOwner Interactions

In this scenario, a federation Member has already been authenticated to the Fed Service. As illustrated in Figure 5, Members are being considered that are *FedAdmins* and *ResOwners* as they perform administrative functions. Only the Member Browser and the Federation Service are involved with a single pair of request and response interactions. This threat analysis applies to both FedAdmins and ResOwners since the threat model is structurally the same. A FedAdmin makes requests to the FedService to manage a federation's members, e.g., to grant or revoke authorization attributes. A ResOwner makes requests to register service endpoints and manage the associated discovery and access policies.

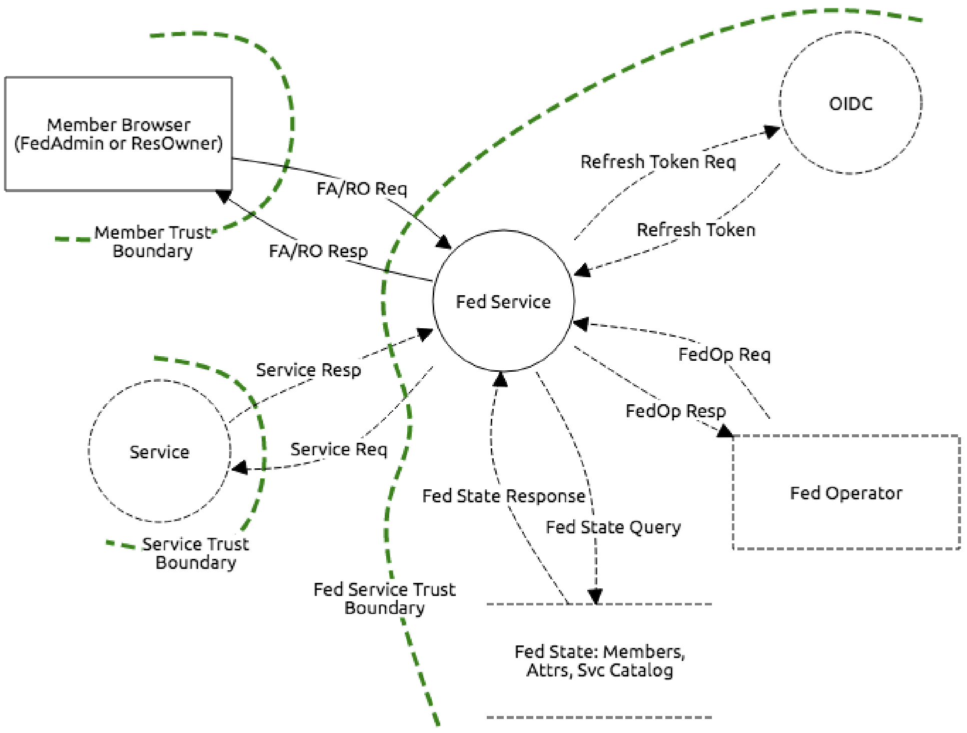


Figure 5. A FedAdmin and ResOwner Threat Model.

Threat Analysis Summary and Discussion:

This threat scenario is clearly much simpler than the authentication process. Since the members are already authenticated, the enables the FedService to associate the member with a trusted set of authorization attributes. Aside from mitigating possible DoS attacks on either the Member or the Fed Service, the key security goal is to mutually secure the communications between them, e.g., to use mTLS. While this is an entirely reasonable approach, the next question is what are the threat vulnerabilities of TLS and mTLS? This is discussed in Threat Analysis Observations.

8.2.3. A Threat Analysis of User Interactions

In Figure 6, members that are general *Users* are considered. Again, these Users have already been authenticated to the Fed Service for a specific federation.

General members can perform two functions: (1) query the federation's Service Catalog, and (2) invoke a Service. A federation's Service Catalog is maintained by the Fed Service. In response to a query, the Fed Service first applies the discovery policies defined by the ResOwners. The Fed Service then replies with a set of service metadata that (a) the User is authorized to discover and invoke, and (b) meets the User's query parameters.

The User can then invoke one of these services. In this deployment model, the Fed Service proxies the call thereby allowing the Fed Service to apply the Service's access policy. Upon granting access, the request is sent to the Service. The Service processes the request and sends a response which is forwarded to the User.

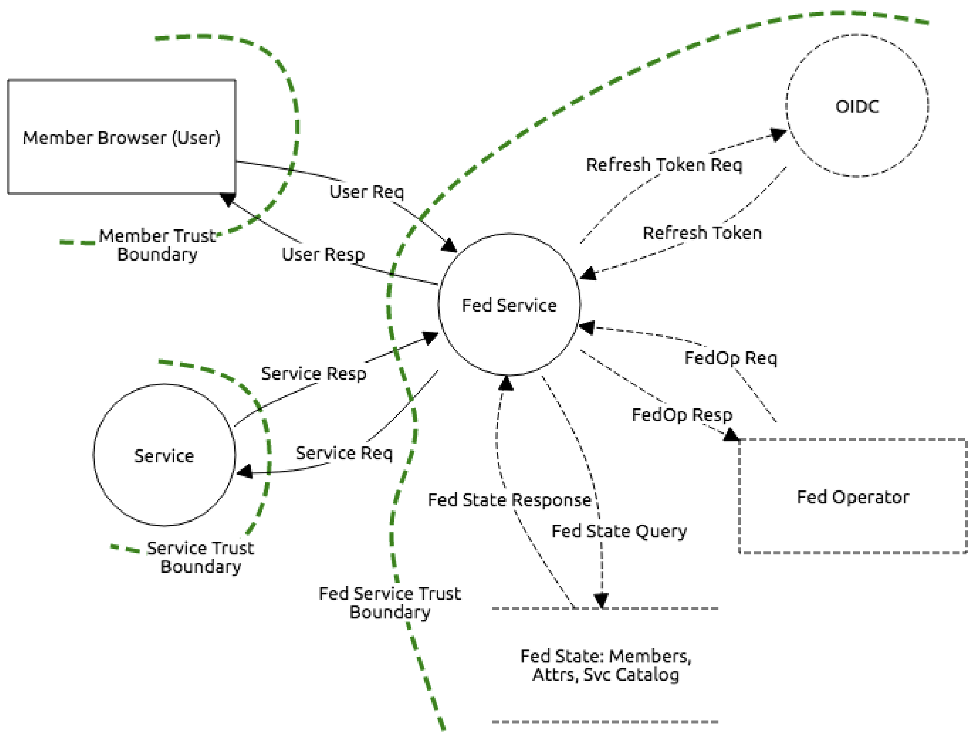


Figure 6. A User Interaction Threat Model.

Threat Analysis Summary and Discussion:

Threats to the interactions between general Users and the FedService are essentially the same as they are for FedAdmins and ResOwners. In this threat analysis, however, Services finally come into the picture. Services never ask the Fed Service for anything. Services simply respond to requests. As before, while DoS attacks are possible and can be mitigated, the key security goal is to mutually secure the communications between the Service and the Fed Service.

8.2.4. A Threat Analysis of FS-FS Interactions

The previous three threat scenarios have all dealt with a single Federation Service and its interactions with Members and Services. Figure 7 presents a final scenario that illustrates the interactions between two Federation Services.

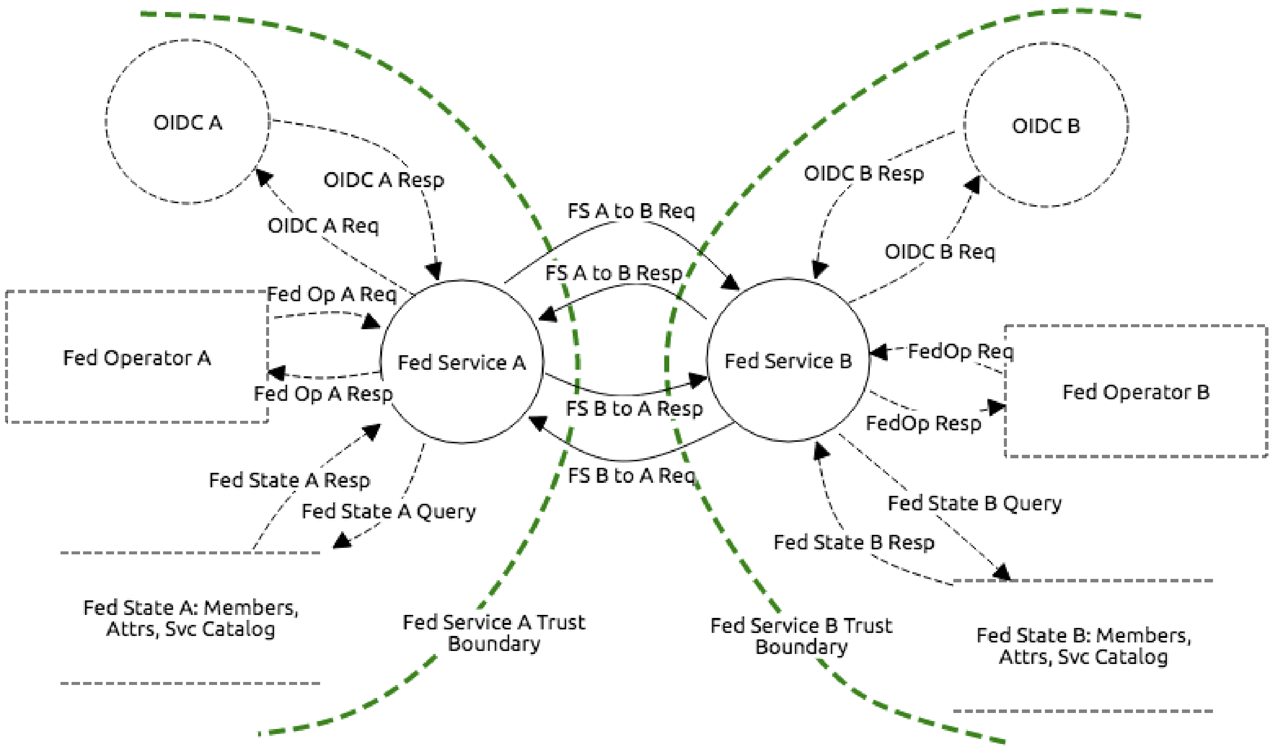


Figure 7. A Fed Service to Fed Service Threat Model.

As noted, each Fed Service can make requests to the other. As before, doing this securely depends on securing the communication channel. Fed Services do not authenticate to each other in the same way that Members authenticate to a specific federation. However, there does have to be a trust relationship between them, as part of the trust relationship between the organizations involved. Once this trust relationship has been established, the Fed Service Operators can configure their respective Fed Services to accept a connection from each other.

Threat Analysis Summary and Discussion:

As part of this simple deployment model, the identity of the Fed Services involved is being established out-of-band. Aside from mitigating possible DoS attacks, as before, the key security goal is to mutually secure the communication link between the two Fed Services.

8.3. Threat Analysis Observations

This structural approach to threat analysis emphasizes several points:

- *Authentication is critical.* Once a trusted identity has been established, the identity can be associated with authorization attributes that determine what an entity can do. In many ways "Who you are" is "What you can do".
- *Communications must be protected at all times.* Whenever communication must cross a trust boundary, it must transit a zone of *implicit distrust*. The common security practice today is to secure the communication channel as it crosses that zone of distrust. However, to assume that everything within a trust boundary is, in fact, secure carries its own risk. This is the central motivation for developing Zero Trust Architectures.
- *Many security threats are "idiosyncratic" to specific implementations.* That is to say, many exploits exploit an implementation flaw or oversight, such as failing to check a registered

redirection URL. A structural threat model for a specific system deployment can identify which tools are being relied upon. Once that is done, the threat analyst must look for threats against those specific tool implementations. The actual tool implementers are in the best position for doing such analyses.

Threats can also arise from how a standard is defined. TLS was cited repeatedly as a mitigation strategy for communication threats. However, the POODLE (Padding Oracle On Downgraded Legacy Encryption) attack cite:[TLSAlert2014] is considered to be fault of the protocol. When a client initiates a TLS handshake, it sends a list of supported SSL/TSL versions. A Man-in-the-Middle attacker could impersonate the server and try to get the client to agree to downgrade to SSL 3.0. SSL 3.0 has a vulnerability in the *Cipher Block Chaining* mode. Cipher blocks can have padding to reach a fixed length that an attacker can manipulate. By watching the server responses to this padding, the attacker can eventually determine how to decode the encrypted block. Since the POODLE attack was considered to be inherent in the protocol, the recommended mitigation was to simply disable the use of SSL 3.0.

This one threat analysis made definite assumptions about the deployment structure, and specifically where the trust boundaries were. This clearly affects what vulnerabilities might exist. In OIDC, impersonation attacks are possible if the ID Tokens are not validated. (ID Tokens are used in OIDC, but are not part of the OAuth standard.) When OIDC is within a trust boundary, there is no problem. However, if outside the trust boundary, then the client (i.e., the Fed Service) must validate that the ID Tokens are properly signed by the OIDC Issuer. Hence, mitigations must always be in place concerning trust boundaries.

The above comments serve to emphasize the threat analysis presented here should not be considered complete, definitive nor authoritative. This analysis does, however, point to many places where "recursive" threat analyses could be done on all tools and standards used in a federation deployment. The limitations of these threat analyses to identify and mitigate vulnerabilities are also motivating the development of new architectural approaches that will simply eliminate some vulnerabilities, while making others more manageable. This is the topic of the next chapter.

Chapter 9. Emerging Security Technologies

The previous chapter examined the security threats and mitigations of a specific deployment of a CFRA-based federated environment using established security approaches and tools. That approach was essentially based on (a) establishing identity, and (b) securing communications channels, specifically when *trust boundaries* had to be crossed. The need to establish a trusted identity was critical since identity was associated with authorizations. Securing communications channels at the network level is the established thing to do because it is the obvious thing to do. This is a manageable, *infrastructure-based*, security approach. However, are there other ways of establishing identity and secure communications that are less dependent on a particular infrastructure and generally more secure — especially in distributed, federated environments? To this end, other approaches to managing identity and trust are examined.

9.1. Identity and Credential Management

All security architectures share some fundamental requirements. Federated environments inherent all of these, while introducing additional considerations. These fundamental requirements are nicely organized in the NIST *Digital Identities Guidelines*, SP 800-63-3 cite:[NIST800-63-3]. These are briefly reviewed here prior to discussing federation-specific concerns.

9.1.1. Identity Proofing

Identity proofing is the process of verifying who a *subject* or *applicant* is when first establishing them as a valid *subscriber* in an identity system. To be more concrete, this is when a *Credential Service Provider (CSP)* associates an applicant with a digital identity that is documented with one or more digital credentials.

The NIST Digital Identities Guidelines cite:[NIST800-63-3] defines three *Identity Assurance Levels*:

- **IAL1:** "There is no requirement to link the applicant to a specific real-life identity." That is to say, the applicant can self-identify and self-assert any attributes about themselves. This is a common approach for many on-line services, such as chat rooms.
- **IAL2:** "Evidence supports the real-world existence of the claimed identity and verifies that the applicant is appropriately associated with this real-world identity." This requires some type of independent verification of the applicant's identity claims.
- **IAL3:** "Physical presence is required for identity proofing." The applicant must appear in person and have their identity claims independently verified by an authorized and trained representative of the CSP.

These Identity Assurance Levels can all be applied in federated systems.

9.1.2. Authentication

When authenticating a *user* or *claimant*, the claimant must demonstrate that they possess and control one or more *authenticators*, such as a username, credential, token, etc., that are used to prove their identity. The use of a single authenticator, such as account name and password, has been common for decades. However, stronger security can be achieved by using multiple authenticators in conjunction, such as *Multi-Factor Authentication (MFA)*.

Authenticators can be broadly categorized into:

- *Something you know:* This could be an account name and password, an RSA passcode, or some other type of digital challenge.
- *Something you have:* This can be an ID badge, a Common Access Card, an RSA SecurID Hardware Token, a cryptographic key, or even a mobile device for presentation of a challenge.
- *Something you are:* This can be a fingerprint, retina scan, facial recognition, or some other biometric data.

A common MFA approach uses account name and passcode, in conjunction with an RSA fob that presents a PIN code with limited time validity, for example 60 seconds.

The NIST Digital Identities Guidelines cite:[NIST800-63-3] defines the following *Authenticator Assurance Levels*:

- **AAL1:** "AAL1 provides some assurance that the claimant controls an authenticator bound to the subscriber’s account. AAL1 requires either single-factor or multi-factor authentication using a wide range of available authentication technologies. Successful authentication requires that the claimant prove possession and control of the authenticator through a secure authentication protocol."
- **AAL2:** "AAL2 provides high confidence that the claimant controls authenticator(s) bound to the subscriber’s account. Proof of possession and control of two distinct authentication factors is required through secure authentication protocol(s). Approved cryptographic techniques are required at AAL2 and above."
- **AAL3:** "AAL3 provides very high confidence that the claimant controls authenticator(s) bound to the subscriber’s account. Authentication at AAL3 is based on proof of possession of a key through a cryptographic protocol. AAL3 authentication SHALL use a hardware-based authenticator and an authenticator that provides verifier impersonation resistance; the same device MAY fulfill both these requirements. In order to authenticate at AAL3, claimants SHALL prove possession and control of two distinct authentication factors through secure authentication protocol(s). Approved cryptographic techniques are required."

While all of these authentication methods and Authenticator Assurance Levels can be applied in federated environments, federation raises the additional possibility of *federated identity management*. Federated identity entails the ability of a user to use identity credentials issued by an Identity Provider (IdP) in one domain to access resources in another domain. Doing this, however, depends on credential management and trust. This leads to the next discussion topic.

9.1.3. Federated Credential Management

Managing the use of credentials across identity domains, and the identity attributes that might be carried on them, i.e., *released*, is a significant issue in federation. To give some structure to this issue, the NIST Digital Identity Guidelines cite:[NIST800-63-3] define three *Federation Assurance Levels*. These refer to the strength of an identity assertion issued by an Identity Provider (IdP) in a federated environment when communicating that information to a Relying Party (RP):

- **FAL1:** "Allows for the subscriber to enable the RP to receive a bearer assertion. The assertion is signed by the IdP using approved cryptography."
- **FAL2:** "Adds the requirement that the assertion be encrypted using approved cryptography such that the RP is the only party that can decrypt it."
- **FAL3:** "Requires the subscriber to present proof of possession of a cryptographic key referenced in the assertion in addition to the assertion artifact itself. The assertion is signed by the IdP and encrypted to the RP using approved cryptography."

While these assurance levels are useful, the added dimension of federation that must be emphasized is that the RP and IdP will be in different identity domains.

A key function of a federated environment is to enable an RP in one domain to *trust* and *understand* the credentials issued by an IdP in another domain. The RP must trust that the credential:

a. Legitimately "belongs" to the subject,

b. Has integrity — it has not been maliciously modified, and

c. If privacy is desired, cannot be snooped to determine the identity attributes being released from the IdP to the RP.

Once this trust is established, the RP must have a common understanding about what the identity attributes mean, such as how they can be used to evaluate an access policy and make a valid access decision. A fourth requirement can be added from the User’s perspective:

d. Only the minimal identity attributes are released to enable the RP to make a valid access decision.

Item (a) above requires a credential belongs to a user. Please note, however, FAL1 is based on the concept of a *bearer assertion*. This is something the user or subscriber (the *bearer*) presents to the Relying Party as *sufficient proof of identity*. Bearer assertions, or bearer tokens, are vulnerable to man-in-the-middle or snooping attacks, and simple device theft. If an attacker acquires a bearer token, by any means, they attain all the rights of the legitimate user.

The need to avoid the limited security offered by bearer tokens and achieve the three trust requirements described above has motivated the development of various security architectures, such as Public Key Infrastructure (PKI) cite:[PKI]. PKI utilizes a *Certificate Authority* that issues signed and encrypted certificates that are only valid for the intended subject. PKI is widely used with the established X.509 certificate format cite:[X509]. When a User requests service and presents a certificate, the RP can verify the certificate with the Certificate Authority. Importantly, this type of approach also addresses Items (b) and (c).

While PKI is widely used and considered quite secure, another design concept that increases security even further is to not divulge credentials directly to the user at all. PKI certificates can be directly installed on a user’s browser or device. As discussed in A Threat Analysis of Authentication to a Federation Service, OpenID Connect (OIDC) cite:[OIDC] avoids doing this. When a User requests service from an RP, the RP can *redirect* the User to authenticate directly to an *OIDC Provider (OP)*. On successful authentication, the User is redirected back to the RP with an *authorization code*. The RP uses this code to get an *ID Token* and *Access Token* directly from the OP. The User never sees their ID and Access Tokens, and has no control over them.

This architectural approach for OIDC, however, was actually developed to support *Federated Identity Management* (FIM). On the surface, the goal of FIM is to enable a User A from Organization A to access resources at Organization B using their identity credentials from Organization A. However, the market-driven usage scenarios that motivated this, however, were quite limited in scope. One example is enabling a User's Facebook account to access photographs on the User's Snapchat account. This does require that Snapchat trusts the User's IdP and having the User authenticate to approve the access by Facebook.

While there is certainly a mass-market use of federated identities, it is far from a general federation capability. First, it assumes that a User knows about and can invoke any service that is available on the Internet. There is no notion of a Service Owner being able to manage the discovery of their services through policy. The Service Owner must determine *where the User is coming from*, and *who is the User's IdP*. The Service Owner must then determine if they know and trust the IdP. Usually this means that Service Owner must maintain a list of IdPs that they trust. When making a service request, the User must select an IdP from the Service Provider's approved list — which hopefully includes the User's IdP. Even when this is done, there is no guarantee that the User's IdP can provide the attributes necessary for the Service Owner to make an access decision. This may be because the IdP does not know or understand what type of attributes are being requested, or because the IdP does not wish to release those attributes for some organizational reasons. For all of these reasons, simple FIM is insufficient for managing general federations.

The NIST CFRA addresses all these issues by enabling "purpose built" federations to be defined and instantiated. By considering a federation to be a *virtual administrative domain*, it is possible to manage *membership* in a federation, along with the *roles*, *attributes* and *policies* necessary to manage its operation. That is to say, defining the "rules of the road" for members and establishing an expectation of what it means to be a member in good standing is possible. There is no question about what the roles and attributes mean, or how they are used by the federation's service owners. What the NIST CFRA does is provide a *scoping mechanism* whereby existing authentication and authorization tools can be leveraged to instantiate any number of federation instances. These instances can be independently managed according to their own governance requirements.

