Securing Digital Identities in the Cloud by Selecting an Apposite Federated Identity Management from SAML, OAuth and OpenID Connect

Nitin Naik and Paul Jenkins
Defence School of Communications and Information Systems
Ministry of Defence, United Kingdom
Email: nitin.naik100@mod.gov.uk and paul.jenkins683@mod.gov.uk

***Abstract*—Access to computer systems and the information held on them, be it commercially or personally sensitive, is naturally, strictly controlled by both legal and technical security measures. One such method is digital identity, which is used to authenticate and authorize users to provide access to IT infrastructure to perform official, financial or sensitive operations within organisations. However, transmitting and sharing this sensitive information with other organisations over insecure channels always poses a significant security and privacy risk. An example of an effective solution to this problem is the Federated Identity Management (FIdM) standard adopted in the cloud environment. The FIdM standard is used to authenticate and authorize users across multiple organisations to obtain access to their networks and resources without transmitting sensitive information to other organisations. Using the same authentication and authorization details among multiple organisations in one federated group, it protects the identities and credentials of users in the group. This protection is a balance, mitigating security risk whilst maintaining a positive experience for users. Three of the most popular FIdM standards are Security Assertion Markup Language (SAML), Open Authentication (OAuth), and OpenID Connect (OIDC). This paper presents an assessment of these standards considering their architectural design, working, security strength and security vulnerability, to cognise and ascertain effective usages to protect digital identities and credentials. Firstly, it explains the architectural design and working of these standards. Secondly, it proposes several assessment criteria and compares functionalities of these standards based on the proposed criteria. Finally, it presents a comprehensive analysis of their security vulnerabilities to aid in selecting an apposite FIdM. This analysis of security vulnerabilities is of great significance because their improper or erroneous deployment may be exploited for attacks.**

***Index Terms*—Federated Identity Management; FIdM; SAML; OAuth; OpenID Connect; SSO; DoS; MITM; XSS**

I. Introduction

In cyberspace, digital identities are used to represent an individual, organization or electronic device, which controls access to critical corporate information by the authentication and authorization of their users providing access to organisational resources. Businesses are required to exchange information both financial and personnel with government agencies and other businesses electronically. This collaborative working and sharing of sensitive information is strictly controlled and protected by legislation in the countries in which the organisation operates. However, the transmission of this sensitive data over insecure channels poses a significant security and privacy risk. This risk can be mitigated by using the Federated Identity Management (FIdM) standard adopted in the cloud environment. Federated identity links and employs users' digital identities across several identity management systems [1], [2]. FIdM defines a unified set of policies and procedures allowing identity management information to be transportable from one security domain to another [3], [4]. Thus, a user accessing data/resources on one secure system could then access data/resources from another secure system without both systems needing individual identities for the single user. In this way, it avoids the transmission of sensitive information/credentials. For example, it is probable that users could possess several accounts with the service providers such as Google, Amazon, eBay and AOL. These service providers require the users' identity to be confirmed by a trusted central policy framing authority in terms of scope and visibility [5], [6], [7], [8]. This relieves the user of the burden of dealing with multiple credentials thereby improving usability and security [2], [7], [8]. The FIdM approach separates the authentication and authorization functions for the better management of both.

There are a number of FIdM standards available, some of the most popular and successful FIdM standards, are Security Assertion Markup Language (SAML), Open Authentication (OAuth), and OpenID Connect (OIDC). SAML is an XML-oriented framework for transmitting user authentication, entitlement and other attribute information [9]. OAuth is a scalable delegation protocol allowing a user to permit access to an application to accomplish authorized tasks on behalf of the user [10]. OpenID Connect is an emerging suite of lightweight specifications that provide a framework for communicating identity via RESTful APIs [11]. These three FIdM standards virtually cover the entire FIdM cloud industry.