9.1.4. Attribute Release and Verifiable Credentials

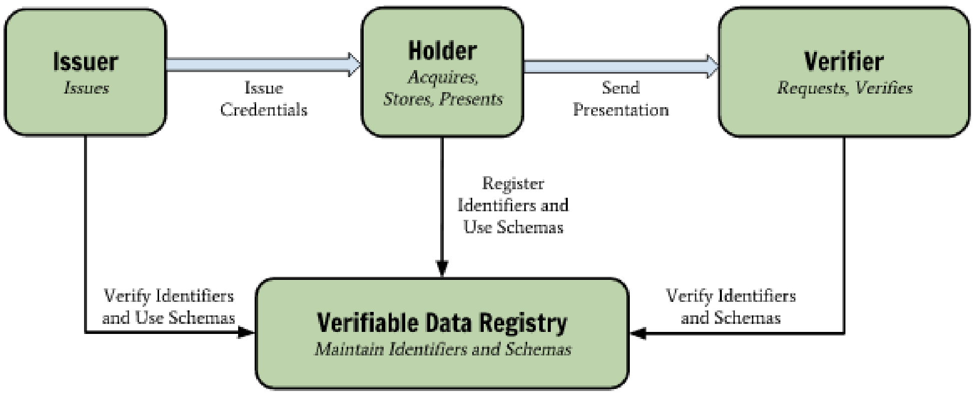
The issue of *attribute release* — Item (d) above — has already been mentioned several times. This is a significant issue for many existing federation systems and is examined here in more depth.

While thinking of a credential as documenting everything there is to know about a User is easy, an RP may only need one or two attributes to make a decision. Aside from not sending extraneous information, there may be attributes that a User (or their organization's IdP) may not want to divulge to an RP that does not need to know. In a single domain, attribute release is less of a problem since information is "all within the family", i.e., the same trust domain. In a federated environment, however, attribute release becomes more of an issue and must be explicitly managed.

When an organization is must managing their own environment, the attributes used and how they are encoded into credentials is typically not an issue. However, if such an organization — or some subset of their users — later wishes to federate with another organization for any purpose, their IT security architecture was probably not designed with this use case in mind. From an organizational governance perspective, they may have been unprepared to share identity attributes with other organizations, of rather, undecided about how to do so. Organizational IdP administrators may be faced with competing requirements concerning how to expose attributes to external RPs. The only tractable solution for many administrators will be to release nothing.

The NIST CFRA avoids this problem by defining a virtual administrative domain where attribute release is simply not a problem. The goals and purpose of a federation are well-defined. Hence, the attributes and how they are used are well-defined. The necessary attributes to make discovery and access decisions are well-defined. This means their release can also be well-defined.

There are, however, other ways of addressing the attribute release issue that could also be used in federated environments. The approach taken by *W3C Verifiable Credentials* cite:[W3C_VCs] enables the user to directly control their credentials and how attributes are released. This design approach is illustrated in Figure 8.



Issuer
Issues
Issue Credentials
Holder
Acquires, Stores, Presents
Send Presentation
Verifier
Requests, Verifies
Register Identifiers and Use Schemas
Verify Identifiers and Use Schemas
Verifiable Data Registry
Maintain Identifiers and Schemas
Verify Identifiers and Schemas

Figure 8. The W3C Verifiable Credential Model.

While *Issuers* (IdPs) continue to issue credentials to *Holders* (Users), the Holder maintains all of their credentials in a repository that can be called a *wallet* (not illustrated). Credentials are only valid for a specific *subject*, which may the Holder but could be another entity associated with the Holder. When requesting service from a *Verifier* (Relying Party), the Holder creates a custom credential *presentation* built from any of the credentials in their wallet. Hence, the Holder has complete control over their attribute release when interacting with Service Providers. The credentials and presentations are constructed and signed such that they can be *verified* through a *Verifiable Data Registry*. Hence, *Verifiable Credentials are not bearer tokens*, even though they can be used as bearer tokens by leaving the subject field in a credential presentation empty.

Verifiable Credentials (VCs) handle credentials in a way that is very different from third-party IdPs. VCs were deliberately patterned to work the same way as *physical* credentials. A typical person will have a wallet with physical credentials, such as a Driver's License, credit cards, auto club card and so forth that can be presented for identification and receive services. Of course, all this depends on a set of public Issuers that can be consulted to verify credentials. Hence, VCs are a close parallel to how humans go about their business right now. For this simple reason, VCs might gain market adoption in the coming years.

However, this design approach also makes it difficult for controlling how a user may use their credentials in subsequent presentations. For many application domains, after granting a credential, the Credential Issuer may wish to stipulate that the User may only use the credential in presentations to entities on an approved "white list", and must never use them in presentations to entities on a "black list". Such requirements are not known to be handled at this time.

The VC approach to credential management is actually quite compatible with the NIST CFRA approach to federations. A fundamental tenet of the CFRA model is that a federation is a virtual administrative domain whose governance should be defined and agreed upon by the potential members prior to instantiation. That is to say, the policies used to manage resource discovery and access are defined over a set of common, well-known attributes. In the VC case, these attributes can be provided by public issuers. The members simply need to agree on the Issuers and attributes to be used. A Federation Service, however, will still need to maintain a service catalog for each federation whereby discovery policies can be enforced. Likewise, some form of Policy Decision Points and Policy Enforcement Points would have to be used whereby access policies can be enforced.

These architectural concerns for managing identity, credentials, and the attributes they carry, are certainly important. However, these mechanisms must be deployed in conjunction with trust and understanding, i.e., semantic interoperability.

9.2. Trust

Trust is an immense subject cite:[Cho2015,Braga2019]. As a concept, trust is relevant across many fields of human endeavor, such as sociology, psychology, economics, game theory, business, philosophy, international relations, and jurisprudence, just to name a few. For this reason, trust is examined in a little more depth than in the top-level Trust definition.

9.2.1. Defining Trust

Trust can be broadly categorized in a hierarchy cite:[Braga2019]:

- *Cognitive Trust:* Trust derived from the mental processes of an intelligent entity.
- *Social Trust:* Trust derived from a human social network built on pair-wise relationships.
- *Information Trust:* Trust that information shared over an information network is trustworthy; i.e., meets some definition of objectivity, reliability, integrity, accuracy, fairness, etc.
- *Communication Trust:* Trust that a system of nodes and links will demonstrate a set of quantifiable metrics, such as connectivity, capacity, latency, fault tolerance, etc.

The bottom two layers in this hierarchy can be termed *computational trust*. For most systems, establishing computational trust means that a *set of requirements have been satisfied*. Depending on the use case scenario and the stakeholders' risk tolerance, that necessary set of requirements will vary. In any case, trust can be said to be a binary, yes/no decision. (While *reputation systems* have been devised to manage *varying degrees* of trust, these are out of scope for this engineering report.) This ER chapter explores what trust requirements may be for federated systems and how they can be realized.

The hierarchy above can be further refined into the concept of *trust models* cite:[OASIS_Trust_Guidelines]. Linn, et al., define the notions of *Business Anchor* and *Trust Anchor*. A business anchor "represents an entity with which its holder has a direct business relationship." That is to say, some type of business agreement has already been established between two human organizations. A trust anchor "represents an entity and key that the anchor’s holder has determined to trust directly for cryptographic authentication purposes." This allows two organizations "connected" by a business agreement to establish trusted electronic communications based on cryptographic methods. Human organizations can maintain *Business Anchor Lists* and *Trust Anchor Lists* whereby sets of trust relationships can be managed. Based on these concepts, trust models can be categorized as follows:

- *Pair-wise Trust* is based exclusively on bilateral business agreements. While fundamental, pair-wise trust has limited scalability.
- *Brokered Trust* relies on active intermediaries that can *construct a path* of trust relationships from one entity to another. While certainly more scalable, brokered trust implies that all parties must agree to a set of rules governing the use of *transitive* trust relationships.
- *Community Trust* is based on a shared membership in a community. This implies a common, well-known set of rules whereby trust can be said to be established. In essence, establishing community trust implies, that an all-to-all set of pair-wise trust relationships has been established among the community members.

In practical scenarios, these trust models need to be managed in the context of a *trust framework*. This is especially true for brokered/transitive trust and community trust where establishing paths of *trust chains* or *community trust requirements* must be managed.

9.2.2. Trust Frameworks

To this end, Temoshok and Abruzzi present a guide for developing trust frameworks for identity federations cite:[NIST_Trust_Frameworks]. While focusing on identity federations, this guide is directly relevant to general federations, as defined by the NIST CFRA. As the basis of authentication, this work takes the approach of OIDC (as illustrated in the Threat Dragon models). When a User requests service from a Relying Party, a pair of redirections is used whereby the User authenticates to the IdP, and the IdP provides the authentication status directly to the RP. The NIST CFRA augments such identity federation by integrating it with *resource federation*. That is to say, the resources in a federation are managed according to *discovery* and *access policies* based on the identity roles and attributes that are well-known within a federation. Hence, this ER section extends the trust framework approach presented in cite:[NIST_Trust_Frameworks] to general, CFRA-based federations.

NIST Trust Frameworks govern the interactions among three main actors:

- Federation Administrators (aka, IdPs or Credential Providers),
- Relying Parties (aka Resource Providers), and
- Users (aka, Members).

A Trust Framework encompasses a set of rules governing how:

- Identity management responsibilities are conducted,
- Identity information is shared,
- Shared identity information is used,
- Identity information is secured and protected,
- Specific roles in the federation are performed, and

- Legal and liability issues are managed.

To accomplish this, a Trust Framework is comprised of four components:

- *System Rules* (above) which govern members,
- *A Legal Structure* which identifies the rights, responsibilities, and liabilities of membership and participation in a federation;
- A way of *Establishing Conformance* of individual members; and
- A way of *Recognizing that Conformance* and communicating the fact of conformance among the federation members.

In cite:[NIST_Trust_Frameworks], these components are discussed specifically for identity federations. Their implications for general federations are now considered.

9.2.2.1. System Rules

System Rule includes issues of *member registration, identity proofing* and *credential management*. These have already been extensively discussed. The issue of *privacy* is clearly related to the issue of attribute release, which was discussed concerning credential management. Aside from mechanisms for managing the exposure of member's attributes to resource providers, different federations may have different privacy concerns and requirements. As noted in cite:[NIST_Trust_Frameworks], a federation should have a clearly defined *privacy policy* concerning how member attributes are handled. This emphasizes the importance of CFRA-based general federations being able to define their own set of attributes. These do not have to include any personal information about a member and may only have meaning within the federation. Of course, a link from a member's federation identity to their "home" or "personal" identity would always have to be maintained.

Temoshok and Abruzzi raise the issue of federation members being tracked, allowing a Credential Service Provider to "build a narrative" about a user that the user did not consent to. Note that this is another potential benefit of Verifiable Credentials. A credential Holder may use a credential, such as presenting the credential to a Verifier, without the Issuer necessarily knowing about the credential usage. While Verifiers could track their customers, a Credential Issuer would not be able to track Holders across multiple Verifier interactions. With CFRA-based federations, we note that even if a credential system such as OIDC is being used, the Credential Issuer should only be able to track the federation member within a given federation. From a technical perspective, it would be possible for a single CFRA-based Federation Service to track a member's behavior across multiple federations if those federations were being hosted on the same Fed Service. This is a distinct policy issue concerning CFRA-based implementation that should be addressed. Outside of a federation, for a Federation Service to track a member's behavior would certainly not be possible.

Security and *Data Handling* requirements for identity federations are how discussed. General federation clearly inherits these requirements. The mechanisms to achieve communication security are extensively discussed in the Threat Dragon analyses. In addition to handling identity data, general federation need to securely manage and minimize the volume of service data is exchanged, such as service endpoints, metadata, discovery and access policies.

The final *System Rules* topic is Technical Specifications. Having a "common set of technical protocols and standards" is beneficial and even necessary for all distributed systems. Developing and establishing standards for CFRA-based federations is the goal of the IEEE P2302 WG cite:[P2302]. While this is still work-in-progress, P2302 has defined a core API single, Third-Party CFRA deployments, and simple peer-to-peer deployments.

Beyond the API and protocol work, P2302 is also working on the notion of a *formal definition methodology* for federations. A formal federation definition would include information such as the "Rules of the Road" for participation: What the purpose and goals of the federation are, what types of services and data are to be shared, what type of attributes will be needed and available to manage discovery and access policies, how the federation will be governed, who will be the FedAdmins, and so forth. This would be similar to an *Acceptable Use Policy* that defines the roles and responsibilities of federation membership. While this is also work-in-progress, candidate approaches could be *Fed-ML* or an *OWL-Fed*. These could all be part of the System Rules for a general federation Trust Framework.

9.2.2.2. Legal Structure

Small-scale federations can be quite informal requiring nothing more than verbal agreements among friends. This is true for both identity federations and general federations. However, as federations become larger and involve more significant amounts of resources, the participants may feel that *legal agreements* are a necessary part of the federation trust framework. When a federation represents a significant investment of time, money and people, then the rights and obligations of federation members must become legally binding. These rights and obligations could include an Acceptable Use Policy, as discussed above, but could involve much more.

All federations must be in *regulatory compliance*. Federations operate within governmental jurisdictions that may define legally binding compliance requirements. Examples include the Child Online Privacy Protection Act, the Financial Services Modernization Act, and the Health Insurance Portability and Accountability (HIPPA) Act, which all govern how data must be protected and how it can be used. Such compliance requirements must be reflected and observed as part of a federation’s legally binding System Rules.

Something that is explicit in a general federation, and not in an identity federation, is the potential financial cost of the deployment and use of a service with a federation. If the cost of operating a federated service is not covered "by other means", then the service owner may wish to establish a *pricing model* whereby costs can be recouped, or even a profit made. Any such pricing models must be part of the System Rules in a Trust Framework. This must be done in conjunction with an *accounting mechanism*. Such a mechanism must log and collate all the necessary transaction events, such that an itemized bill can be presented at the end of a billing period. While identity federations may enable services to be used, costing and billing is simply out of scope for them.

Legal agreements typically address the issue of *risk and liability allocation*: Who is responsible if something goes wrong. In an identity federation, if an identity credential is mishandled or identity attributes are erroneously associated with the wrong credential, then an erroneous access decision could be made. Access could be granted when it should not have, or a legitimate access request could be denied. In either case, one party or another could feel they have suffered a loss. This loss could be financial or exposure of private, sensitive information.

This is certainly an issue in general federation, as well. In addition, though, a general federation is managing service endpoints, metadata, and discovery/access policies. If these are mishandled, these could also result in an erroneous discovery or access decision, and some party could feel they have suffered a loss. While every precaution to be taken to ensure that a Federation Service does not "make mistakes", a Federation Trust Framework should clearly identify what happens when one does.

The final legal topic is *Multilateral Agreements*. One of the fundamental tenets of a federation is that rather than having a set of differing, potentially conflicting, bilateral trust agreements, there is a single, multi-party agreement that all participants agree to. When agreeing to such a contract, all parties have a consistent understanding of the "Rules of the Road" for participating in a federation. In a general federation, the scope of such agreements is expanded beyond that of just an identity federation. Beyond the handling of identities, a legal, multilateral agreement for a general federation can include the types of services to be shared, their metadata, and roles and attributes governing how the services can be used.

9.2.2.3. Establishing Conformance

While defining a trust framework that members must agree to is essential, some federation scenarios will demand that actual compliance to those System Rules must be established through a concrete assessment process. The need for such assessments is the same for both identity and general federations, although the scope of such assessments will be greater for general federations. Temoshok and Abruzzi identify three aspects of assessing conformance:

- *Self-Assessment:* Self-assessment is the first step beyond simply having a signed membership agreement. As an internal process, this can require the least overhead.
- *Third-Party Assessment:* Third-party assessment raises the level of confidence in the assessment results. A third-party assessor should be independent of the organization being assessed and the federation itself, such that the assessor has no motivation to bias any results one way or the other. While such independence is valuable, it does require a greater investment of resources.
- *Formal Audit:* A formal audit is a standardized compliance evaluation method. While a federation's System Rules may define the trust criteria, a formal audit defines how compliance will actually be assessed. A Trust Framework that requires formal audits could also define when and how often audits are done, and with how much notice.

While assessing compliance is certainly important, any assessment method must be driven by specific criteria. Hence, a formal method of defining all aspects of a federation needs to be developed whereby a formal method of assessing conformance can also be defined.

9.2.2.4. Recognizing Conformance

Once compliance has been assessed, there must be a trusted mechanism whereby the results of that assessment can be communicated to, and thereby recognized by, other stakeholders in the federation. General federations use the same mechanisms as just identity federations. Temoshok and Abruzzi identify the following:

- *Registries and Listing Services:* This can be as simple as maintaining a web site where assessment results are reported. Reported results should also identify the assessment methodology and whether it was a self-assessment or third-party assessment.
- *Compliance Marks:* A compliance mark is a visually recognizable image or icon that denotes the

satisfaction of a specific set of compliance requirements under a known methodology. The use of a compliance mark is strictly controlled because of the achievement it is meant to represent.

- *Electronically Verifiable Marks:* Beyond simply a visual icon, an electronically verifiable mark can have an embedded URL that links to a registry or the organization that issued the compliance mark.
- *Cryptographically Signed Compliance Marks:* Called Digital Certificates or *Trustmarks* by Temoshok and Abruzzi, these compliance marks are cryptographically signed and can be linked to a *root of trust*, such as a PKI Certificate Authority. While offering the highest level of trustworthy recognition, such compliance marks have the highest overhead in terms of the necessary infrastructure that must be operated.

An important distinction in these recognition methods is whether they are done at the human organization level, or whether they are automated somehow as part of the IT infrastructure. Identity federations enable a Relying Party to trust credentials from different IdPs. While both the RPs and IdPs may want to know that all entities are in compliance with a set of requirements that denote trust, identity federations do not have an integral mechanism verifying signed compliance marks. On the other hand, a set of CFRA-based Federation Services supporting general federations could also be used to manage sets of signed compliance marks that could be requested and inspected on-demand. Federation Services could also be used to help automate the assessment process, but this raises the issue of how much independent, third-party humans should be "in the loop".

9.2.2.5. Examples of Federation Trust Frameworks

One direct benefit of Temoshok and Abruzzi's work identifying and organizing the components of trust frameworks is that this organization can be used to evaluate existing trust frameworks that have been developed. These trust frameworks have been developed in an *ad hoc* manner to serve their user communities. Nonetheless, they represent pioneering work in this area.

- *Interoperable Global Trust Federation (IGTF):* IGTF cite:[IGTF] started as the International Grid Trust Federation in 2004, during the heyday of grid computing. The goal of grid computing was to enable international science projects to share data and resources. Trust among the participants was established by *self-audit and peer review*. Once a participant demonstrated that their PKI Certificate Authorities were configured and operated according to IGTF guidelines, other participants would trust certificates signed by their CA.
- *Federal Bridge Certificate Authority (FBCA):* The FBCA cite:[FBCA] developed some years later, provides a very similar capability. The FBCA provides a way to link or bridge CAs in the Federal PKI (FPKI) network that may operate under different certificate policies. FBCA can map certificate policies thereby enabling them to be validated with the Federal Common Policy Certificate Authority (FCPCA) root certificate. CAs with certificates signed by the FBCA 2016 are said to be *cross-certified* and have a trust relationship established with the FPKI. This relationship is audited annually for conformance to the certificate policies.
- *InCommon:* Starting around the same time as IGTF, InCommon cite:[InCommon] is a federation-based collaboration systems that is run as part of Internet2 cite:[Internet2]. Beyond just enabling trust among PKI certificate authorities, InCommon maintains a *federation metadata* file. This is an XML file of *identity providers* and *service providers* from different universities and organizations, enabling these organizations to collaborate. InCommon operates a

Registration Authority that vets IdP and SP information before adding it to the metadata. InCommon also has a *resolution process* whereby disputes between participants can be addressed.

Other trust framework examples are possible. However, the above sampling serves to illustrate some differences. Each has a different scope for what they are providing, and have a different level of automation, albeit scant, for dealing with their System Rules and Conformance processes. IGTF is more of a loose federation with self-audit and peer review. FBCA actually operates a root CA that serves as the trust anchor for the entire trust federation. InCommon goes beyond this by maintaining a flat file of IdPs and SPs from which different federated environments can be built, but without any inherent mechanisms for keeping those environments separate and private.

Such differences underscore the need to develop general federations whose governance and trust frameworks are based on standards and automation as much as possible. Furthermore, it must be possible to tailor such deployments to various federation *profiles* that meet the different collaboration requirements of the participants. Further discussion about these systems and issues can be found in cite:[CFRA2020,Chadwick2014,Lee2019].

9.2.3. Trust Framework Approaches

Trust is an immense subject. Trust frameworks and their approaches are an equally immense subject. There are, however, several tools and efforts being developed. These directly support the formal definition of general federations, their legal structure, and the establishment and evaluation of conformance. Such tools are necessary for creating secure and trustworthy federations.

9.2.3.1. Trustmarks

Trustmarks were originally developed for e-commerce, such as BBBOnline cite:[BBBOnline], VeriSign cite:[VeriSign], and TRUSTe cite:[TRUSTe]. Since these were targeted for e-commerce, they have a specific, fixed scope. The BBBOnline trustmark denotes compliance for advertising standards. To combat fraudulent use, the trustmark is clickable and links to a registry where the company's trustmark issuance is documented. VeriSign offers a similar clickable trustmark, but issuance denotes the recipient is authentic and that VeriSign does daily malware scanning. TRUSTe also provides clickable trustmarks, but for a variety of different purposes, such as the U.S. Health Insurance Portability and Accountability Act (HIPAA), the EU General Data Protection Regulation (GDPR), and the Asia-Pacific Economic Cooperation (APEC) Privacy Framework.

While these efforts are commendable, how do trustmarks relate to federations? Trustmarks are intended to denote confidence bestowed by a trusted third-party, but how can they be used in federations that can have a wide variety of requirements for very different application domains? Besides meeting technical requirements of operation, can they be used to manage legal requirements? These were the goals of the Trustmarks project cite:[Moyer2016] at the Georgia Tech Research Institute (GTRI).

In the GTRI approach,

```
"A trustmark is a machine-readable, cryptographically signed digital artifact
representing that a specific named entity conforms to a well-scoped set of
requirements, as attested by a trusted, third-party assessor."
```

Such trustmarks enable the "componentization of trust characteristics and requirements at an arbitrary level of granularity". That is to say, with this approach, trustmarks can be defined for any aspect of a federation where compliance needs to be maintained. Since trustmarks are machine-readable, this enables *automated reasoning* about their requirements and conformance criteria. This means that trustmarks can be created, managed and validated in a *trustmark framework*. This includes *trustmark policies* that are managed in a *trustmark legal framework*. GTRI even envisioned a *trustmark marketplace* where trustmarks could be cataloged with metadata, and reused in a wide array of application domains.

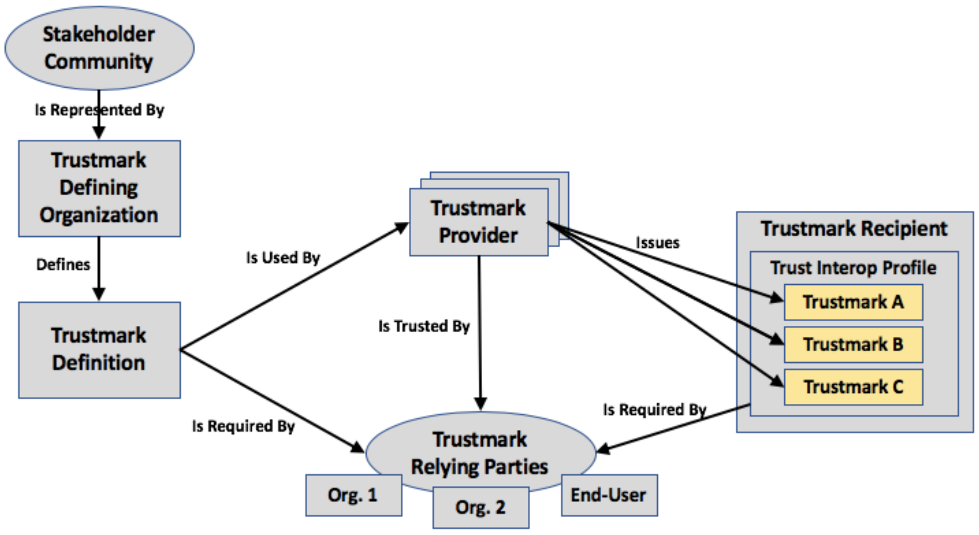


Figure 9. The Basic Trustmark Framework.

Figure 9 illustrates the GTRI Trustmark Framework. (This is Figure 1 from cite:[Moyer2016] redrawn.) A *Trustmark Defining Organization* that represents a *Stakeholder Community* defines various *Trustmark Definitions*. A *Trustmark Provider* can issue various trustmarks based on those definitions to *Trustmark Recipients* that satisfy the trustmark requirements. A recipient can assemble these issued trustmarks into a *Trust Interop Profile*. When a recipient wishes to interact with a *Trustmark Relying Party*, such as various end-users or organizations, the trustmark profile must be presented. The Relying Party will verify that the trustmarks are signed by a Trustmark Provider that is trusted.

This arrangement of trusted Trustmark Providers that issue cryptographically signed trustmarks that are relied upon by end-users and organizations is clearly patterned after Public Key Infrastructure (PKI). Trustmark Providers are analogous to *Certificate Authorities*, while trustmarks are analogous to *X.509 certificates*. Relying Parties can validate these trustmarks/certificates with a trusted Trustmark Provider, in the same way that X.509 certificates can be validated with Certificate Authority.

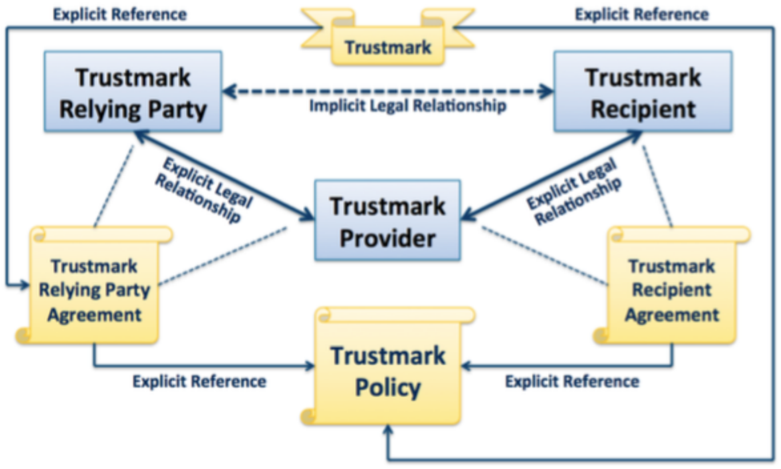


Figure 10. The Trustmark Legal Framework. (Used by permission.)

Trustmarks can also be used to denote *legal agreements*. However, since a legal agreement is between two or more parties, there is an added twist, as illustrated in Figure 10. (This is Figure 2 from cite:[Moyer2016].) A legal contract is denoted as a *Trustmark Policy*. A *Trustmark Recipient Agreement (TRA)* includes a Trustmark Policy by reference. When both a *Trustmark Provider* and *Trustmark Recipient* "sign" a TRA, then a trustmark is issued to the Recipient. This represents an explicit legal agreement between the Provider and Recipient. A similar arrangement occurs with a *Trustmark Relying Party*. A *Trustmark Relying Party Agreement* represents a legal agreement, i.e., the explicitly referenced Trustmark Policy. However, in the GTRI framework, this is a "click-through" agreement that does not need to be "signed" since the TRP is relying on the trustmark issued by the Trustmark Provider.

Extending this kind of trust framework to federations is actually quite straightforward. Moyer, et al., make the point that bilateral trust agreements can be extended into *multilateral trust agreements*. Moyer, et al., contend that a problem with multilateral trust agreements is the assumption that trust requirements and policies must be homogeneous across all participants. They make the point that a framework can still specify a "core" or "minimal" Trust Interoperability Profile. Participants can then add trust requirements to build a customized trust profile.

While the flexibility to customize can be important, having a "core" profile that everybody agrees on is equally important. This is especially true in federations. While it may be possible for a federation to grow "organically" in an *ad hoc* manner, one of the great strengths of a federation is for the participants to agree on the purpose and need of a federation to achieve common goals. This includes the requirements and expectations of membership, what kinds of resources are to be shared, and the policies governing their use.

As discussed in System Rules, this is essentially the *definition* of a federation. Such a definition could be cast as a set of trustmarks, including any legal requirements for membership. This set could be considered a trustmark profile that defines the minimal, core necessities of a federation. The IEEE P2302 WG has only begun to investigate such definitional elements. These could include:

- Federation Name
- Federation Mission Statement
- Description of expected data and service types
    - Description of input/output arguments and semantics

    - Any classification into categories
- Description of member types
    - Description of roles and responsibilities of each type
    - Description of data and services types they are allowed to access, how, and for what purposes
- Set of role/attribute names
    - Must support the definition of all necessary Discovery and Access policies
- Definition of:
    - How the Fed_Admin role will be fulfilled,
    - How many Fed_Admins there will be, and
    - How they will be coordinated
- Definition of:
    - How new members are on-boarded,
    - How roles/attrs will be granted, suspended or revoked,
    - How membership will be suspended or revoked, and
    - If #Fed_Admins > 1, any necessary or desired coordination process must be defined for the above administrative functions

The exact elements and how they are formally expressed is an open question. This represents a large field for future work and standardization.

The notion of a federation definition can be used in a *Federation Trust Framework*, as illustrated in Figure 11 for a hypothetical *Federation Foo*. As noted above, a federation definition constitutes a trustmark profile. Federation members must satisfy all of the trustmark requirements, including legal agreements, if any, to be issued a trustmark profile by a *Federation Trustmark Profile Provider*. Significantly here, all federation members are both *recipients* and *relying parties* of these trustmark profiles. Besides passing the Trustmark Provider's assessment for trustmark compliance, each member must be able to verify at any time that other members possess the same trustmark profile.

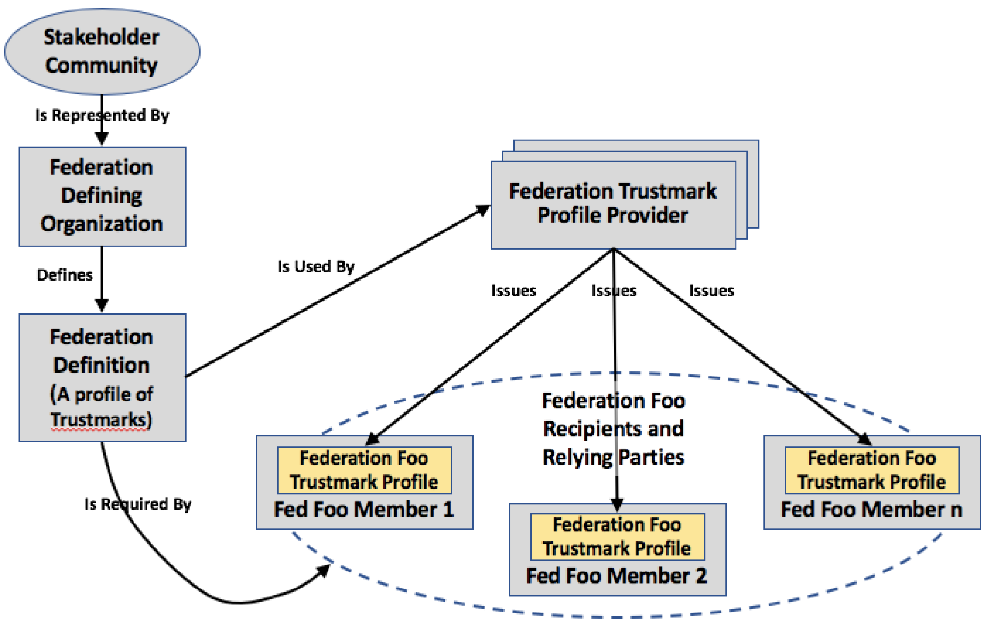

Stakeholder Community
Is Represented By
Federation Defining Organization
Defines
Federation Definition (A profile of Trustmarks)
Is Used By
Federation Trustmark Profile Provider
Issues
Issues
Issues
Is Required By
Federation Foo Recipients and Relying Parties
Federation Foo Trustmark Profile
Fed Foo Member 1
Federation Foo Trustmark Profile
Fed Foo Member 2
Federation Foo Trustmark Profile
Fed Foo Member n

Figure 11. A Federation Trustmark Framework.

Moyer, et al., make a point of clearly establishing the parallels between PKI and trustmarks. These parallels can be extended to federation trustmarks, as listed in Table 4. In most cases, the parallel terminology is clear. However, as noted, federation members are both trustmark recipients and relying parties. For legal contracts, the recipient agreement and relying party agreements are both member agreements. Trustmark definition and interoperability profile both constitute the federation definition. Finally, we note that the creation of a *Federation Framework Technical Specification* is another outstanding area for future work and standardization.

Table 4. Parallels between PKI, Trustmarks, and Federations

PKI	Trustmarks	Federations
Certificate	Trustmark	Federation Trustmark
Certificate Authority	Trustmark Provider	Federation Trustmark Provider
Subscriber	Trustmark Recipient	Federation Member
Certificate Relying Party	Trustmark Relying Party	
Certificate Policy	Trustmark Policy	Federation Policy
Subscriber Agreement	Trustmark Recipient Agreement	Federation Member Agreements
Certificate Relying Party Agreement	Trustmark Relying Party Agreement	
N/A	Trustmark Defining Organization	Federation Defining Organization
N/A	Trustmark Definition	Federation Definition
List of Trusted Certificate Authorities	Trust Interoperability Profile	

PKI	Trustmarks	Federations
X.509 Specification	Trustmark Framework Technical Spec.	Federation Framework Technical Spec. (TBD)

As Moyer, et al., point out, this analogy can also be made between trustmarks and ABAC. An IdP issues an attribute to a recipient. A relying party that trusts the IdP subsequently uses the attribute to make an access decision concerning the recipient. This commonality among PKI, ABAC, trustmarks (also federation trustmarks) simply serves to underscores the fact that having three, independent parties in any security transaction is a fundamental feature of modern security architectures.

Trustmarks have been under development at GTRI since 2013 and are becoming more mature. GTRI is currently partnering with SAFECOM cite:[SAFECOM] and NCSWIC cite:[NCSWIC] to achieve a nationwide, federated Identity and Access Control (ICAM) capability to enable secure disaster response communications that are cross-disciplinary and cross-jurisdictional. To achieve these real-world goals of SAFECOM and NCSWIC, eleven formal trustmarks have been defined cite:[Trustmark_Def_List]. These cover a number of related topics, such as SAML and OIDC requirements for Relying Parties and Identity Providers. Trustmarks are included for further support and development by CISA cite:[SAFECOM_FY20_Guidance] in FY20.

While the GTRI approach to Trustmarks does provide a promising technical approach for defining the components of federation trust and assessing compliance, note that the wide-scale adoption of trustmarks will depend on many non-technical issues. As noted in this EU report cite:[Kool2012], existing commercial trustmarks "vary on scope, business model, quality and enforcement". Four main problems with trustmark adoption in Europe were identified: "there are too many, they are poorly regulated, they vary in quality, enforcement and credibility and they are national". As trustmarks and trustmark frameworks get developed for wider applications — such as federated environments — concomitant efforts should be pursued to minimize these non-technical challenges that may cause divergence and impede their effective adoption.

9.2.4. The OIDC Federation Specification

An example of how OIDC could be integrated into an API Gateway as a candidate implementation approach for the NIST CFRA was given in Appendix B.2 of the CFRA cite:[CFRA2020]. This model was also used for the CFRA threat analysis presented in A Federation Service Threat Model and Analyses which was, in a large part, a threat analysis of OIDC. This development was motivated by the increased use and market adoption of OIDC. In recognition of the increased importance of federation, the OpenID Foundation has also developed the OIDC Federation Specification, v1 cite:[OIDC_Fed].

The OIDC Federation Specification enables two OIDC Providers (OPs) to establish a trust relationship. This is done by enabling the two OPs to determine that they have a *common Trust Anchor*, which is simply a trusted third-party. This determination is made by resolving a *Trust Chain*.

Since the OIDC Federation Specification is an additional capability that can be added to existing OIDC Providers, it perfectly dovetails with the API Gateway/OIDC example of a Federation Service (Federation Manager) as presented in cite:[CFRA2020]. Figure 12 illustrates how the OIDC

Federation Specification can be used to support NIST CFRA-based federations.

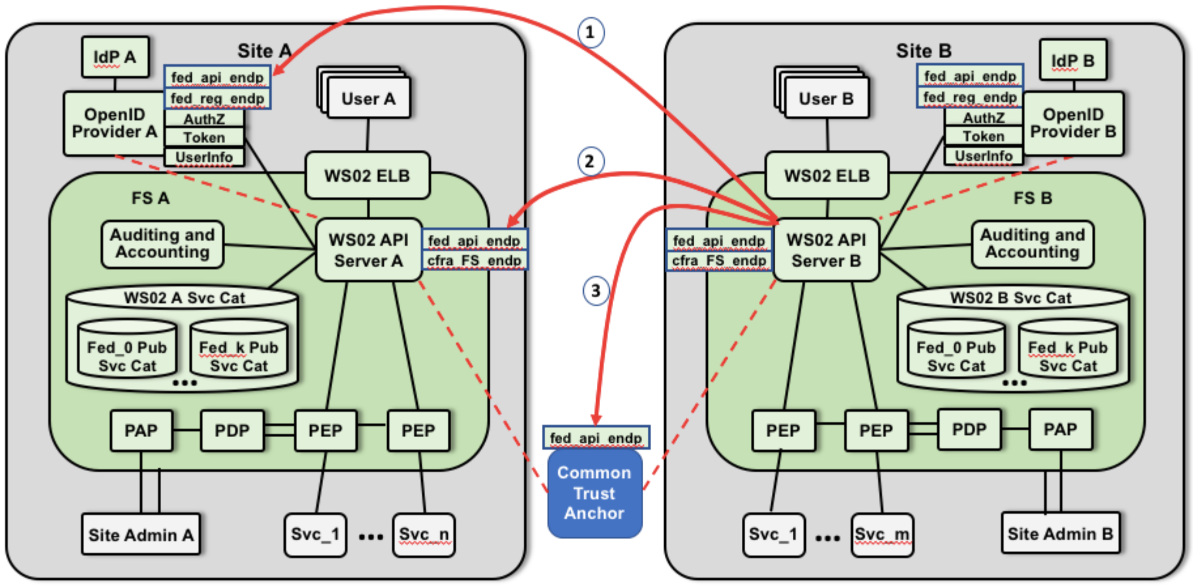


Figure 12. Using the OIDC Federation Specification.

Here we see that the OIDC Fed Spec adds two new endpoints to the OIDC protocol: The *federation API endpoint* and the *federation registration endpoint*. In this example, Site B wishes to federate with Site A. OP A's federation API endpoint is initially discovered using the site's well-known OIDC URI. Site B uses the federation API endpoint to fetch an *entity statement* containing metadata about Site A. This statement will contain a list of *intermediate* entities. At this point, Site A and Site B may *negotiate* trust. This simply means that Site A (the OP) may tell Site B (the RP) what operations, scopes and claims the Site B would be allowed use if a specific Trust Anchor was used.

Site B will make a sequence of such calls to intermediate federation API endpoints until a *Common Trust Anchor* is found, all possible intermediates have been examined, or a maximum path length of intermediates has been hit. This is illustrated by the sequence of (1), (2), and (3) arrows. At each step of this process, JSON Web Key Sets (JWKSs) are exchanged whereby the entities in the trust chain can be validated. If no Common Trust Anchor has been found, trust has not been established. If more than one Common Trust Anchor has been found, then Site B must choose one to use.

Once this is done, the Trust Anchor can be used to exchange JWKSs allowing each side to encrypt and sign messages. Upon receipt of a message, the recipient has the proper keys to verify the identity of the sender and decrypt the message. One of the first such interactions is for Site B to *register* with the Site A OP as a *client* using the *federation registration endpoint*. This will enable Site B to verify the credentials of a Site A federation member when that member is requesting access to a Site B resource.

As extensively described in the NIST CFRA, Federation Managers (Federation Services) must exchange information through their *CFRA FM endpoints*. This is to ensure a consistent presentation and operation of hosted federations to the members regardless of which Federation Managers the member is using to access the federation. The OIDC community has recognized a very similar need for OPs to exchange information for a variety of reasons. Developing a standard capability for this is the charter of the *OIDC Shared Signals and Events (SSE) Working Group* cite:[OpenID_SSE_WG]. This kind of signaling capability could be directly adopted in Federation Service implementations.

The SSE WG is developing an *Event Stream Management API* cite:[Scurtescu2018] whereby an *Event*

Receiver can control how an *Event Transmitter* generates events. This is a straight-forward API enabling a Receiver to:

- Configure how a stream of events are delivered from the Transmitter,
- Query the status of a stream,
- Add and remove *subjects* from an event stream, and
- Request *verification events* allowing the Receiver to determine that the stream is configured properly, including encryption and signature verification.

This basic API was developed to support several specific requirements:

- Enable information to be updated in federated environments to ensure that access decisions are not being based on stale information,
- Preventing malicious actors from gaining access to unauthorized resources, and
- Mitigate the impact of account compromise through Risk & Incident Sharing and Collaboration (RISC).

These are all capabilities that could be used in CFRA-based implementations. Moreover, as mentioned above, the Federation Services must exchange information on all the hosted federations to ensure their proper and consistent operation. Hence, such a signaling capability could be used to:

- Update service catalogs,
- Update discovery and access policies, and
- Update the roles and attributes defined with a federation.

Other uses could be possible, depending on how the FSs are implemented and how the federation semantics must be managed. While this Event Stream Management API was developed to support identity operations in an OIDC Provider, it clearly had broader applicability. An outstanding issue then is to determine how such a signaling capability could be properly integrated to support both the OIDC Provider and the Federation Service requirements as a unified whole.

9.2.5. Hardware Roots of Trust

While the Common Trust Anchors in the OIDC Federation Specification are implemented as software services, the same concept of a *root of trust* can be implemented in *hardware*. As defined by the Trusted Computing Group cite:[TPM2020], the *Trusted Platform Module (TPM)* has an *Endorsement Key (EK)* which is burned into the hardware by the manufacturer. This Endorsement Key consists of a public and private key pair. There can also be an *EK certificate* that is issued for the EK public key. Either the EK public key or the EK certificate is used to establish trust with a *Certificate Authority*.

TPMs have multiple functions, such as performing cryptographic operations, computing hash functions, key management, but also *integrity measurement* and *attestation*. At load-time, the TPM can use a hash function to uniquely identify a binary executable, and even its input data. That is to say, the TPM can establish *code identity*.

This capability can be used to build *Attestation Architectures*. In an Attestation Architecture, a remote *relying party* wishes to verify that a *target system* is intact, i.e., it has not been modified from a known, trusted state, and hence, is trustworthy. Upon an attestation request, the target system TPM will generate a certificate identifying the state of the software that is currently running. If this is not identical to the load-time certificate, a change has occurred.

Such attestation architectures can be used in virtualized, cloud environments. A cloud will have multiple virtual machines running on one physical machine. While there is only one hardware TPM on the physical machine, each virtual machine can be supplied with a *virtual TPM (vTPM)* cite:[Lauer2016]. As far as the VM is concerned, the vTPM behaves exactly like a hardware TPM. Hence, it is possible to make an attestation request for just the VM. However, it is also possible to do a *deep attestation* that assesses the state of the VM, its hypervisor, and the host OS — all the way down to the hardware TPM.

Attestation can also be done for containers. The *TapCon* system cite:[TapCon2017] is an *attesting container manager* built on Docker. This solution can provide *source-based attestations* of the containers it launches. *TapCon* can also provide attestations of container *provenance* back to a trusted container repository.

TapCon supports *layered attestation* in cloud environments from containerized services, to virtual machines, to hypervisors, etc. Technically speaking, *TapCon* can use a TPM hardware root of trust, but can also root trust in a software-based cloud provider or a code repository.