This paper presents an assessment of these standards considering their architectural design, working, security strength and security vulnerability, to understand and ascertain effective usages to protect digital identities and credentials. Firstly, it explains the architectural design and working of these standards. Secondly, it proposes several assessment criteria and compares functionalities of these standards based on the proposed criteria. FIdM standards offer the solution to protect digital identities and personal information; however,

their implementation requires thoughtful administration and carefully enforced security and privacy policies. The improper or erroneous deployment of the FIdM standard could have serious consequences and open several security vulnerabilities, which can be easily exploited for attacks. Therefore, it is essential to understand various message flows and their associated security vulnerabilities, which is comprehensively covered in the final section to aid in selecting an apposite FIdM.

The rest of the paper is organised as follows: Section II presents the detailed architectural design and working analysis of the three FIdM standards SAML, OAuth and OIDC; Section III presents the comparative analysis of the three FIdM standards SAML, OAuth and OIDC based on the proposed evaluation criteria; Section IV elucidates potential vulnerabilities of FIdM standards due to their improper or erroneous deployment; Section V concludes the paper and suggests some future work. At the end of this paper, a list of acronyms and their full forms are presented to simplify the discipline specific terminologies.

II. Architectural Design and Working of Predominant Federated Identity Management (FIdM) Standards

This section explains the three predominant FIdM standards SAML, OAuth and OpenID Connect and their working in details. All these standards have a commonality, and they use security tokens for their services. Security Tokens are a key concept in FIdM as they are the device of choice for authenticating and authorizing a users identity or "digital identity". They are also known as Identity Tokens, Authentication Tokens and Authorization Tokens [12].

A. Security Assertion Markup Language (SAML)

Security Assertion Markup Language (SAML) was developed by the Security Services Technical Committee of OASIS (Organization for the Advancement of Structured Information Standards) [9]. SAML is an XML-oriented framework for transmitting user authentication, entitlement, and other attribute information [9]. This framework provides two federation partners to select and share identity attributes using a SAML assertion/message payload, on the condition that these attributes can be expressed in XML [11]. SAML assumes three key roles in any transaction Identity Provider (IDP/IdP), Service Provider (SP) and User:

- **Identity Provider (IDP/IdP)** is a trusted organisation that authenticates and authorizes users. It issues security assertion tokens for authentication and authorization services.
- **Service Provider (SP)** is an organisation that provides Web and other services. A SP relies on a trusted IDP for authentication and authorization services. It acts on information encoded in assertion tokens to determine whether a user is to be allowed access to a resource or not.

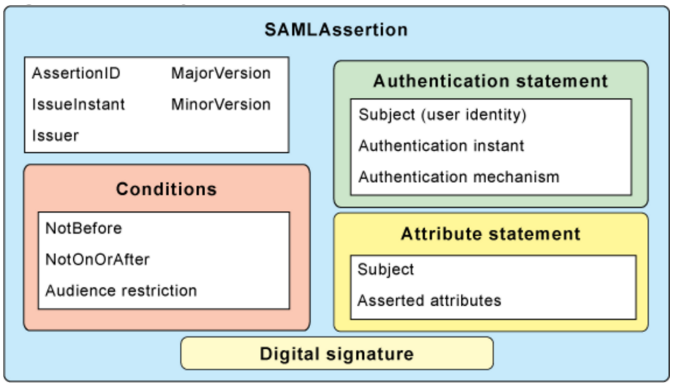

Fig. 1. SAML Assertion Structure [13]

- **User** is an entity that initiates a sequence of protocol messages and consumes the service provided by the SP. A user may be an application program that is requesting access to a resource.

The latest version of the SAML specifications is SAML 2.0, which describes the following components [13]:

- **Assertions** state how identities are represented.
- **Protocols** represent a sequence of XML messages designed to achieve a single goal.
- **Bindings** describe how protocol messages are transported over a lower-level protocol such as HTTP.
- **Profiles** combine a number of bindings to describe a solution for a use case.

The SAML assertion is the main notion in SAML. It is a claim, statement, or declaration of a digital identity which is made by the IDP and trusted by the SP. The identity information required by the SP, is usually agreed in advance by the IDP and SP [14]. However, there is a provision after the initial transaction to request additional information. The structure of a SAML assertion is shown in Fig. 1. There are three types of assertions: authentication, attribute, and authorization. Authentication assertion validates the user's identity. Attribute assertion contains specific information about the user. Authorization assertion identifies what the user is authorized to do [3].