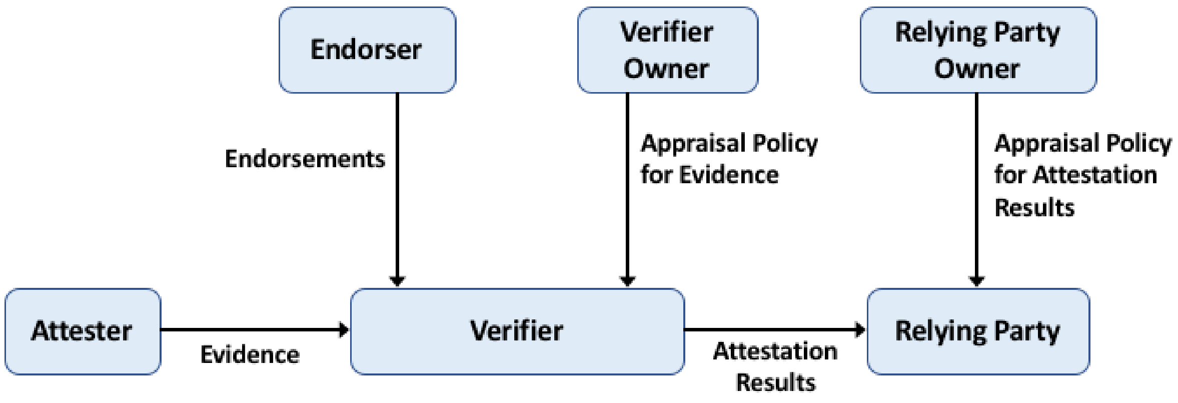


Figure 13. The RATS Conceptual Data Flow.

The fact that such attestation architectures are based on a relying party that is remote from the target system has prompted the development of a *Remote Attestation Procedures Architecture (RATS)* cite:[RATS2020].

The RATS conceptual architecture is shown in Figure 13. Here, an *Attester* provides *evidence* to a *Verifier*. The *Verifier* can also take *endorsements* from an *Endorser*. In practice, such endorsements could simply be the manufacturer vouching for the device’s signing capability. The *Verifier Owner* defines the *appraisal policy* for the evidence, on which the *Verifier* bases the *attestation results*. Such appraisal policies might include recognizing the manufacturer’s endorsement of the device and verifying the format of the evidence received, but ultimately results is a decision of compliance or non-compliance. Finally, the *Relying Party Owner* defines the *appraisal policy* whereby the *Relying Party* can evaluate the *attestation results*. While this appraisal policy might be simply based on the code’s compliance status, it might also involve more detailed information about the Attester and the attestation environment, such as who the manufacturer is.

This high-level architecture is further developed for different *target environments* and *attestation environments*, along with different *topological models*. In the *Passport Model*, the *Verifier* provides the attestation results to the *Attester*, which the Attester can present to the *Relying Party*, i.e., as a "passport". In the *Background-Check Model*, the *Attester* provides all evidence to the *Relying Party*, who then passes the evidence to the *Verifier*. The *Verifier* does a "background-check" and returns the attestation results to the *Relying Party*. The format for all message exchanged in an attestation can include JWTs, CWTs, X.509 certificates. The root of trust for these tokens/certificates can be software or hardware-based, i.e., TPM.

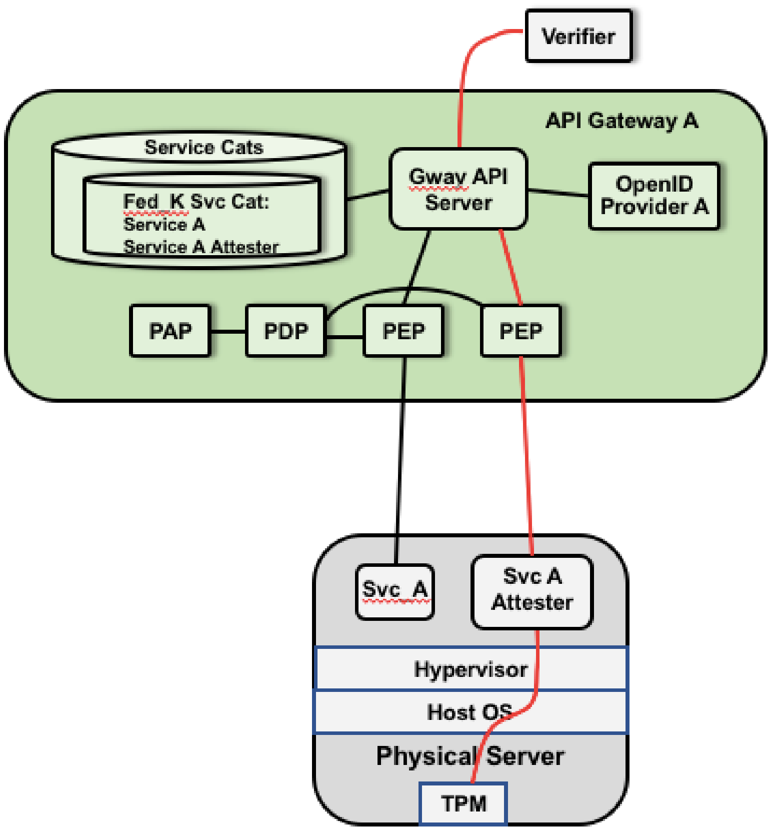


Figure 14. A Federation Attestation Flow.

Hardware Roots of Trust enable a specific version of software to be given a unique identifier, i.e., an *identity*. Attestation is the process of verifying that identity. This is not unlike that of establishing a user's identity in order to make an access decision.

Figure 14 illustrates how attestation evidence might be collected in a federated environment. Here, a *Service A* is running on some physical server with a TPM. Service A could be running in a VM (as illustrated here with a Host OS and Hypervisor), but could also be in a container. Service A has been registered in the Service Catalog of Federation K by Service A's owner. Service A's owner also has a *Service A Attester* running on the same physical machine that is also registered in the Service Catalog.

In this example, the *Verifier* is a member of Federation K that has the authorization to make *attestation requests* of Federation K services. That is to say, the Verifier can discover all attestation services in the Federation K Service Catalog and invoke each one to verify the service's code identity. Such an attestation is illustrated by the red line in Figure 14.

Note that the Service A Attester could attest for any other services hosted in that VM. Also, a Verifier could have the authorization to request attestations at any site in the federation. Furthermore, there could be multiple Federation K members that have the authorization to do attestations.

Once the Verifier has collected the evidence, there are other entities and steps involved. In the RATS model, the Verifier might receive Endorsements, typically from the manufacturers. The Evidence Appraisal Policy might be defined for the entire federation or might be Attester-specific. This is another governance issue that could be decided on a federation-by-federation basis. The *Relying Parties* could possibly be any other federation member that wishes to verify the integrity of a service before using it, including a Fed_Admin. The Relying Party Owners could certainly define their own Appraisal Policies for the attestation results.

Trust is the final issue that must be addressed. In the typical attestation architecture, the Verifier trusts the attestation evidence because it is signed by a TPM whose key is certified by a trusted PKI Certificate Authority. In this example, when the Service A Attester is registered in the Service Catalog, the OIDC Provider could also issue an ID Token for the Attester and the TPM the Attester uses. When an attestation is done, the Verifier validates the TPM ID token with the issuing OIDC.

This conceptual architecture indicates that attestation could be done in federated environments. The next question is how federation could leverage the significant adoption that Trusted Computing has had in the marketplace. Major chip manufacturers, such as Intel and AMD, include TPM hardware. Microsoft Trusted Computing includes TPM support in Windows 10, Windows Server 2016 and Windows Server 2019. Microsoft's *IOT Hub Device Provisioning Service* supports TPM-based attestation. Google Compute Engine can provide *Shielded VMs* that have an attached vTPM. All of this implies that if general, standards-based federation tools were adopted, then existing hardware-based attestation mechanisms could be brought to bear within federations.

9.2.6. Blockchain in Federated Environments

The definition of non-repudiation essentially says two parties to a transaction cannot later deny that the transaction took place. In IT systems, such transactions are always the exchange of information. Federations are clearly dependent on the exchange of information that is understood to be shared and trusted.

Blockchain is a technology whereby such trusted information can be established with non-repudiation. Rather than rely on securing the channels and sessions between partners, the Blockchain approach creates an unforgeable record of the data and the data exchange transactions among users. A Blockchain cite:[Yaga2018] is a *replicated ledger* that uses cryptographic techniques and *consensus algorithms* to build trustworthy systems in an otherwise trustless world.

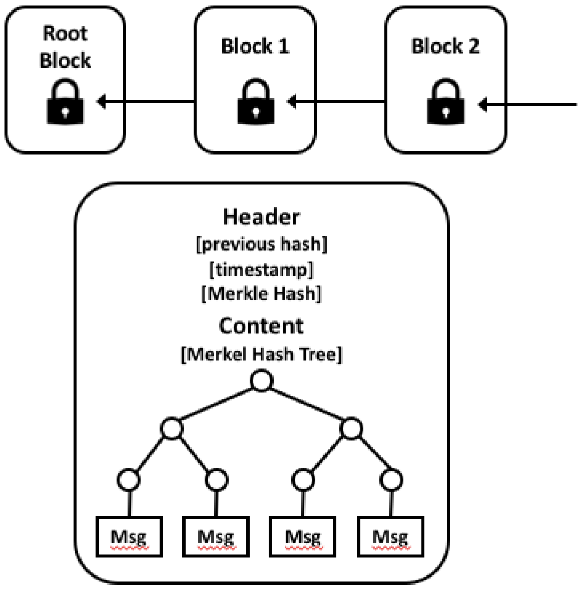


Figure 15. The Blockchain Replicated Ledger.

Figure 15 illustrates the Blockchain concept. A Blockchain is simply a chain of blocks, or data structures. Each block contains a cryptographic link to the previous block. The beginning block is the *root block*. Each block consists of a *header* and *content*. The header contains a link to the previous block, a time stamp, and a *Merkle hash value*. This Merkle hash value is the value of the *Merkle hash tree*. This is a tree of one or more application messages. The Merkle hash value is cryptographically linked to the entire contents of the tree. The upshot is that by cryptographically replicating the ledger among participants, fraudulent information and transactions can be identified and rejected.

The Blockchain "magic" happens in how new blocks are added to the chains; i.e., how a *consensus algorithm* is used to establish agreement among participants for adding a new block. *Proof of Work* is the most common consensus mechanism used in Blockchain implementations. This proof of work in early Blockchain implementations required Blockchain *miners* to experimentally determine which *cryptographic nonce* makes the hash of its current header fit the current target. *Proof of Stake* gives advantage to those miners that have a larger stake in the Blockchain ecosystem. The Practical Byzantine Fault Tolerance (PBFT) algorithm provides a lower latency mechanism where arriving messages are signed, and if enough identical messages are received, then consensus is achieved. As such, these consensus algorithms as well as others have very different performance characteristics.

So how can a Blockchain be used in federations, and specifically CFRA-based federations? The answer to this question depends on the federation deployment model used. The deployment model that is expected to be commonly used entails P2P Federation Services where the Service Catalog, service discovery policies, and authorization attributes are replicated among the FSs. Service Owners have the authority to make a resource available in a federation. This includes all metadata concerning the resource, e.g., the URL, structure of the request/response data, and resource discovery policies. Authorization attributes can be defined by the Fed Admin(s). All of this information must be replicated and trusted among the Federation Services. This could be accomplished by posting any changes to the Service Catalog or attributes to a Blockchain. The Federation Services could even run their own *private* Blockchains for this purpose. Federation Services could even offer private Blockchain services to the federation instances they are hosting, to be used by the federation members to establish trusted information within any given federation. Hence, the use of Blockchains in federation is a very interesting possibility, but does entail the resolution of many issues, including consensus algorithms, performance, and block data size.

9.3. Zero Trust

A very different approach to trust is taken in the *Zero Trust Architecture (ZTA)* concept cite:[Rose2019]. A large segment of current enterprise security models is based on *network* security. Entire enterprise networks and network segments are protected by firewalls. Point-to-point communication channels are secured by using tools such as TLS. This approach to trust and security is very *coarse-grained* and *network-centric*. As such, the protection is at a very low level in the system stack, i.e., the network. This makes it more difficult to use this approach to protect specific system components, since it relies on the network configuration. As such, this network-centric approach is not very agile.

9.3.1. The Zero Trust Architectural Concepts

The ZTA approach is to enable a "protection boundary" to be placed around any set of enterprise resources, large or small. Any such set of resources is called a *zone of implicit trust*. Everything between such implicit trust zones is considered to be *untrusted*, including the network, regardless of whether it is the local, enterprise network or the Internet. The ZTA design philosophy is to make these trust zones as small as possible, i.e., to make protection as *fine-grained* as possible.

These concepts are embodied in a set of ZTA design tenets in cite:[Rose2019]. Each of these tenets has distinct implications for the implementation, deployment and use of ZTAs. Hence, we follow each of these tenets with an *Interpretation* and discussion of *Implications (I&I)*:

```
ZTA Tenet 1: All data sources and computing services are considered resources.
```

I&I 1: This enables system designers to identify any number of system components or groups of components and protect them individually. How big or small these groups are depends on the system designer’s decision as to what needs protection and the overall risk tolerance.

```
ZTA Tenet 2: All communication is secured regardless of network location.
```

I&I 2: All data must be encrypted when it exits a trust zone. Since all communication outside a trust zone is assumed to be untrusted, the assumption is that anybody can see the data. However, only legitimate recipients can decrypt the data and validate the identity of the sender. From an implementation perspective, this implies that any recipient must have the proper keys to decrypt the data and validate the signature. This implies that there must be a method for *secure key distribution.*

```
ZTA Tenet 3: Access to individual enterprise resources is granted on a per-session
basis.
```

I&I 3: This implies that access decisions need to be made. Making access decisions also implies that there are access policies for each resource, and a mechanism for evaluating these policies. Rose, et al., use the common term *Policy Enforcement Point (PEP)* to denote this mechanism.

The credentials/tokens and policies must have semantic interoperability. That is to say, the attributes in the credentials/tokens must be understandable and have a well-known meaning when applied to a policy to make an access decision.

Access decisions can be made either by: (a) Submitting a credential or token with each request, or (b) Using the credential or token to authenticate a *session* between the requestor and the resource. If a session is established, there must be some way for the user to terminate the session when it is no longer needed. Sessions must also have a limited period of validity after which they expire, but with the possibility of re-validation. Resources or IdPs must be able to revoke a session at any time.

```
ZTA Tenet 4: Access to resources is determined by dynamic policy—including the
observable state of client identity, application, and the requesting asset, and may
include other behavioral attributes.
```

I&I 4: This implies that policies may also include contextual or environmental information. For example, access may be declined if a user on a mobile device using a coffee shop Wi-Fi at 10 PM. This implies that there must be a way to determine the user's device and physical location when requests are being made.

```
ZTA Tenet 5: The enterprise ensures that all owned and associated devices are in the
most secure state possible and monitors assets to ensure that they remain in the most
secure state possible.
```

I&I 5: Ideally this implies that there is a formal definition of what the "secure state" is and also an audit mechanism that can automatically evaluate compliance to that secure system state. If there is any deviation from the desired secure state, an alarm can be raised and actions taken to rectify the deviation. If the deviation is associated with an actual compromise, then actions should be taken to mitigate and contain any negative impact.

```
ZTA Tenet 6: All resource authentication and authorization are dynamic and strictly
enforced before access is allowed.
```

I&I 6: If not authenticating on each call, a user must be periodically re-authenticated. Any authentication session must have a finite period of validity, but with the possibility of being refreshed. There must also be a way of *revoking* authentication and authorization at any time.

```
ZTA Tenet 7: The enterprise collects as much information as possible about the current
state of network infrastructure and communications and uses it to improve its security
posture.
```

I&I 7: One could argue that this tenet is not specific to ZTAs, and could be applied in all architectural philosophies. However, since the ZTA tenets imply the use of fine-grained PEPs, this gives ZTAs a place to monitor and harvest as much network and communication information as possible. This information could be used by a range of cybersecurity tools to improve the security posture. Such tools could include Artificial Intelligence and Deep Learning-based approaches.

While all of the tenets are important in the overall ZTA concept, Tenet 3 is the one that clearly points to the need for some type of Policy Enforcement Point. This is clearly called out by Rose, et al., cite:[Rose2019] and Kerman, et al., cite:[Kerman2020] as illustrated in Figure 16.

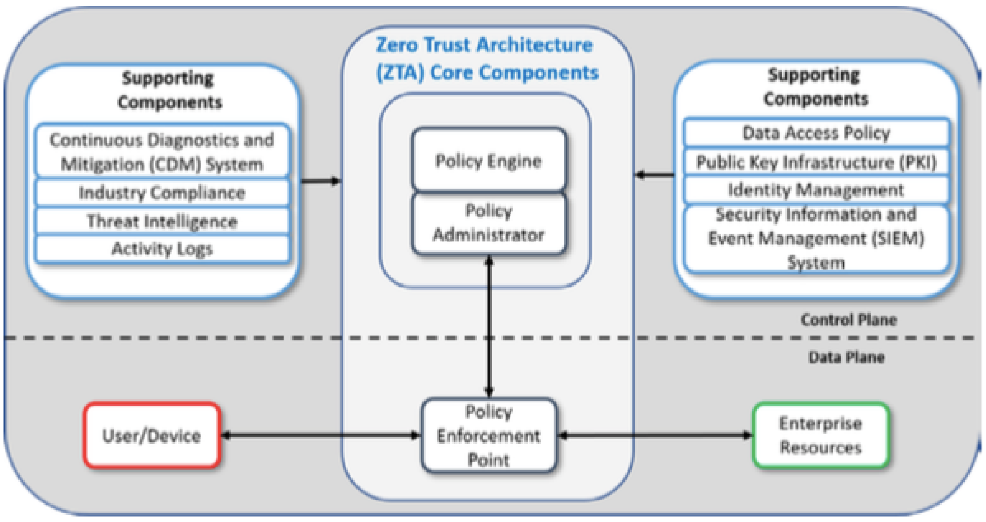


Figure 16. The ZTA High-Level Architecture.

This high-level architecture is partitioned into a *Control Plane* and a *Data Plane*. The *Policy Enforcement Point (PEP)* resides in the Data plane and is used to control access from the external (untrusted) world to a protected *Enterprise Resource* in a Zone of Implicit Trust. The PEP is controlled by a *Policy Administrator* that uses a *Rule Engine* to make access decisions. The rule set evaluated by the Rule Engine is informed by a host of supporting components, such as *Data Access Policy* and *Threat Intelligence*.

This is a straight-forward repurposing of the terms first used in the OASIS eXtensible Access Control Markup Language (XACML) standard cite:[XACML]. XACML defines far more than just a protocol for exchanging access control assertions. It defines a general security architecture based on *Policy Enforcement Points (PEPs), Policy Decision Points (PDPs)* and *Policy Administration Points (PAPs)*. In Figure 16, the Policy Administrator is serving the function of the PDP.

While the ZTA approach is sound, this presentation glosses over critical interactions needed to make the architecture work. First, while it is implicitly there, Figure 16 does not explicitly identify the entity that manages the Access Policies being enforced by the Rule Engine and its Rule Set. Second, while both PKI and Identity Management are called out, how identity credentials containing authorization attributes are granted and revoked for Users and Devices is a critical function. While this is certainly manageable in a local Identity Silo, how is a ZTA managed over a large, distributed environment with multiple organizations and stakeholders? Also note that while PKI will certainly be of interest to many stakeholders, other credential systems exist. While the User-managed wallet approach of W3C Verifiable Credentials is very different from the third-party PKI Certificate Authorities, the OIDC approach shares this design element.

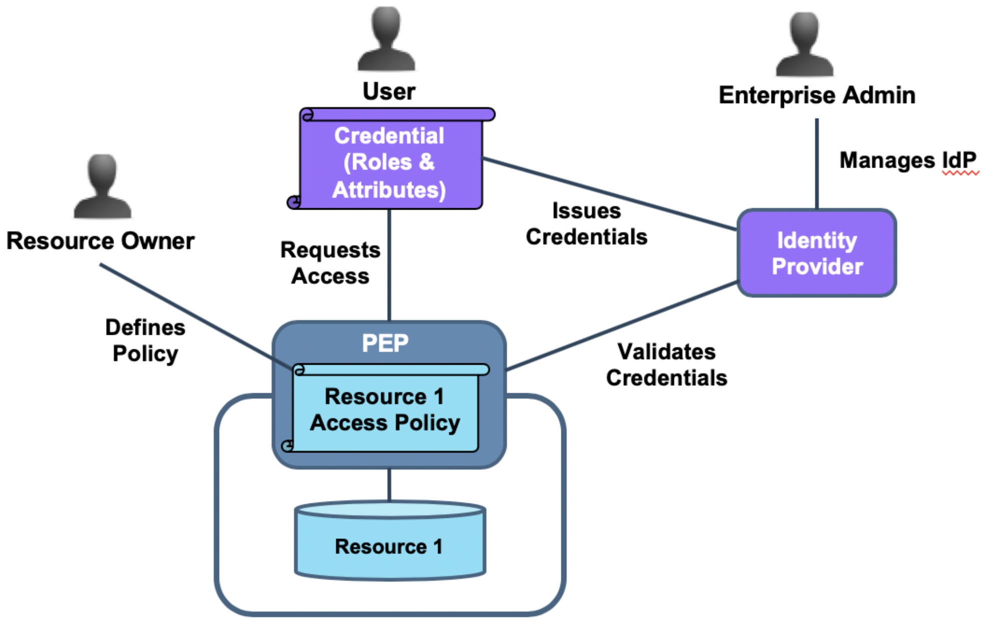


Figure 17. A Basic PEP-based Security Architecture.

Figure 17 illustrates the interactions that must take place to enable a PEP to do its job. It is clear that this PEP-based approach is tantamount to *Software-Defined Boundaries* and has strong similarities to the NIST CFRA model.

9.3.2. The Microservice Mesh ZTA Approach

While Kerman, et al., cite:[Kerman2020] is silent about *Microservice Mesh Architectures*, this is the initial approach that NIST is taking to ZTA implementations. This is well-documented by the presentations at the *NIST Identity Management & Access Control in Multiclouds Workshop and Conference* cite:[NIST_IMAC_2020].

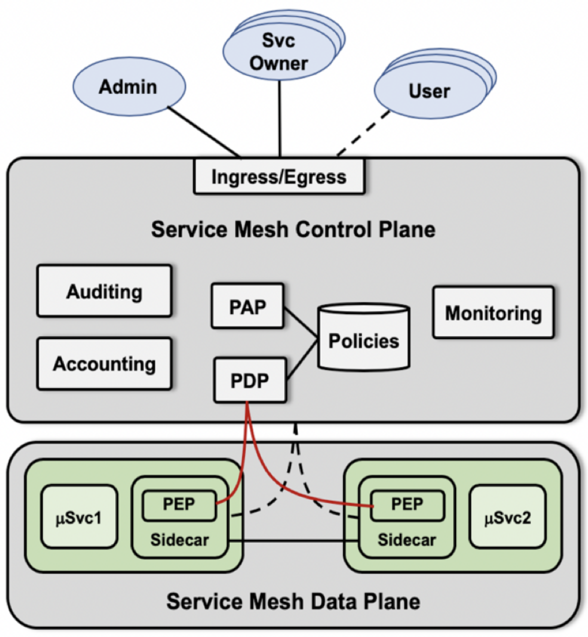


Figure 18. A Microservice Mesh Architecture.

The Microservice Mesh Architecture concept is illustrated in Figure 18 (after Chandramouli and Butcher cite:[Mouli2020]).

This is also partitioned into a Control Plane and Data Plane. The Control Plane manages many microservices that are running in the Data Plane. Every microservice, however, is paired with a *Sidecar*. These Sidecars give the Control Plane a uniform way to manage all microservices. Besides managing all "east-west" communication among microservices, Sidecars can be used to implement PEPs that are controlled from the Control Plane. The Control Plane can also provide monitoring, accounting and auditing services for the entire set of microservices. Finally, the Control Plane manages all ingress and egress of data and requests, i.e., all "north-south" communication, from Admins, Service Owners and common Users.

The specific ZTA implementation approach that NIST is taking is to integrate the *Istio* microservice mesh system with *Next Generation Access Control (NGAC)* previously developed at NIST. Istio cite:[Istio] is an open-source project. Tetrate cite:[Tetrate] is the main Istio contributor that is working with NIST. NGAC was started at NIST as the *Policy Machine* cite:[Ferraiolo2015] in 2003 and evolved into a set of INCITS standards under the NGAC name.

The Istio tool suite includes *Envoy*, a Sidecar implementation. Istiod comprises the Istio control plane, which includes:

- *Galley* for configuration management,
- *Pilot* that abstracts platform-specific service discovery mechanisms,
- *SW* that provides microservice telemetry, and
- *Citadel*, a certificate authority that provides TLS certificates to Envoys, whereby all data is encrypted in-transit.

As an access control mechanism, NGAC has fundamental differences from other established, rule-based access control systems, such as XACML. Based on the comparisons made by Ferraiolo, et al. cite:[Ferraiolo2016], the main NGAC differences are:

- The use of *Relational Graphs* rather than logical rule sets,
- Administrative requests are handled in-band with access requests,
- Administrative governance is handled through assignments and graph associations, the same way that other access policies are handled,
- Obligations are only handled *after* a *successful* access request,
- Environmental/contextual attributes are not handled,
- NGAC can dynamically change access policies to prevent data leakage,
- NGAC's relational graph approach makes administrative review a linear process, and
- NGAC can support discovery of resources in the association graphs, but this is not a general resource discovery mechanism.

Implementing a ZTA at the microservice level is a very fine-grained approach. Doing so will raise performance concerns. In recognition of this, Istio can utilize very simple, lightweight checks that do not use multiple rules with multiple predicates. While this will have a smaller performance impact, it limits the expressiveness of policies that can be defined and enforced.

9.3.3. Zero Trust Federations

While the microservice mesh approach does enable very fine-grained protection of zones of implicit trust, this is not the only possible approach. The PEP function illustrated in Figure 16 and Figure 17 are directly supported in *API Gateways*. The use of PEPs, and specifically the XACML model, in a Federation Manager is explicitly described in the CFRA, Appendix B.2 cite:[CFRA2020]. The integration of the OIDC Federation Specification into a Gateway-based Federation Service illustrated in Figure 12 was derived from this example. While an "ordinary" implementation of a ZTA may be in an Identity Silo, the NIST CFRA model specifically manages the semantic interoperability of credentials, resources, and policies across distributed, multi-organizational environments, i.e., federations.

So what would a *Zero Trust Federation* actually look like? We review how the NIST CFRA relates to and can address the ZTA design tenets:

```
ZTA Tenet 1: All data sources and computing services are considered resources.
```

ZTF Approach: This is already an integral part of the CFRA, albeit at the service endpoint level for arbitrary, application-level services that can be handled in an API Gateway.

```
ZTA Tenet 2: All communication is secured regardless of network location.
```

ZTF Approach: The ZTA approach is to encrypt data when in-transit between any two trust zones. ZTFs can take the same approach. However, this approach must be done between Users, Services, and the Federation Services. This could be done using mTLS or adding an endpoint for retrieving JWKSs.

```
ZTA Tenet 3: Access to individual enterprise resources is granted on a per-session
basis.
```

ZTF Approach: Access control on a per-request or per-session basis is already an integral part of the NIST CFRA.

```
ZTA Tenet 4: Access to resources is determined by dynamic policy -- including the
observable state of client identity, application, and the requesting asset, and may
include other behavioral attributes.
```

ZTF Approach: This is already an integral part of the NIST CFRA.

```
ZTA Tenet 5: The enterprise ensures that all owned and associated devices are in the
most secure state possible and monitors assets to ensure that they remain in the most
secure state possible.
```

ZTF Approach: While not explicitly addressed in the NIST CFRA, the notion of monitoring compliance to a desired system state or set of operating principles is, in fact, addressed by the Trustmark concept. Clearly, though, practical tooling to automatically monitor for compliance will require extensive development. Please note, however, that API Gateways already provide local logging and monitoring services. The challenge for Gateway-based Federation Services is how to do this across a distributed federation.

```
ZTA Tenet 6: All resource authentication and authorization are dynamic and strictly
enforced before access is allowed.
```

ZTF Approach: This is already an integral part of the NIST CFRA.

```
ZTA Tenet 7: The enterprise collects as much information as possible about the current
state of network infrastructure and communications and uses it to improve its security
posture.
```

ZTF Approach: This tenet is strongly related to Tenet 5 in that it requires monitoring, interpretation, and improvement. The logging and monitoring done by API Gateways could be used for this purpose.

From this quick review, it is evident that gateway-based federations are quite compatible with the ZTA approach. Tenets 1, 3, 4 and 6 are already central to the NIST CFRA. To show how the NIST CFRA can address the other Tenets, we can further develop the concepts of (a) the encryption of all data in-transit to address Tenet 2, and (b) an effective, distributed monitoring and analysis capability to address Tenets 5 and 7.

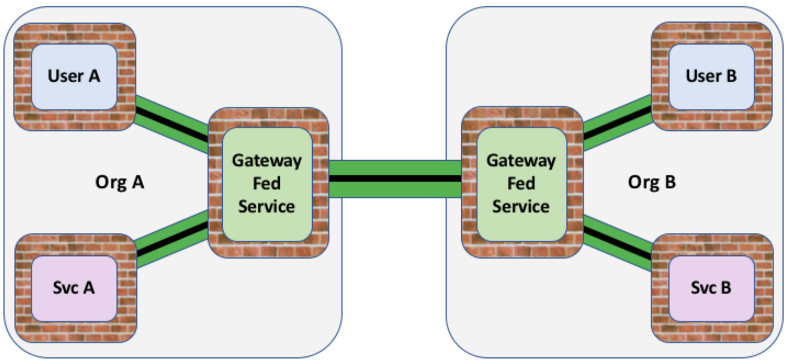


Figure 19. A High-Level Zero Trust Federation Architecture.

Without loss of generality, we can say that the implicit zones of trust in a federation are (a) the Users, (b) the Services, and (c) the Federation Services. This is illustrated in Figure 19 where each organization operates its own Gateway-based Federation Service in a peer-to-peer deployment mode. All of the communication between these entities is encrypted. While the *Gateway-to-Gateway* communication could be encrypted using mTLS, piggy-backing on the secure signaling between OIDC Providers would be even more secure. This is due to the fact that the OIDC Federation Specification requires the identification of a Common Trust Anchor. This Trust Anchor facilitates the exchange of JWKSs whereby messages can be encrypted and signed.

When on-boarding a new User, or perhaps the first User from a new Organization, the User and the Gateway Federation Service must be able to establish each other's identity. Depending on the federation’s security requirements, this could be done simply using mTLS. However, ZTFs requiring stronger identity proofing could require the use of an independent third-party — much like a Trust Anchor — to establish mutual trust prior to establishing the encrypted communication.

Services are always registered in a Gateway Federation Service by a User that has the Service_Owner authorization. When registering a service, the Service_Owner essentially acts as the trusted third-party introducing the Service to the Gateway Federation Service. Once this introduction is made, the encrypted communication can be established.

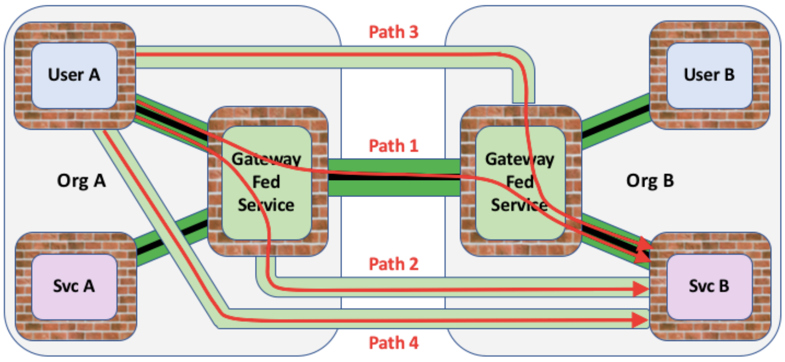


Figure 20. Alternate invocation paths in a ZTF.

While Figure 19 illustrates a basic ZTF, Figure 20 illustrates several, possible *invocation paths* to be considered. If only the dark green connections are available in a Zero Trust configuration, then for any User that invokes a Service, the Gateway Fed Services must *proxy* that invocation. This is *Path 1*. That is to say, User A invokes the Service through their local Gateway Fed Service, which routes the invocation through a peer network of Gateways, which ultimately invokes Service B itself. As per the ZTA tenets, the Gateway Fed Service (presumably closest to the Service) will validate User A’s credentials to make an access decision prior to forwarding the request to Service B.

While this invocation path is certainly possible, it does mean that all request traffic is being proxied over the Gateway Fed Service peering network. First, this could represent many "hops" and layers of software that will introduce significant latency to response times. Furthermore, if the Gateway peering network is not capable of supporting that traffic, then the peering network would become a performance bottleneck. This can be avoided if alternate routes are considered.

Path 2 avoids the Gateway Fed Service peering network by establishing another encrypted connection between User A’s Gateway and the remote Service B. However, this requires that User A’s local Gateway makes the access decision. That is at odds with the common deployment approach of each organization deploying and operating their own Gateway Fed Service, thereby retaining control of the Gateway Fed Service that is making final access decisions to their resources. Path 2 would require that Organization B trust all other organizations running Gateways when making an access decision for a service in Organization B.

This brings us to Path 3. Here a different encrypted connection is established between User A and Service B’s Gateway Fed Service. This avoids the Gateway peering network, yet allows Organization B to retain control of the final access decision process for their services. Path 3 does increase the number of encrypted connections but will significantly reduce response latencies while avoiding potential network bottlenecks.

Note that Path 4 is possible where a direct connection is established between a User and a Service, after being authorized through the Gateways. This would avoid the Gateways altogether, along with their overhead. If this direct connection is handled as a *session* that has a *time-to-live* and can refreshed, terminated or revoked, then such an invocation path technically satisfies ZTA Tenet 3.

With regards to these different path options, while these paths may be logically distinct at Layer 7 in the OSI network model, they may, in fact, be mapped to the same physical network links at Layers 2 and 3. That is to say, all four paths may put the same bandwidth demand on the physical links that are used for routing traffic between Organizations A and B. However the routing is configured, it is always the case that the available bandwidth between organizations should be adequately provisioned to meet demand. Also note that organizational security policy may prohibit the use of some path options, e.g., anything other than Path 1. This ER must identify and acknowledge all such issues, however, resolving them is unfortunately beyond this ER's scope.

9.4. Data-Centric Security

The OGC has been pursuing the concept of *Data-Centric Security (DCS)* for several years which is definitely related to the ZTA concept. In DCS, data is always encrypted, either at rest or in transit, and decryption keys are only issued to legitimate recipients of the data. Hence, DCS could directly support ZTA implementations. Here we examine how DCS could be used in Zero Trust Federations.

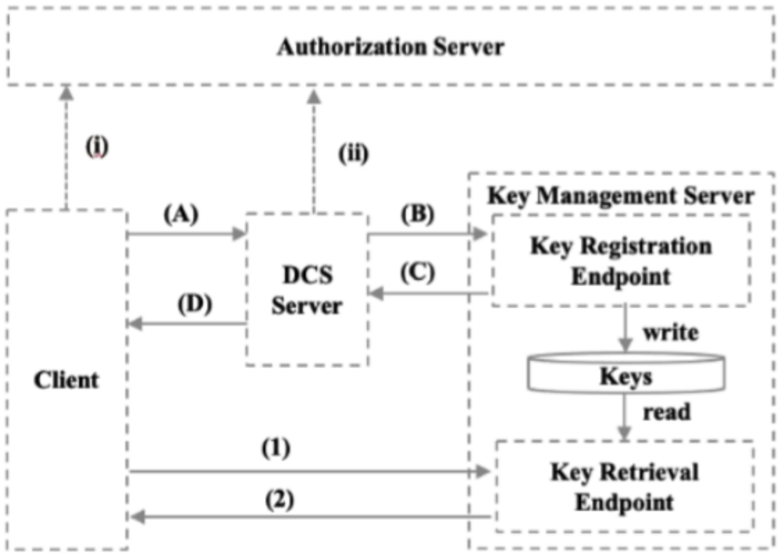


Figure 21. Testbed-16 DCS Components. (Figure 2 from OGC 20-021.)

The current DCS prototype in Testbed-16 is illustrated in Figure 21. This is Figure 2 from OGC 20-021 cite:[DCS-TB16] which includes a full explanation of the interactions shown by arrows (i), (ii), (A), (B), ©, (D), (1) and (2). In brief, when the Client requests data from the DCS Server, the DCS Server registers a decryption key with the Key Management Server (KMS). The DCS Server then returns the encrypted data with a *key_id*. The Client can then retrieve the decryption key from the KMS using the *key_id*.

The DCS and KMS can actually be deployed and used in different configurations and scenarios. The DCS can store data encrypted and register the decryption key with the KMS. Whenever the DCS provides encrypted data to a Client, the Client can retrieve the decryption key from the KMS. The DCS can also store data unencrypted, and only encrypt the data *on-the-fly* when the data is being returned to the Client. Immediately prior to this, the DCS will register the encryption key with the KMS for later retrieval by the Client. In addition, each DCS does not have to be colocated with its own KMS. A DCS could be remotely located and store data from multiple different sources. Hence, the DCS could serve various, encrypted data sets, and the Clients may have to retrieve the decryption keys from various different KMSs.

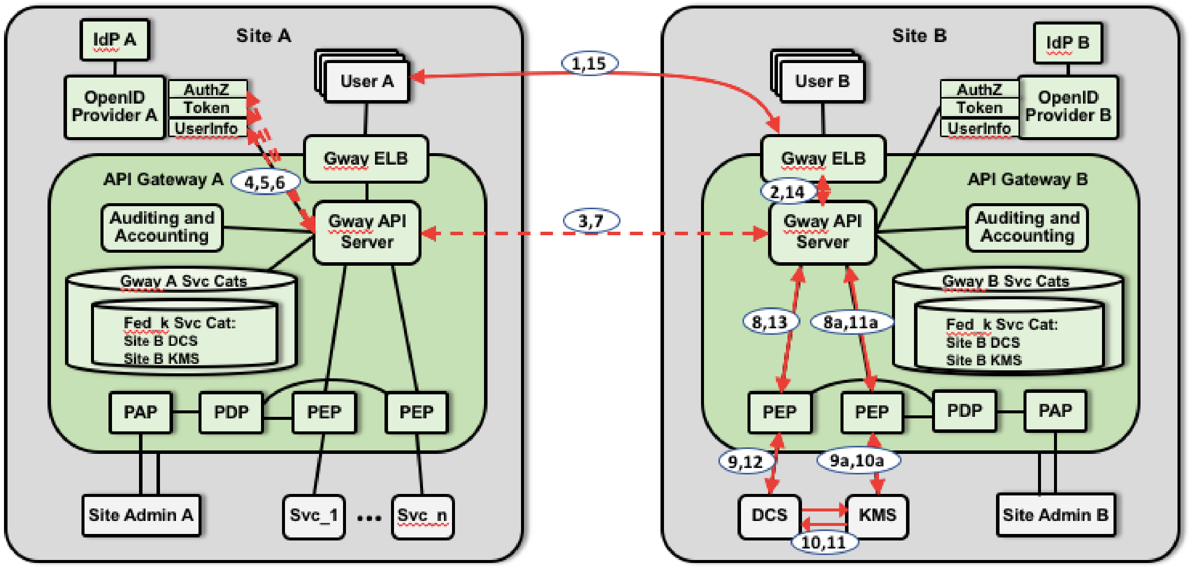


Figure 22. DCS and KMS services registered in a federation.

It is a definite possibility that the DCS and KMS can be managed in a federated environment in the same way as any other services. Figure 22 illustrates such an environment derived from the CFRA, Appendix B.2 cite:[CFRA2020]. (The example in Appendix B.2 show many more steps that are omitted here for brevity.) Federation Services A and B both use OIDC, as does the DCS prototype. Without loss of generality, assume that a DCS and KMS are both deployed at Site B. These service endpoints and associated metadata have been registered in the Federation K Service Catalog at Site B, and this information has been propagated to the Federation K Service Catalog at Site A.

The numbered arrows jump ahead in the example several steps where User A has authenticated to Federation K, and looked-up the DCS and KMS servers in the Site A Service Catalog. User A invokes the DCS with the following sequence of operations:

- Steps 1-2: User A (the Client) requests access to the Site B DCS (corresponding to Arrow (A) in Figure 21).
- Steps 3-7: The Site B FS validates User A's credentials with OP A.
- Steps 8-9: The Site B PEP enforces the access decision.
- Steps 10-11: The DCS registers a decryption key with the KMS (corresponding to Arrows (B) and (C) in Figure 21).
- Steps 12-15: The encrypted data and key_id are returned to User A (corresponding to Arrow (D) in Figure 21).

Having retrieved the encrypted data, User A must now have their authorizations checked to retrieve the decryption key in the following, similar sequence:

- Steps 1-2: User A (the Client) requests access to the Site B KMS (corresponding to Arrow (1) in Figure 21).
- Steps 3-7: The Site B FS validates User A's credentials with OP A.

- Steps 8a-9a: The Site B PEP enforces the access decision.
- Steps 10a,11a,14,15: The decryption key is returned (corresponding to Arrow (2) in Figure 21).

This exercise indicates that a Data-Centric Security model could be employed in federated environments. As noted, DCSs could be located at different sites and store data that have different data owners. In a common governance model it is expected that even if the encrypted data is stored on a DCS that is not under direct control of the data owner, the KMS will be. The different usage scenarios that are anticipated for data-centric security imply that different trust relationships must be established. Nonetheless, these should be supportable in federated environments, possibly as part of Zero Trust Federations.

9.5. Trusted Data Format

A common use case for federations will be to manage the secure sharing of data. The Office of the Director of National Intelligence (ODNI) has developed a relevant approach. The Trusted Data Format cite:[TDF] is a mechanism to control how data can be accessed after it has been disseminated.

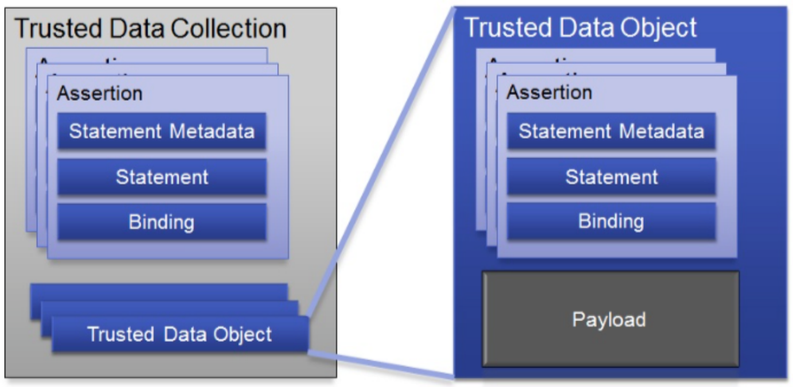


Figure 23. Trusted Data Objects and Collections.

The basic TDF concept is illustrated in Figure 23 (Figure 5 from cite:[TDF]). A Trusted Data Object (TDO) consists of one or more Assertions about a Data Payload. An Assertion consists of a Statement, Statement Metadata, and a cryptographic Binding of the Assertion to the Payload. The Assertions can include information such as the responsible party for the data, access rights, information security marking, time of creation, etc. TDOs can be collected into a Trusted Data Collection (TDC) which has their own sets of Assertions.

TDF is used to provide selective access control for all manner of files, such as PDFs, email attachments, multimedia files, etc. When a user attempts to open a TDO, the TDO “wrapper” contacts an external server to verify if the user is authorized to decrypt, read, and use the data.

While an interesting concept, it appears that TDF has not found much use outside of ODNI. Two immediate concerns are that opening a TDO, the location of the verifying external server must be known and accessible. While the location can be defined in the Assertion, there is the possibility that it may not be accessible from the user’s network. Likewise, the user must have a well-known identity credential and authorization attributes that the verifying server must understand. The verifying server must also know and trust the IdP that issued the credentials. We note that the TDO approach is to protect data after it has been disseminated, whereas the CFRA approach is to protect data before it has been disseminated. It might be possible to integrate both of these approaches in a federated environment. However, specific use cases should be worked out to identify the constraints and benefits of their joint use.

9.6. Attribute-Based Encryption

After trust relationships have been established, federations require secure communication among the participants. While traditional data security approaches involve securing communication channels, another approach is to always encrypt the data at the time of creation and issue decryption keys to only the authorized recipients. This approach is supported by Attribute-Based Encryption (ABE) cite:[Sahai2005]. ABE is a type of public-key encryption where the keys used to encrypt and decrypt a ciphertext are generated from the desired access policy defined over a set of attributes. (In cryptography, ciphertext or cyphertext is the result of encryption performed on plaintext using an algorithm, called a cipher.) ABE can be implemented as Key-Policy ABE (KP-ABE) cite:[Goyal2006], where ciphertexts are simply labeled with a set of attributes. An authorized consumer's private key is associated with an access tree structure that defines which types of ciphertexts the key can decrypt. This access structure consists of nodes that are gates and leaves that are attributes that, in effect, define an access policy.

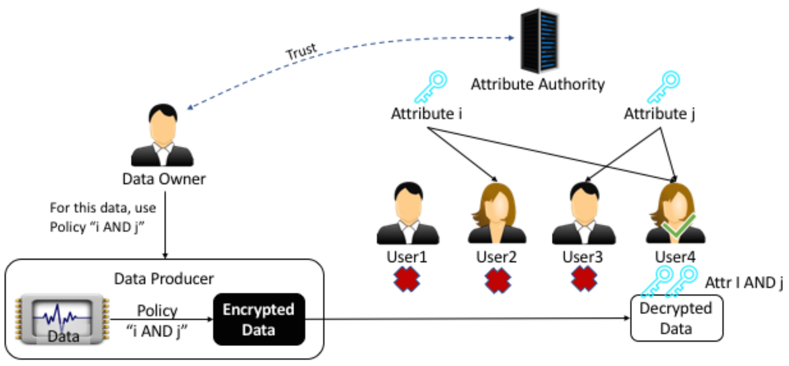


Figure 24. Cyphertext-Policy Attribute-Based Encryption.

However, in Ciphertext-Policy ABE (CP-ABE) cite:[Beth2007] this is reversed – a consumer's keys are associated with a set of attributes, while the encrypter can define the access tree structure associated with the encrypted data, essentially defining its access policy. That is to say, a data object's access policy is encrypted with the data object itself. This is illustrated in Figure 24. Here a Data Owner instructs a Data Producer to encrypt its data according to a specific policy defined over a set of attributes. In this example, the policy is simply i AND j which is used to define the access tree structure.

Clearly what the ABE approach does is exchange a data channel security requirement for a key distribution security requirement. However, we can argue that secure key distribution is easier to achieve since it can be a short event (a channel does not have to be secured for an indefinite period of time), and could even be done by out-of-band methods. One secure key distribution mechanism for Virtual Organizations was demonstrated in cite:[Lee2018], albeit within a Named Data Network environment. Nonetheless, the same approach could be implemented in TCP/IP-based networks.

9.7. Trusted Execution Environments

While *Trusted Execution Environments (TEEs)* may not be directly related to federations, the development and strong interest in TEEs as a hardware-based security mechanism merits further consideration. The question is: *Can TEEs be integrated and used in federated environments to achieve even greater security in federations.*

A TEE is a region of memory that is protected by specialized hardware in the processor and managed through additional instructions in the processor’s instruction set cite:[Shepherd2018,LeeD2020]. Data and code are only decrypted when they are placed into a TEE. Code running in the TEE can access all parts of memory, including data that is also in the TEE. Applications outside the TEE, however, cannot read nor write any memory locations within the TEE. This is enforced by processor hardware.

TEEs are offered on several chipsets cite:[LeeD2020]. The Intel *Software Guard eXtensions (SGX)* to the x86-64 instruction set enables applications to create secure *memory enclaves*. The *GlobalPlatform TEE* in an *ARM TrustZone* maintains two separate environments for all trusted and untrusted applications. Each environment has its own virtual core. The physical processor has an extra CPU bit that determines which environment is being serviced. The AMD *Secure Encrypted Virtualization (SEV)* enables memory in a virtual machine to be encrypted. The SEV can compute a signature for the encrypted data that the (potentially remote) data owner can use for attestations of integrity.

These vendor-specific TEEs are all quite different in their approach and capabilities. This has prompted work to develop hardware *security primitives* whereby customizable TEEs cite:[LeeD2020] can be constructed. Other efforts are underway to virtualize TEEs for mobile devices cite:[Li2019], such as cellphones. Since TEEs could be used in cloud environments where a data owner is hosting their data on somebody else’s hardware, work is also being done on managing *TEE-to-TEE* communication with *remote credential management* cite:[Shepherd2018].

This strong interest and wide range of efforts has prompted the establishment of the *Confidential Computing Consortium (CCC)* cite:[CCC] that is being hosted under the Linux Foundation with the goal of accelerating the development and adoption of TEE technologies through open source projects and eventually standards. To this end, the Consortium leadership has identified many use cases for TEEs cite:[CCC_WP]. The *Multi-Party Computing* use case is called out to support *federated analytics*, where data confidentiality and integrity can be ensured when the data is shared across platforms. In this use case, encrypted data can be sent to a remote TEE where it is decrypted. The data can be analyzed, possibly in conjunction other data, without exposing the data or analytical code to entities outside the TEE. When processing is complete, the results are encrypted and returned. The CCC emphasized that this capability will "help drive the advancement of a global data-sharing playing field" for "collaborative analysis and exchange".

This Multi-Party Computing use case has obvious similarities with federated environments. Can the discovery and use of remote TEEs be managed as a service in a federation? One possible approach is for a remote user to containerize and encrypt their code and data for execution in a remote TEE. As with other mobile code applications, the owner of the hosting TEE may want to vet the container to prevent malicious activity. Other mechanisms should be in place whereby the owner of the hosting TEE can limit and control what other data or resources the container hosted within the TEE can access. These are interesting possibilities that need to be resolved. A more in-depth investigation, however, must be left for future work.

9.8. Homomorphic Computing

Homomorphic computing is the ability to compute on encrypted data. This is not a new concept, but offers a clear benefit in the age of cloud computing. Homomorphic computing enables data to be sent to "the cloud", processed, and returned without ever being decrypted when in "the cloud". The same could be said for computing in a federated environment where data is being shared among federation members across different organizations.

The concept of homomorphic encryption was first proposed in 1978 and it was not clear if any solutions existed. Several partial solution algorithms were eventually developed, but the first algorithm that was considered *fully homomorphic* was not developed until 2009. This led to Microsoft developing the *Simple Encrypted Arithmetic Library (SEAL) 1.0 Cryptonets* in 2015. SEAL cite:[Kannepalli2019] was open-sourced in 2018.

While a full review of the use of lattice-based cryptography to produce a homomorphic encryption scheme is outside of the scope of this Engineering Report, what is directly relevant here is the *performance* of these algorithms. While the reduction in processing times for homomorphic computing has been impressive, it’s still prohibitive for most application domains. Current implementations impose a ~1000x increase in processing time, and a ~6x increase in storage requirements cite:[Kannepalli2019a]. The overhead of homomorphic encryption can be amortized by doing multiple operations on the data once encrypted, but the overhead is still very significant.

While this seems daunting, it may be attractive in some application domains, such as healthcare informatics, where the need for privacy is paramount. To this end, work is being done to reduce process times further. This includes the use of GPUs, FPGAs, and ASICs cite:[roy2019fpga], and investigating specific application domains where the structure of data can be exploited to reduce processing times cite:[foltz2019erp]. Further development on Microsoft SEAL is being documented at https://sealcrypto.org and work on standardization is being documented at https://homomorphicencryption.org.

At some point, progress on reducing the overheads may "tip the scales" in favor of using homomorphic computing when the need for privacy makes the trade-off worthwhile. However, in a cloud environment, the overhead of homomorphic encryption translates not only into longer compute times but also into additional costs. If running on a cloud, compute times that are 1000x longer will translate into a bill that is 1000x larger at the end of the month. Even at a factor of 10x, this may still be prohibitive for many application domains. For these reasons, it is safe to conclude at this time that homomorphic computing is not generally practical or applicable for security in federated environments.

9.9. Quantum Key Distribution

We mention Quantum Key Distribution (QKD) since there is wider interest in quantum computing in DoD and other government agencies. As mentioned elsewhere, federation depends on establishing common, trusted knowledge across the participants of a federated environment. This can be facilitated by the distribution of cryptographic keys among those participants.

When two parties share a random, secret key, this key can be used to encrypt and decrypt messages. QKD enables such keys to be shared in such a way that any eavesdropping or corruption of the keys being distributed can be detected. When using quantum entanglement to transmit keys in quantum states, any measurement of those quantum states disturbs the system. Hence, any eavesdropping or corruption of the keys could be detected.

While a fascinating communication technology, QKD only addresses a small portion of the overall security problem cite:[QKD]. QKD only addresses the distribution of keys, but does not address the communication of messages themselves. As such, QKD does not address verifying identities, data integrity, or access control. These functions must still be done by traditional symmetric key cryptography. QKD also has the drawbacks of currently having a short range of transmission, and the hardware is simply expensive. For these reasons, we conclude that QKD is not a practical option for managing federations at this time.

Chapter 10. Possible Development and Adoption Strategies

Out of all of these security concepts and approaches, which ones are the most relevant for supporting federations? Which approaches are viable? What are the pros and cons? How do they rank in terms of market interest and/or investment of resources?

Where possible, any federation development plan should clearly leverage, where possible, current efforts that have significant interest and investment. To this end, this section provides a review of the security concepts and technologies reviewed to assess their *viability*, *relevance*, and *maturity*. This assessment will be used to construct a possible *federation development roadmap* with the implicit assumption that the *target architecture* is based on the NIST CFRA concepts.

10.1. An Assessment for Viability, Relevance, and Maturity

To assess the possible approaches to enhance security in federated environments, these factors will be taken into account:

● *Viability:* Some technologies may have some very interesting properties but their practical application is just not feasible. While this may change over time, technologies that can be used sooner than later have a clear advantage.

● *Relevance:* Technologies that embody fundamental concepts will be more impactful, and thus relevant. That is to say, fundamental technologies are broadly applicable across a wide range of application domains.

● *Maturity:* A mature technology is gaining interest, adoption and investment from stakeholders in the marketplace. Any relevant technology that already has broad interest and investment should be leveraged for the development of federations.

From these considerations, three categories can be defined to assess technologies and guide their organization into a roadmap.

● *Not Ready:* As interesting as some technologies may be, many are just not ready for practical applications, and specifically for OGC testbed tasks.

● *Established:* Many relevant technologies are already well-established. Because of their wide use, such technologies should be considered in federation tooling.

● *Emerging:* The development of technologies and their adoption in the marketplace occurs over years. However, technologies that are viable and relevant may be clearly gaining maturity. These technologies merit special consideration for federations.

There are several technologies that are *established*, widely used, and trusted by many OGC stakeholders. These include PKI and SAML. However, OIDC is gaining more traction and the OpenID Foundation is facilitating the use of OIDC in distributed environments through the development of

OIDC Federation and also the OIDC Shared Signals an Events concept. These are sizeable investments that federated environments should leverage.

Since Verifiable Credentials (VCs) enables users to manage their own credentials, VCs may gain adoption in the wider commercial marketplace. However, controlling how a user may use their credentials in presentations is currently problematic. For this reason, VCs are considered *not ready*. While PKI and SAML will certainly remain to be widely used, OIDC and OIDC Federation are considered to be *emerging* for the purposes of future work to leverage the federation support that is being developed in the OIDC community.

Being able to handle legal agreements and assess regulatory compliance with cryptographically signed documents will be an immense milestone for the adoption of federations. Being able to do so in an automated fashion will be an even larger milestone. Federation trustmark frameworks are certainly a method to accomplish this. However, while trustmark frameworks are definitely foundational, the organizational and cultural adoption challenges will be significant. Hence, trustmark frameworks are *emerging* but with a developmental timeline that is expected to be quite long.

Blockchain is certainly an interesting technology that has public recognition owing to its use in cryptocurrency. The ability to record transactions in a way that is exceedingly hard to deny or corrupt has broad applicability. For example, blockchain could be used to establish shared, trusted information across federated environments. This could also include establishing the *provenance* of data and resources being shared. However, work needs to be done to better understand the potential blockchain behavior with regards to different consensus algorithms, block data size, and overall performance in different application use cases in federated environments. For these reasons, Blockchain is considered *emerging*.

In contrast, *Zero Trust Architectures* provide an alternative security approach through data confinement. The Zero Trust concept is gaining recognition in many sectors, including US government agencies that have strong security requirements. This is driving the investment by NIST and others in this technology. From the discussion in Zero Trust Federations, the integration of this concept into federations appears to be viable and relevant. Hence, ZTFs are *emerging*.

The investment in *Data-Centric Security (DCS)* that OGC has made in Testbeds and standards work indicates relevance and impact that should be leveraged in the near-term. Since possible integration approaches have already been identified, DCS is *emerging*. The OGC Security DWG and a significant set of OGC stakeholders have also expressed an interest in *Trusted Data Formats (TDFs)*. TDFs represent another approach to managing data confinement and preventing data leakage. While TDFs may not have wide adoption in the general marketplace, they have been deployed in operational government systems. Given the high relevance of data confinement in federated environments, TDFs are considered *established*.

While Attributed-Based Encryption (ABE) has strong similarities with TDF, it does have a drawback. While encoding the access policies with the data may be advantageous in some scenarios, this does mean that changing the policy after the ABE-encoded data has been disseminated will be problematic. Unless specific use cases can be identified that directly benefit from the ABE approach, ABE are considered to be *not ready* for now.

Trusted Execution Environments (TEEs) are gaining interest in the wider marketplace, as demonstrated by the creation of the *Confidential Computing Consortium*. Its relevance to federations is clearly indicated by the *Multi-Party Computing* use case. Since TEEs represent yet another approach to data confinement with significant interest to be leveraged, it is considered *emerging*. Since TEEs involve *Hardware Roots of Trust* and can support attestation, Hardware Roots of Trust is considered to be *emerging*, as well.

The overhead of Homomorphic Computing is still quite prohibitive. Since there are several other, more viable approaches to data security and confinement, Homomorphic Computing is considered to be *not ready* for now. Progress in this field can still be monitored to determine if the overheads can be reduced to an acceptable point for application domains that have a strong requirement for this type of security.

Finally, Quantum Key Distribution (QKD) is considered *not ready*. QKD is far too complicated for the partial benefits it can provide in the foreseeable future. As with all interesting technologies, QKD should be monitored for any significant progress being made.

10.2. A Federation Development Roadmap

Based on this review, a basic roadmap consisting of three phases can be defined. *Phase 1 Tasks* can be undertaken immediately to develop federations with greatly enhanced security. *Phase 2 Tasks* are quite important but need to build on Phase 1. *Phase 3 Tasks* are important but involve foundational concepts that will need time to development and mature with the benefit of experience. The nomenclature of *Task Px.y* to denote *Task y* in *Phase x* is used.

10.2.1. Roadmap Phase 1: Near-term Tasks

- **Task P1.1: Develop Gateway-based Federation Hosting Services**
  Based on the FHS design concepts incorporating OIDC as described in the NIST CFRA cite:[CFRA2020], the the IEEE P2302 WG cite:[P2302] had developed a core API set. This includes APIs for Operator-to-FHS, Member-to-FHS, and FHS-to-FHS interactions. As part of P2302's planned API roadmap, at the time of this writing, monitoring support is being added to the API. Subsequent work includes support for accounting, auditing, legal agreements, and regulatory compliance, i.e., all other capabilities necessary for commercially viable and sustainable federations. OGC Stakeholders could gain valuable experience by building a prototype to the currently existing IEEE P2302 API as part of Testbed-17. This would also provide a platform on which to base further testbed development in support of larger OGC goals, such earth observation.
- **Task P1.2: Develop the Use of Blockchain in Federations**
  Given that federations directly depend on shared, trusted information, the applicability of blockchain is clear. Near-term work could be done to evaluate consensus algorithms and scalability in the operation of Federated Hosting Services. Studies could also be done to compare the blockchain consensus approach with other data confinement approaches, such as Zero Trust Federations.
- **Task P1.3: Establish Trust Using the OIDC Federation Specification**
  If a gateway-based FHS incorporating OIDC is available, then the OIDC Federation Specification, v1 cite:[OIDC_Fed], could also be integrated. As discussed in the The OIDC Federation Specification section, this can be used to establish trust and secure communication between

multiple FHSs. There are at least three implementations underway, and implementation feedback is being incorporated into subsequence drafts of the standard.

- **Task P1.4: Integrate Data-Centric Security**
  The Data-Centric Security task in Testbed-16 has created a capability to manage key distribution for encrypted data objects. As discussed in the Data-Centric Security section, there are multiple scenarios for using the DCS approach in a federated environment. Doing so would greatly increase the security posture of federations.

- **Task P1.5: Develop Zero Trust Federations**
  Integrating a DCS capability within FHS-managed federations would be an integral part of developing Zero Trust Federations. As discussed in the Zero Trust Federations section, this would be done in conjunction with making the implicit trust zones as small as possible *at the gateway level*. To achieve even finer-grained protection, an FHS could be deployed "on top of" a Microservice Mesh Control Plane. While certainly workable, a longer range question is how the two could be integrated into a whole. For example, how to federate microservice control planes, and how to integrate arbitrary, external resources into a control plane federation. These are interesting questions but are not recommended for Phase 1.

10.2.2. Roadmap Phase 2: Subsequent Tasks

- **Task P2.1: Integrate Trusted Data Formats**
  Given that Trusted Data Formats are known to a user base with significant security requirements, the integration of TDFs into federated environments would provide an additional security mechanism. This will involve methods to establish trust and accessibility between a Trusted Data Object and an external verifier.

- **Task P2.2: Develop Policy-Based Data Confinement Methods**
  The use of NGAC's graph-based approach to policies could be used to manage federations, and specifically to prevent intentional third-party data leakage, also called *data confinement*. NGAC addresses this through the use of obligations to dynamically change policies. When the event "User reads datatype Y" occurs, the User's access policy is dynamically changed to include "User can no longer write datatype Y". The use of such dynamic policy management should be investigated for wider use in federations.

- **Task P2.3: Integrate Trusted Execution Environments**
  TEEs offer another method for data confinement. The strong industry interest in TEEs, and the Confidential Computing Consortium's focus on Multi-Party Computing, is a potential area of synergy with federations. Work in this area would focus on enabling TEEs to be accessed through an API Gateway and making credentials compatible or interoperable in some way.

- **Task P2.4: Integrate Hardware Roots of Trust**
  TEEs involve a hardware root of trust and can also be used to do software stack attestations. Hence, if TEEs are integrated into federations, it should also be possible to do attestations.

10.2.3. Roadmap Phase 3: Longer-term Tasks

- **Task P3.1: Review/Expand NIST SP 800-53**
  The NIST Security and Privacy Controls for Federal Information Systems and Organizations cite:[NIST800-53] is a large document that catalogs 20 families of *security controls*. There are 278 security controls and an even greater number of *control enhancements*. NIST SP 800-53 is used

extensively in US Government systems to define the controls used in their security posture. Different subsets of the NIST SP 800-53 controls are used to define the different levels of FedRAMP certification. This NIST document should be reviewed to identify all security controls relevant to federations, and to identify any new, federation-specific, security controls that should be added.

- **Task P3.2: Develop Formal Federation Definitions**
  A method of formal definitions for federations would be a major step towards the automation of many federation management functions. This task would involve the investigation, comparison, and possible development of a *Fed-ML* or *OWL-Fed*. Such an effort could be codified in a standard or best practices document that identifies the properties of different "types" or "levels" of federations, in much the same way that NIST SP 800-53 is used to define different levels of FedRAMP certification.
- **Task P3.3: Develop Federation Trust Frameworks**
  If formal definition methods are available, these could be used to build federated trust frameworks. Trust frameworks would greatly enable the commercial adoption of federations where firm legal agreements and regulatory compliance would have to be in place. While possible from a technical perspective, the organizational and cultural adoption challenges of federated trustmark frameworks will definitely make this a long-term goal.

Chapter 11. Final Comments

In February 2020, NIST officially published the NIST Cloud Federation Reference Architecture (CFRA) cite:[CFRA2020]. The CFRA approach is to support "purpose-built" federations - federations whose operation and governance is agreed upon by the stakeholders prior to creation. This avoids problems of trying to federate existing environments that were not intended nor designed with federated collaboration in mind. By taking this approach, many issues of policy, attribute release, and manageability can be avoided entirely.

Nonetheless, any federation approach, including the CFRA, has tremendous security concerns since the whole purpose of a federation is to span trust zones and expose users and resources from different organizations to one another. Hence, this OGC Engineering Report took a broad look at many developing trust security concepts to determine how federations could be made as secure as possible. The work began by reviewing the basics of trust and security, followed by a threat analysis based on possible implementation approaches This analysis demonstrated that the vulnerabilities of federations depend on the vulnerabilities of the underlying tools that are used to implement federations, i.e., tools such as TLS and OpenID Connect. What is different is the impact of any compromises.

To address these issues, emerging trust and security approaches were reviewed and evaluated for use in federated environments. A basic roadmap for further development was identified. However, this roadmap has a technical focus. What organizational and cultural adoption issues will dominant the development and adoption of federation currently are to be understated. The fact that the CFRA approach focuses on agreed-upon governance highlights the many differences that many organizations will have. Hence, concomitant efforts should be undertaken to promote organizational adoption. This can mean developing support tooling and demonstrations that unequivocally illustrate the broad value of federation for mission and societal benefits.

Doing so will require more than a single organization. Since federation is inherently *cross-organizational*, some type of *umbrella organization* is needed to facilitate development and adoption. By analogy with the Internet, in the 1970s the Advanced Research Projects Agency (ARPA) created *ARPANET*. ARPANET was essentially an umbrella environment whereby many universities, corporations and government agencies could bring their resources to the table and work together in the nascent field of computer networking. A similar type of umbrella organization would greatly facilitate federation development. The OGC will work to achieve this larger goal.

Appendix A: Threat Dragon Report

Threat Dragon can produce a PDF report file for any analysis that has been done. This report file essentially presents all of the information stored in the JSON project file in a nice, human-readable format. Included below is the report generated for the four threat analysis scenarios of the NIST CFRA-based Federation Service.

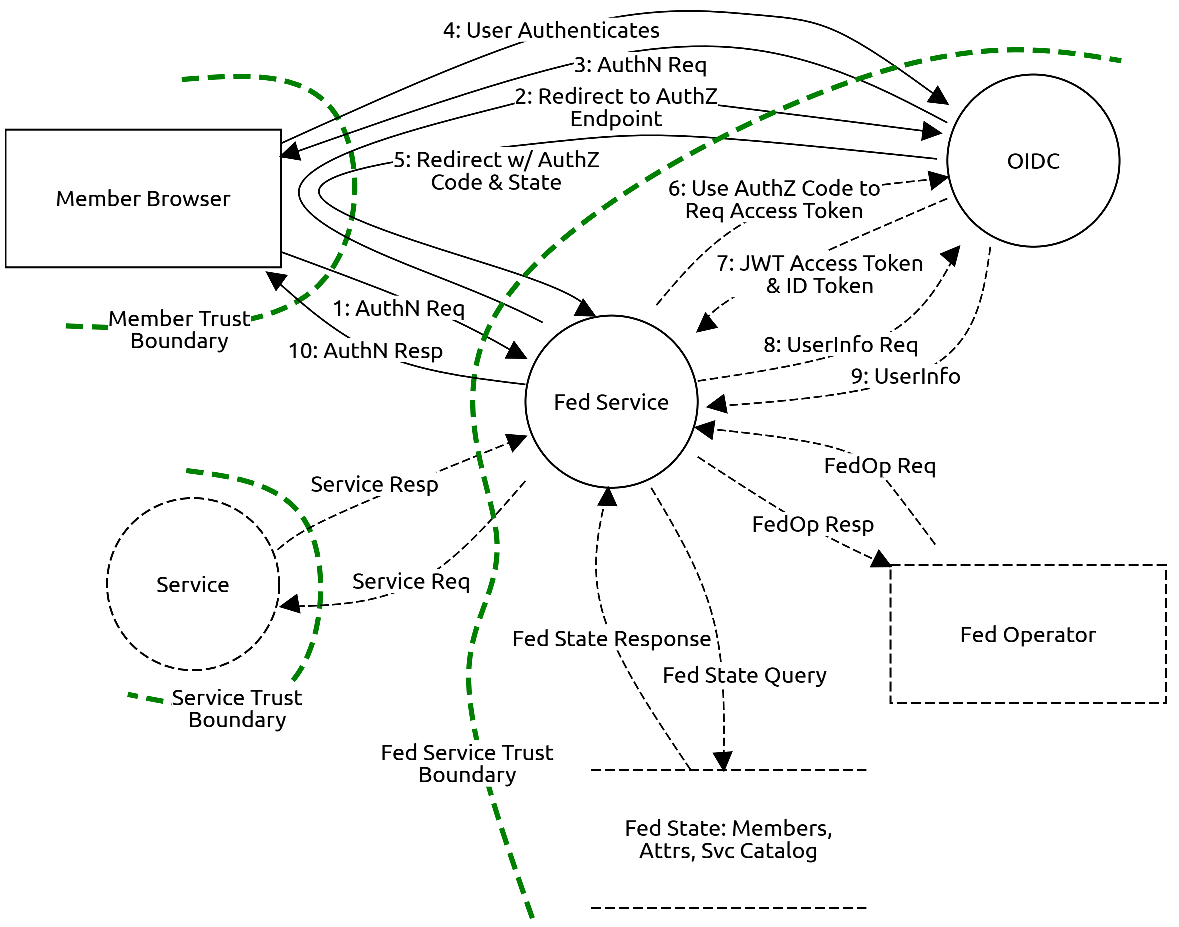
Threat model report for CFRA Threat Models

Owner:
Craig Lee
Reviewer:
Contributors:

High level system description

Initial threat model for a Federation Service (FS) based on the NIST Cloud Federation Reference Architecture. The Deployment & Governance model is one, third-party FS with one FedAdmin per federation instance. Four threat model diagrams are given: Member Authentication, FedAdmin, ResOwner, and User Interactions.

Authentication to a Federation



OIDC (Process)

Description:

[11] Section 4.4.1.4, Malicious client obtains authorization

Spoofing, Mitigated, High Severity

Description:
Malicious client pretends to be a valid client.

Mitigation:
Authorization server must validate client's redirection URI against pre-registered URI. Use of CAPTCHAs, Multi-factor authentication, random questions, etc.

[11] 4.1.5, Open Redirectors on client

Tampering, Mitigated, High Severity

Description:
An open redirector is an endpoint using a parameter. This allows an attacker to construct a URI that passes validation but returns the access code or access token to an endpoint under control of the attacker.

Mitigation:
Require clients to register full redirection URIs.

Repudiation of Authentication

Repudiation, Mitigated, High Severity

Description:
The Access and ID tokens delivered from the OIDC server to the Fed Service provide integrity and non-repudiation. The Fed Service could repudiate a Member's authentication only if the Fed Service was compromised.

Mitigation:
This illustrates how the mitigation of an internal threat depends on the mitigation of external threats, i.e., the prevention of an external malicious actor compromising a Fed Service.

Information disclosure threat

Information disclosure, Mitigated, High Severity

Description:
Attacker could eavesdrop on the redirections from the Fed Service or the interaction with the Member.

Mitigation:
Use transport-layer security mechanisms such as TLS.

[11] 4.4.1.11, Exhaustion of resources attack
Denial of service, Mitigated, Medium Severity

Description:
Attacker issues excess requests that exhaust the pool of authorization codes.

Mitigation:
Rate-limit the access codes issued per user. Rate-limit should be related to the entropy in authorization codes, i.e., the length.

[11] 4.4.1.12, DoS using manufactured authorization codes
Denial of service, Mitigated, Medium Severity

Description:
If an attacker can locate a client's redirection URI that listen on HTTP, then large numbers of random authorization codes on HTTPS connections can concentrate on the authorization server.

Mitigation:
Cross-Site Request Forgery (CSRF) tokens and 'state' parameters should be validated. After invalid requests exceed a limit, the Authorization Server should send an error to the client and rate-limit or disallow further connections.

Fed Service (Process)

Description:

[11] 4.1.4, End-user credentials phished
Spoofing, Mitigated, High Severity

Description:
Credentials, such as passwords, could be phished using a compromised or embedded browser.

Mitigation:
OAuth (and OIDC) are designed such that the Fed Service should never see Member passwords. Members should be educated to recognize phishing attacks.

[11] 4.1.5, Open Redirectors on client
Tampering, Mitigated, High Severity

Description:

An open redirector is an endpoint using a parameter. This allows an attacker to construct a URI that passes validation but returns the access code or access token to an endpoint under control of the attacker.

Mitigation:
Require clients to register full redirection URIs.

Repudiation of Authentication
Repudiation, Mitigated, High Severity

Description:
While the Member is "authenticating" to the Fed Service, it is actually redirected to the OIDC server for authentication. The Access and ID tokens for the Member are subsequently delivered directly from the OIDC server to the Fed Service. These tokens must be protected and the Fed Service must not be compromised.

Mitigation:
The signed Access and ID tokens that the OIDC delivers to the Fed Service provide authentication with integrity and non-repudiation. These tokens are stored in a separate store that is not accessible to external actors. To mitigate repudiation of authentication, all other methods to compromise a Fed Service must be mitigated.

[11] 4.1.1, 4.1.2, 4.1.3, Obtain client secrets.
Information disclosure, Mitigated, High Severity

Description:
In this deployment model, the Fed Service is the OIDC client. This means the Fed Service is storing the Access and ID Tokens for all Members using this Fed Service.

Mitigation:
These tokens are stored in secure local storage, and never in binary or source code. Standard web server protection should be applied. Token scope should be limited to only the necessary authorizations. Token lifetime should be limited. If a token is compromised, the OIDC Server should immediately revoke the token to prevent any further abuse.

Exhaustion of resource attacks
Denial of service, Mitigated, Medium Severity

Description:
Similar to [11] 4.4.1.11, a malicious actor (member or not) could send excessive requests to a Fed Service.

Mitigation:
Fed Service should rate limit the number of requests per requestor, member or not. Distributed Denial of Service attacks are more problematic.

Elevation threat

Elevation of privilege, Mitigated, High Severity

Description:
Similar to [11] 4.2.4, an open redirector could allow a current member to maliciously obtain authorization codes intended for another member, and elevate their own privilege.

Mitigation:
Require the Fed Service to register full redirection URIs that can be validated on each redirection.

Disclosure of client information during transmission

Information disclosure, Mitigated, High Severity

Description:
Similar to [11] 4.3.3, an attacker could eavesdrop on communications.

Mitigation:
Data in transmission should be protected using transport-layer mechanisms such as TLS.

Member Browser (External Actor)

Description:

[11] 4.4.1.9, Click-jacking attack against authorization

Spoofing, Mitigated, Medium Severity

Description:
An attacker loads the target site (a Fed Service) in a transparent iFrame constructed to look just like the target site, allowing the attacker to intercept keystrokes and button clicks, thus allowing the attacker to steal authentication credentials.

Mitigation:
Only allow the use of newer browsers that support the X-FRAME-OPTION header. This enables iFrames to be completely denied, allowed only from the same origin, or from a list trusted origins.

Repudiation of Authentication

Repudiation, Mitigated, High Severity

Description:

If a Fed Service repudiates that a Member has successfully authenticated, this indicates that the Fed Service, OIDC Server, the Fed State, or the FM Operator has been compromised, or any combination thereof.

Mitigation:
The mitigation of this threat depends on the mitigation of any compromise threats to these internal actors. Attestation based on hardware Trusted Computing Modules could be used to ensure that these software stacks have not been modified. Log files could be constantly monitored to detect any unusual or unexpected activities. All threats to externally accessible endpoints must be identified and mitigated.

1: AuthN Req (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:
Man-in-the-Middle attacks could occur. DNS or ARP spoofing attacks could occur.

Mitigation:
The Member should validate the binding of the Fed Service to its domain name. Transport-layer security mechanisms, such as TLS, should be used.

Information disclosure threat

Information disclosure, Mitigated, Medium Severity

Description:
An attacker could eavesdrop on communication between a Member and the Fed Service.

Mitigation:
Use transport-layer security mechanisms such as TLS.

2: Redirect to AuthZ Endpoint (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:
Man-in-the-Middle attacks could occur. DNS or ARP spoofing attacks could occur.

Mitigation:
The Fed Service should validate the binding of the OIDC Server's AuthZ endpoint to its domain name. Transport-layer security mechanisms, such as TLS, should be used.

Information disclosure threat

Information disclosure, Mitigated, Medium Severity

Description:
An attacker could eavesdrop this redirection.

Mitigation:
Use transport-layer security mechanisms such as TLS.

[11] 4.4.1.8, Cross-Site Request Forgery against redirection URI

Elevation of privilege, Mitigated, High Severity

Description:
After [11] 4.4.1.8: an attacker could authorize an authorization code to their own protected resources on an authorization server. The attacker then aborts the redirect flow back to the Fed Service and tricks the victim Member into executing the redirect back to the Fed Service. The Fed Service receives the redirect, fetches the token(s) from the authorization server, and associates the victim Member's session with the resources accessible using the token. This allows the attacker to access data from the Member's resources in this federation.

Mitigation:
The state parameter should be used to link the authorization request with the redirection URI. See [11] 5.3.5.

5: Redirect w/ AuthZ Code & State (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:
Man-in-the-Middle attacks could occur. DNS or ARP spoofing attacks could occur.

Mitigation:

The OIDC Server should validate the binding of the Fed Service's externally accessible Member API Endpoint to its domain name. Transport-layer security mechanisms, such as TLS, should be used.

Information disclosure threat

Information disclosure, Mitigated, Medium Severity

Description:
An attacker could eavesdrop this redirection.

Mitigation:
Use transport-layer security mechanisms such as TLS.

[11] 4.4.1.8, Cross-Site Request Forgery against redirection URI

Elevation of privilege, Mitigated, High Severity

Description:
After [11] 4.4.1.8: an attacker could authorize an authorization code to their own protected resources on an authorization server. The attacker then aborts the redirect flow back to the Fed Service and tricks the victim Member into executing the redirect back to the Fed Service. The Fed Service receives the redirect, fetches the token(s) from the authorization server, and associates the victim Member's session with the resources accessible using the token. This allows the attacker to access data from the Member's resources in this federation.

Mitigation:
The state parameter should be used to link the authorization request with the redirection URI. See [11] 5.3.5.

3: AuthN Req (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:
Man-in-the-Middle attacks could occur. DNS or ARP spoofing attacks could occur.

Mitigation:
The OIDC Server should validate the binding of the Member to its domain name. Transport-layer security mechanisms, such as TLS, should be used.

Information disclosure threat

Information disclosure, Mitigated, Medium Severity

Description:
An attacker could eavesdrop on communication between a Member and the OIDC Server.

Mitigation:
Use transport-layer security mechanisms such as TLS.

4: User Authenticates (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:
Man-in-the-Middle attacks could occur. DNS or ARP spoofing attacks could occur.

Mitigation:
The Member should validate the binding of the Fed Service to its domain name. Transport-layer security mechanisms, such as TLS, should be used.

Information disclosure threat

Information disclosure, Mitigated, Medium Severity

Description:
An attacker could eavesdrop on communication between a Member and the OIDC Server.

Mitigation:
Use transport-layer security mechanisms such as TLS.

10: AuthN Resp (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:
Man-in-the-Middle attacks could occur. DNS or ARP spoofing attacks could occur.

Mitigation:
The Member should validate the binding of the Fed Service to its domain name. Transport-layer security mechanisms, such as TLS, should be used.

Information disclosure threat

Information disclosure, Mitigated, Medium Severity

Description:
An attacker could eavesdrop on communication between a Member and the Fed Service.

Mitigation:
Use transport-layer security mechanisms such as TLS.

Fed State: Members, Attrs, Svc Catalog (out of scope Data Store)

Description:
Members, Attributes, Service Catalog

Out of scope reason:
Considered to be secure behind the Fed Service Trust Boundary.

Fed State Query (out of scope Data Flow)

Description:

Out of scope reason:
Considered to be secure behind the Fed Service Trust Boundary.

Fed State Response (out of scope Data Flow)

Description:

Out of scope reason:
Considered to be secure behind the Fed Service Trust Boundary.

7: JWT Access Token & ID Token (out of scope Data Flow)

Description:

Out of scope reason:
Considered to be secure behind the Fed Service Trust Boundary.

Fed Operator (out of scope External Actor)

Description:

Out of scope reason:
Considered to be secure behind the Fed Service Trust Boundary.

Service Req (out of scope Data Flow)

Description:

Out of scope reason:
Not involved in initial authentication.

FedOp Req (out of scope Data Flow)

Description:

Out of scope reason:
Considered to be secure behind the Fed Service Trust Boundary.

FedOp Resp (out of scope Data Flow)

Description:

Out of scope reason:
Considered to be secure behind the Fed Service Trust Boundary.

Service Resp (out of scope Data Flow)

Description:

Out of scope reason:
Not involved in initial authentication.

8: UserInfo Req (out of scope Data Flow)

Description:

Out of scope reason:
Considered to be secure behind the Fed Service Trust Boundary.

6: Use AuthZ Code to Req Access Token (out of scope Data Flow)

Description:

Out of scope reason:
Considered to be secure behind the Fed Service Trust Boundary.

9: UserInfo (out of scope Data Flow)

Description:

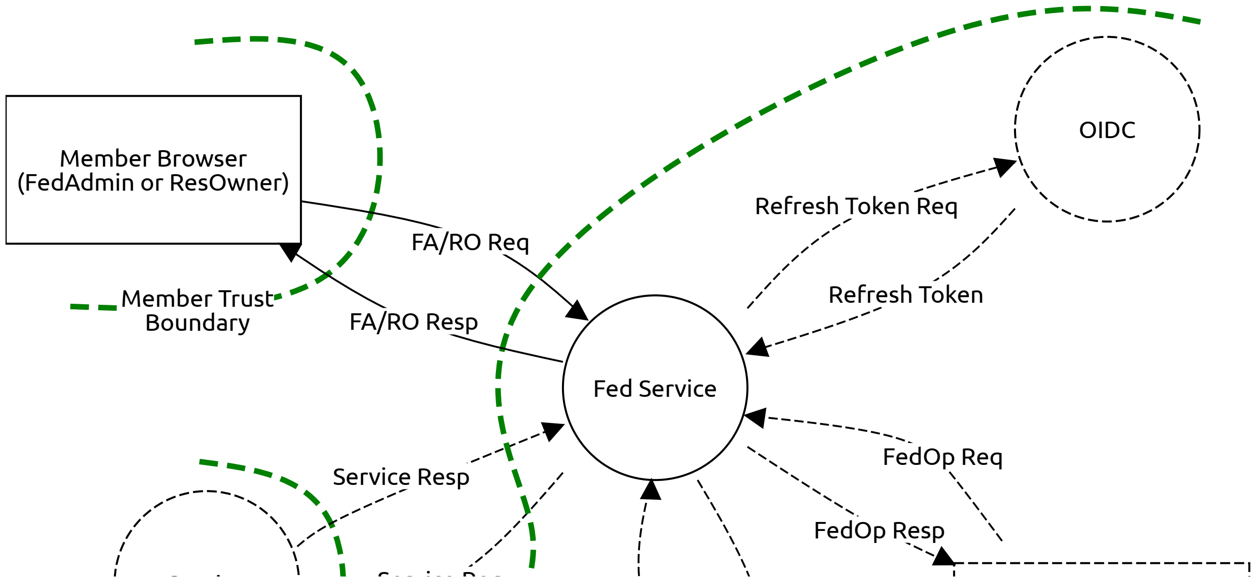
Out of scope reason:
Considered to be secure behind the Fed Service Trust Boundary.

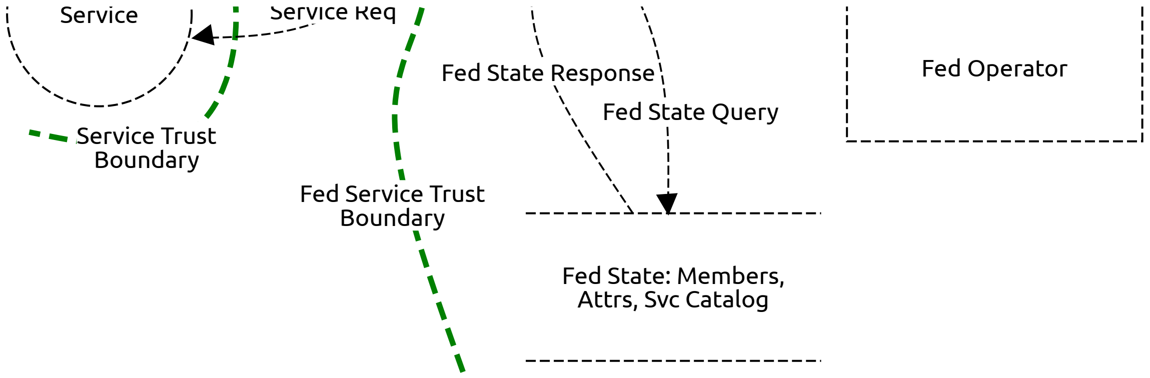
Service (out of scope Process)

Description:

Out of scope reason:
Not involved in initial authentication process.

FedAdmin/ResOwner Interactions





Fed Service (Process)

Description:

[11] 4.1.4, End-user credentials phished

Spoofing, Mitigated, High Severity

Description:
Credentials, such as passwords, could be phished using a compromised or embedded browser.

Mitigation:
OAuth (and OIDC) are designed such that the Fed Service should never see Member passwords. Members should be educated to recognize phishing attacks.

Repudiation of Authentication

Repudiation, Mitigated, High Severity

Description:
When a Member authenticates to the Fed Service, the Access and ID tokens for the Member are subsequently delivered directly from the OIDC server to the Fed Service. These tokens must be protected and the Fed Service must not be compromised.

Mitigation:
The signed Access and ID tokens that the OIDC delivers to the Fed Service provide authentication with integrity and non-repudiation. These tokens are stored in a separate store that is not accessible to external actors. To mitigate repudiation of authentication, all other methods to compromise a Fed Service must be mitigated.

[11] 4.1.1, 4.1.2, 4.1.3, Obtain client secrets.

Information disclosure, Mitigated, High Severity

Description:
In this deployment model, the Fed Service is the OIDC client. This means the Fed Service is storing the Access and ID Tokens for all Members using this Fed Service.

Mitigation:

These tokens are stored in secure local storage, and never in binary or source code. Standard web server protection should be applied. Token scope should be limited to only the necessary authorizations. Token lifetime should be limited. If a token is compromised, the OIDC Server should immediately revoke the token to prevent any further abuse.

Exhaustion of resource attacks

Denial of service, Mitigated, Medium Severity

Description:
Similar to [11] 4.4.1.11, a malicious actor (member or not) could send excessive requests to a Fed Service.

Mitigation:
Fed Service should rate limit the number of requests per requestor, member or not. Distributed Denial of Service attacks are more problematic.

Disclosure of client information during transmission

Information disclosure, Mitigated, High Severity

Description:
Similar to [11] 4.3.3, an attacker could eavesdrop on communications.

Mitigation:
Data in transmission should be protected using transport-layer mechanisms such as TLS.

Member Browser (FedAdmin or ResOwner) (External Actor)

Description:

Repudiation of Authentication

Repudiation, Mitigated, High Severity

Description:
If a Fed Service repudiates that a Member has successfully authenticated, this indicates that the Fed Service, OIDC Server, the Fed State, or the FS Operator has been compromised, or any combination thereof.

Mitigation:
The mitigation of this threat depends on the mitigation of any compromise threats to these internal actors. Attestation based on hardware Trusted Computing Modules could be used to ensure that these software stacks have not been

modified. Log files could be constantly monitored to detect any unusual or unexpected activities. All threats to externally accessible endpoints must be identified and mitigated.

FA/RO Req (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:
Man-in-the-Middle attacks could occur. DNS or ARP spoofing attacks could occur.

Mitigation:
The Member should validate the binding of the Fed Service to its domain name. Transport-layer security mechanisms, such as TLS, should be used.

Information disclosure threat

Information disclosure, Mitigated, Medium Severity

Description:
An attacker could eavesdrop on communication between a Member and the Fed Service.

Mitigation:
Use transport-layer security mechanisms such as TLS.

FA/RO Resp (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:
Man-in-the-Middle attacks could occur. DNS or ARP spoo ng attacks could occur.

Mitigation:
The Member should validate the binding of the Fed Service to its domain name. Transport-layer security mechanisms, such as TLS, should be used.

Information disclosure threat

Information disclosure, Mitigated, High Severity

Description:
An attacker could eavesdrop on communication between a Member and the Fed Server.

Mitigation:
Use transport-layer security mechanisms such as TLS.

Fed State: Members, Attrs, Svc Catalog (out of scope Data Store)

Description:
Members, Attributes, Service Catalog

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

OIDC (out of scope Process)

Description:

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

Fed State Query (out of scope Data Flow)

Description:

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

Fed State Response (out of scope Data Flow)

Description:

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

Fed Operator (out of scope External Actor)

Description:

Out of scope reason:
Not involved in FedAdmin interactions.

Service Req (out of scope Data Flow)

Description:

Out of scope reason:
Not involved in administrative process.

FedOp Req (out of scope Data Flow)

Description:

Out of scope reason:
Not involved in FedAdmin interactions.

FedOp Resp (out of scope Data Flow)

Description:

Out of scope reason:
Not involved in FedAdmin interactions.

Service Resp (out of scope Data Flow)

Description:

Out of scope reason:
Not involved in administrative process.

Refresh Token Req (out of scope Data Flow)

Description:

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

Service (out of scope Process)

Description:

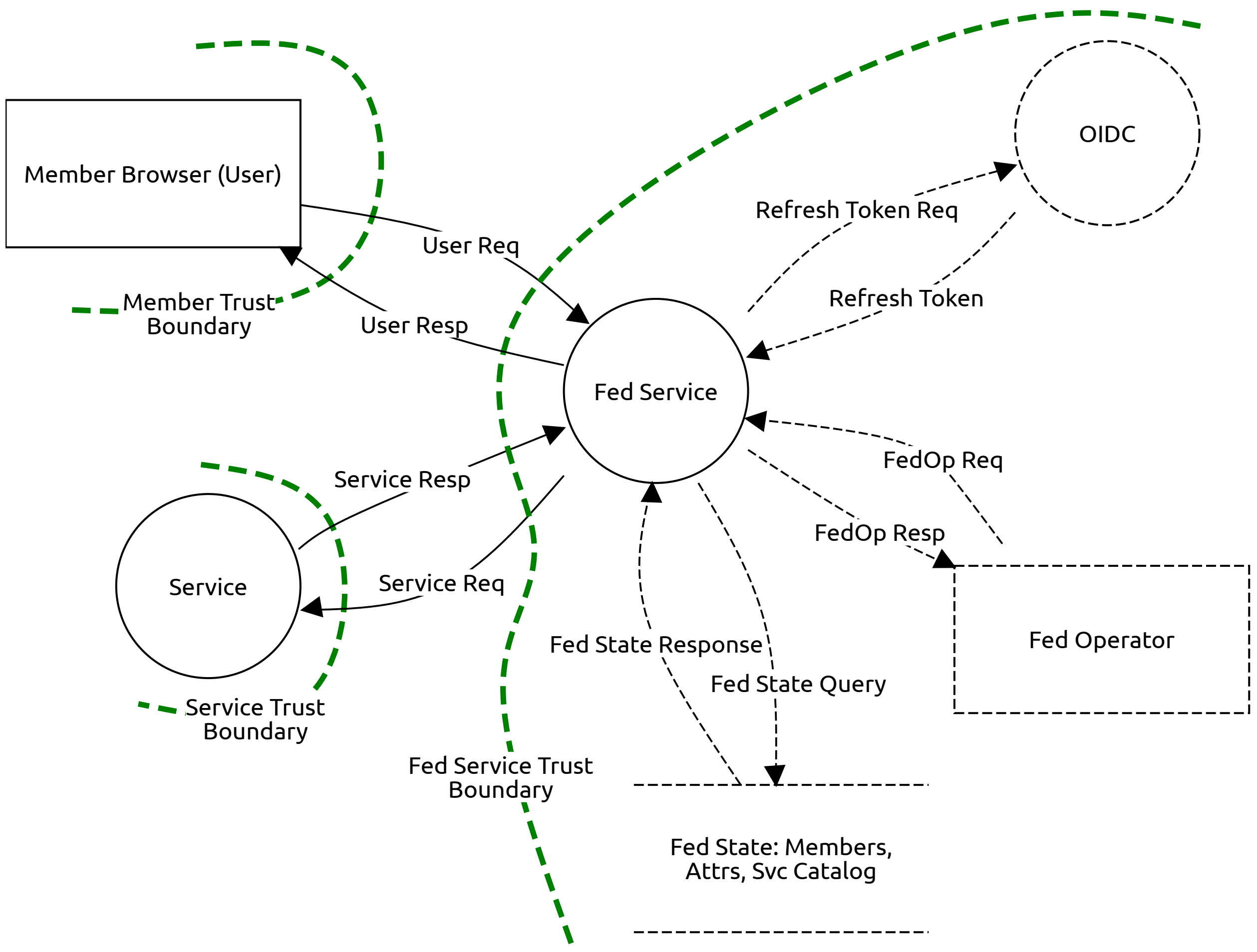
Out of scope reason:
Not involved in administrative process.

Refresh Token (out of scope Data Flow)

Description:

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

User Interactions



Fed Service (Process)

Description:

[11] 4.1.4, End-user credentials phished

Spoofing, Mitigated, High Severity

Description:
Credentials, such as passwords, could be phished using a compromised or embedded browser.

Mitigation:
OAuth (and OIDC) are designed such that the Fed Service should never see Member passwords. Members should be educated to recognize phishing attacks.

Repudiation of Authentication

Repudiation, Mitigated, High Severity

Description:
When a Member authenticates to the Fed Service, the Access and ID tokens for the Member are subsequently delivered directly from the OIDC server to the Fed Service. These tokens must be protected and the Fed Service must not be compromised.

Mitigation:
The signed Access and ID tokens that the OIDC delivers to the Fed Service provide authentication with integrity and non-repudiation. These tokens are stored in a separate store that is not accessible to external actors. To mitigate repudiation of authentication, all other methods to compromise a Fed Service must be mitigated.

[11] 4.1.1, 4.1.2, 4.1.3, Obtain client secrets.

Information disclosure, Mitigated, High Severity

Description:
In this deployment model, the Fed Service is the OIDC client. This means the Fed Service is storing the Access and ID Tokens for all Members using this Fed Service.

Mitigation:
These tokens are stored in secure local storage, and never in binary or source code. Standard web server protection should be applied. Token scope should be limited to only the necessary authorizations. Token lifetime should be limited. If a token is compromised, the OIDC Server should immediately revoke the token to prevent any further abuse.

Exhaustion of resource attacks

Denial of service, Mitigated, Medium Severity

Description:
Similar to [11] 4.4.1.11, a malicious actor (member or not) could send excessive requests to a Fed Service.

Mitigation:

Fed Service should rate limit the number of requests per requestor, member or not. Distributed Denial of Service attacks are more problematic.

Disclosure of client information during transmission

Information disclosure, Mitigated, High Severity

Description:
Similar to [11] 4.3.3, an attacker could eavesdrop on communications.

Mitigation:
Data in transmission should be protected using transport-layer mechanisms such as TLS.

Member Browser (User) (External Actor)

Description:

Repudiation of Authentication

Repudiation, Mitigated, High Severity

Description:
If a Fed Service repudiates that a Member has successfully authenticated, this indicates that the Fed Service, OIDC Server, the Fed State, or the FS Operator has been compromised, or any combination thereof.

Mitigation:
The mitigation of this threat depends on the mitigation of any compromise threats to these internal actors. Attestation based on hardware Trusted Computing Modules could be used to ensure that these software stacks have not been modified. Log files could be constantly monitored to detect any unusual or unexpected activities. All threats to externally accessible endpoints must be identified and mitigated.

User Req (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:

Man-in-the-Middle attacks could occur. DNS or ARP spoofing attacks could occur.

Mitigation:
The Member should validate the binding of the Fed Service to its domain name. Transport-layer security mechanisms, such as TLS, should be used.

Information disclosure threat

Information disclosure, Mitigated, Medium Severity

Description:
An attacker could eavesdrop on communication between a Member and the Fed Service.

Mitigation:
Use transport-layer security mechanisms such as TLS.

Service Req (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:
Man-in-the-Middle attacks could occur. DNS or ARP spoofing attacks could occur.

Mitigation:
Use transport-layer security mechanisms such as TLS.

Information disclosure threat

Information disclosure, Mitigated, Medium Severity

Description:
An attacker could eavesdrop on communication between a Service and the Fed Service.

Mitigation:
Use transport-layer security mechanisms such as TLS.

Service Resp (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:
Man-in-the-Middle attacks could occur. DNS or ARP spoofing attacks could occur.

Mitigation:
The Service should validate the binding of the Fed Service to its domain name. Transport-layer security mechanisms, such as TLS, should be used.

Information disclosure threat

Information disclosure, Mitigated, Medium Severity

Description:
An attacker could eavesdrop on communication between a Service and the Fed Service.

Mitigation:
Use transport-layer security mechanisms such as TLS.

Service (Process)

Description:
This is an arbitrary service being made available to the federation members.

Disclosure of information during transmission

Information disclosure, Mitigated, High Severity

Description:
Similar to [11] 4.3.3, an attacker could eavesdrop on communications.

Mitigation:
Data in transmission should be protected using transport-layer mechanisms such as TLS.

Exhaustion of resource attacks

Denial of service, Mitigated, Medium Severity

Description:
Similar to [11] 4.4.1.11, a malicious actor (member or not) could send excessive requests to a Service.

Mitigation:
Service should rate limit the number of requests per requestor, member or not. Distributed Denial of Service attacks are more problematic.

User Resp (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:
Man-in-the-Middle attacks could occur. DNS or ARP spoofing attacks could occur.

Mitigation:
The Fed Service should validate the binding of the Member to its domain name. Transport-layer security mechanisms, such as TLS, should be used.

Information disclosure threat

Information disclosure, Mitigated, Medium Severity

Description:
An attacker could eavesdrop on communication between a Member and the Fed Server.

Mitigation:
Use transport-layer security mechanisms such as TLS.

Fed State: Members, Attrs, Svc Catalog (out of scope Data Store)

Description:
Members, Attributes, Service Catalog

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

OIDC (out of scope Process)

Description:

Out of scope reason:

Considered secure behind the Fed Service Trust Boundary.

Fed State Query (out of scope Data Flow)

Description:

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

Fed State Response (out of scope Data Flow)

Description:

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

Fed Operator (out of scope External Actor)

Description:

Out of scope reason:
Not involved in FedAdmin interactions.

FedOp Req (out of scope Data Flow)

Description:

Out of scope reason:
Not involved in FedAdmin interactions.

FedOp Resp (out of scope Data Flow)

Description:

Out of scope reason:
Not involved in FedAdmin interactions.

Refresh Token Req (out of scope Data Flow)

Description:

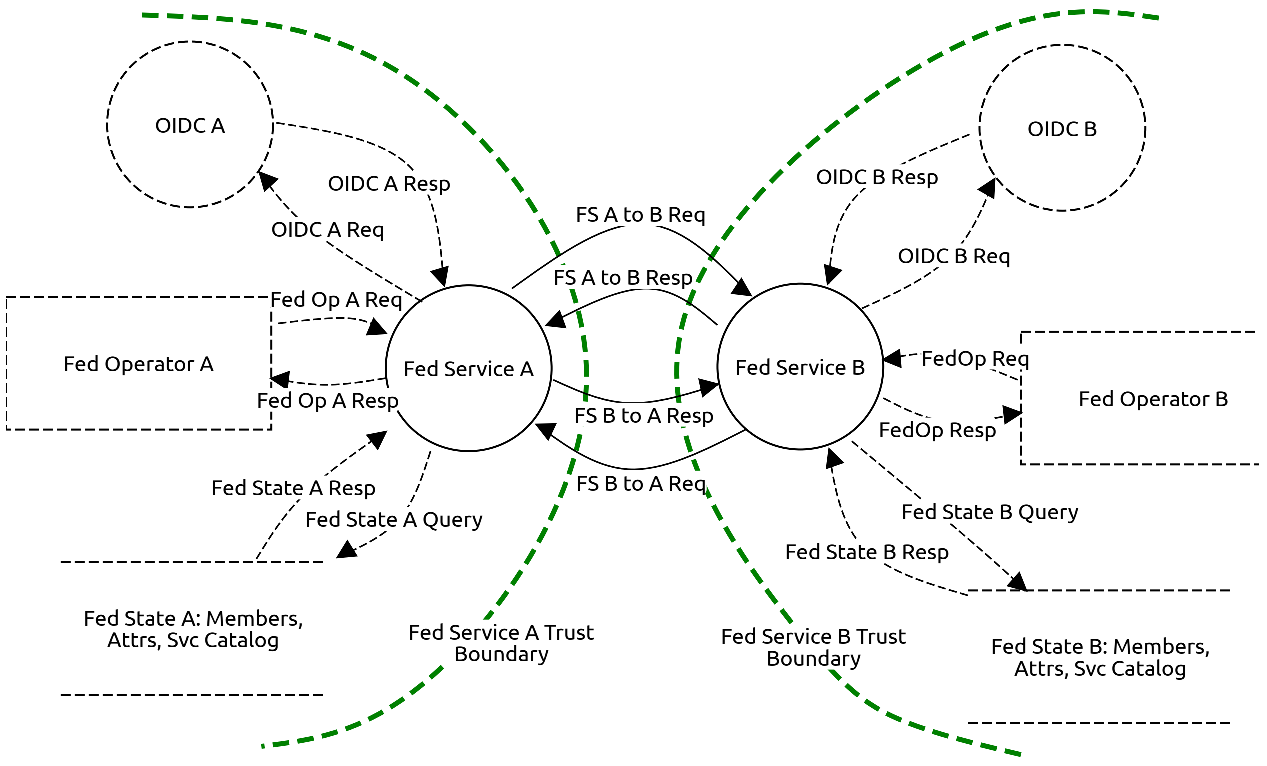
Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

Refresh Token (out of scope Data Flow)

Description:

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

FS-FS Interactions



Fed Service B (Process)

Description:

[11] 4.1.4, End-user credentials phished

Spoofing, Mitigated, High Severity

Description:
Credentials, such as passwords, could be phished using a compromised or embedded browser.

Mitigation:
OAuth (and OIDC) are designed such that the Fed Service should never see Member passwords. Members should be educated to recognize phishing attacks.

Repudiation of Authentication

Repudiation, Mitigated, High Severity

Description:
When a Member authenticates to the Fed Service, the Access and ID tokens for the Member are subsequently delivered directly from the OIDC server to the Fed Service. These tokens must be protected and the Fed Service must not be compromised.

Mitigation:
The signed Access and ID tokens that the OIDC delivers to the Fed Service provide authentication with integrity and non-repudiation. These tokens are stored in a separate store that is not accessible to external actors. To mitigate repudiation of authentication, all other methods to compromise a Fed Service must be mitigated.

[11] 4.1.1, 4.1.2, 4.1.3, Obtain client secrets.

Information disclosure, Mitigated, High Severity

Description:
In this deployment model, the Fed Service is the OIDC client. This means the Fed Service is storing the Access and ID Tokens for all Members using this Fed Service.

Mitigation:
These tokens are stored in secure local storage, and never in binary or source code. Standard web server protection should be applied. Token scope should be limited to only the necessary authorizations. Token lifetime should be limited. If a token is compromised, the OIDC Server should immediately revoke the token to prevent any further abuse.

Exhaustion of resource attacks

Denial of service, Mitigated, Medium Severity

Description:
Similar to [11] 4.4.1.11, a malicious actor (member or not) could send excessive requests to a Fed Service.

Mitigation:
Fed Service should rate limit the number of requests per requestor, member or not. Distributed Denial of Service attacks are more problematic.

Disclosure of client information during transmission

Information disclosure, Mitigated, High Severity

Description:
Similar to [11] 4.3.3, an attacker could eavesdrop on communications.

Mitigation:
Data in transmission should be protected using transport-layer mechanisms such as TLS.

Fed Service A (Process)

Description:

[11] 4.1.4, End-user credentials phished

Spoofing, Mitigated, High Severity

Description:
Credentials, such as passwords, could be phished using a compromised or embedded browser.

Mitigation:
OAuth (and OIDC) are designed such that the Fed Service should never see Member passwords. Members should be educated to recognize phishing attacks.

Repudiation of Authentication

Repudiation, Mitigated, High Severity

Description:
When a Member authenticates to the Fed Service, the Access and ID tokens for the Member are subsequently delivered directly from the OIDC server to the Fed Service. These tokens must be protected and the Fed Service must not be compromised.

Mitigation:
The signed Access and ID tokens that the OIDC delivers to the Fed Service provide authentication with integrity and non-repudiation. These tokens are stored in a separate store that is not accessible to external actors. To mitigate repudiation of authentication, all other methods to compromise a Fed Service must be mitigated.

[11] 4.1.1, 4.1.2, 4.1.3, Obtain client secrets.

Information disclosure, Mitigated, High Severity

Description:
In this deployment model, the Fed Service is the OIDC client. This means the Fed Service is storing the Access and ID Tokens for all Members using this Fed Service.

Mitigation:
These tokens are stored in secure local storage, and never in binary or source code. Standard web server protection should be applied. Token scope should be limited to only the necessary authorizations. Token lifetime should be limited. If a token is compromised, the OIDC Server should immediately revoke the token to prevent any further abuse.

Exhaustion of resource attacks

Denial of service, Mitigated, Medium Severity

Description:
Similar to [11] 4.4.1.11, a malicious actor (member or not) could send excessive requests to a Fed Service.

Mitigation:
Fed Service should rate limit the number of requests per requestor, member or not. Distributed Denial of Service attacks are more problematic.

Disclosure of client information during transmission

Information disclosure, Mitigated, High Severity

Description:
Similar to [11] 4.3.3, an attacker could eavesdrop on communications.

Mitigation:
Data in transmission should be protected using transport-layer mechanisms such as TLS.

FS A to B Req (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:
Man-in-the-Middle attacks could occur. DNS or ARP spoofing attacks could occur.

Mitigation:

Fed Service B should validate the binding of Fed Service A to its domain name.
Transport-layer security mechanisms, such as TLS, should be used.

Information disclosure threat

Information disclosure, Mitigated, Medium Severity

Description:
An attacker could eavesdrop on communication between the Fed Services.

Mitigation:
Use transport-layer security mechanisms such as TLS.

FS A to B Resp (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:
Man-in-the-Middle attacks could occur. DNS or ARP spoofing attacks could occur.

Mitigation:
The Fed Service B should validate the binding of Fed Service A to its domain name.
Transport-layer security mechanisms, such as TLS, should be used.

Information disclosure threat

Information disclosure, Mitigated, Medium Severity

Description:
An attacker could eavesdrop on communication between the Fed Services.

Mitigation:
Use transport-layer security mechanisms such as TLS.

FS B to A Resp (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:
Man-in-the-Middle attacks could occur. DNS or ARP spoofing attacks could occur.

Mitigation:
Fed Service A should validate the binding of Fed Service B to its domain name. Transport-layer security mechanisms, such as TLS, should be used.

Information disclosure threat

Information disclosure, Mitigated, Medium Severity

Description:
An attacker could eavesdrop on communication between the Fed Services.

Mitigation:
Use transport-layer security mechanisms such as TLS.

FS B to A Req (Data Flow)

Description:

Man-in-the-Middle

Tampering, Mitigated, High Severity

Description:
Man-in-the-Middle attacks could occur. DNS or ARP spoofing attacks could occur.

Mitigation:
Fed Service B should validate the binding of Fed Service A to its domain name. Transport-layer security mechanisms, such as TLS, should be used.

Information disclosure threat

Information disclosure, Mitigated, Medium Severity

Description:
An attacker could eavesdrop on communication between the Fed Services.

Mitigation:
Use transport-layer security mechanisms such as TLS.

Fed State B: Members, Attrs, Svc Catalog (out of scope Data Store)

Description:
Members, Attributes, Service Catalog

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

Fed State A: Members, Attrs, Svc Catalog (out of scope Data Store)

Description:
Members, Attributes, Service Catalog

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

OIDC B (out of scope Process)

Description:

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

Fed Operator A (out of scope External Actor)

Description:

Out of scope reason:
Not involved in FS-FS interactions.

OIDC B Req (out of scope Data Flow)

Description:

Out of scope reason:
Considered security behind trust boundary.

OIDC A Req (out of scope Data Flow)

Description:

Out of scope reason:
Considered security behind trust boundary.

Fed Op A Req (out of scope Data Flow)

Description:

Out of scope reason:
Not involved in FS-FS interactions.

Fed State B Query (out of scope Data Flow)

Description:

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

Fed State B Resp (out of scope Data Flow)

Description:

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

Fed Operator B (out of scope External Actor)

Description:

Out of scope reason:
Not involved in FedAdmin interactions.

Fed State A Query (out of scope Data Flow)

Description:

Out of scope reason:
Not Involved in FS-FS interactions.

FedOp Req (out of scope Data Flow)

Description:

Out of scope reason:
Not involved in FedAdmin interactions.

FedOp Resp (out of scope Data Flow)

Description:

Out of scope reason:
Not involved in FedAdmin interactions.

Fed State A Resp (out of scope Data Flow)

Description:

Out of scope reason:
Not involved in FS-FS interactions.

OIDC A (out of scope Process)

Description:
This is an arbitrary service being made available to the federation members.

Out of scope reason:
Not involved in FS-FS interactions.

OIDC A Resp (out of scope Data Flow)

Description:

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

OIDC B Resp (out of scope Data Flow)

Description:

Out of scope reason:
Considered secure behind the Fed Service Trust Boundary.

Fed Op A Resp (out of scope Data Flow)

Description:

Out of scope reason:

Not involved.

Appendix B: Revision History

Table 5. Revision History

Date	**Editor**	**Release**	**Primary clauses modified**	**Descriptions**
April 10, 2020	C. Lee	0.1	all	initial structure defined
June 1, 2020	C. Lee	0.2	Sec. 8	Authn threat model and analysis
November 14, 2020	C. Lee	1.0	all	Final version 1.0

Appendix C: Bibliography

bibliography::[]