A typical SAML use case example is illustrated in Fig. 2 and its corresponding steps are described below:

1) User tries to access a hosted application on the SP's cloud
2) SP generates a SAML request
3) Browser redirects the SAML request to the IDP's cloud
4) IDP authenticates User, generates and returns a SAML response to Browser
5) Browser sends the SAML response to SP
6) SP verifies the SAML response and User logs in

B. Open Authorization (OAuth)

OAuth is a scalable delegation protocol (i.e., you delegate someone to do something with somebody on your behalf). OAuth allows a user to permit access to an application to

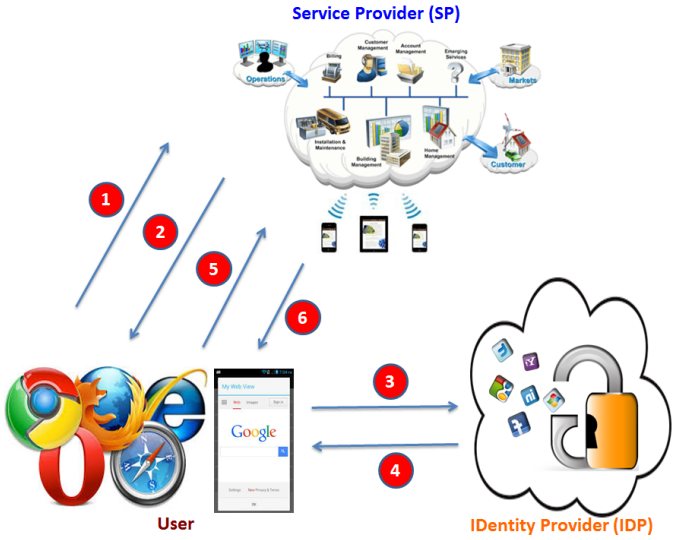

Fig. 2. A typical SAML use case example

accomplish authorized tasks on behalf of the user [10]. Therefore, it allows a third-party program to gain restricted access to an HTTP service. This API authorization process can be securely implemented by a range of desktop, web and mobile applications. It introduces the concept of an authorization token that states the right of the client application to access authorized services on the server [3]. Access to authorised services on the server is controlled through the use of an authorization token. Nonetheless, it does not override any access control decisions that the server-side program may make [14]. The OAuth 2.0 core authorization framework is described by IETF in RFC-6749 alongside with several other specifications and profiles; a few commonly used specifications and profiles are shown in Fig. 3 and described below [15], [16], [17], [18], [19], [20], [21]:

- **OAuth 2.0 Core Spec** describes the generic flows of OAuth operation. It is essentially descriptions of the interactions between a client application, a resource owner and an authorization server to request access tokens [17].
- **OAuth 2.0 Bearer Spec** describes how to use bearer tokens in HTTP requests to access OAuth 2.0 protected resources [18]. The bearer token is a large random number and a symbol of authorization. Since the number is large, then the probability of guessing the correct number is very small. This token is easier to process and use than the signature but requires SSL. Bearer tokens are the default type of access tokens.
- **OAuth 2.0 MAC Spec** describes the HTTP MAC access authentication scheme, an HTTP authentication method using a MAC algorithm to provide cryptographic verification of portions of HTTP requests [19]. It is similar to the OAuth $1.0a$ token and uses a signature instead of SSL. This token securely authenticates users without encrypting all traffic. Therefore, it is the most suitable option for APIs that require the security of OAuth and handle very large requests or responses where SSL is inefficient.
- **OAuth 2.0 JWT Spec** describes the use of a JWT Bearer Token as a means for requesting an OAuth 2.0 access token as well as for client authentication [20]. JWT is a JSON-based security token encoding that enables identity and security information to be shared across security domains.
- **OAuth 2.0 SAML Spec** describes the use of a SAML 2.0 Bearer Token (Assertion) as a means for requesting an OAuth 2.0 access token in addition to use as a means of client authentication. It is available in the OAuth 2.0 [21]. It extends the support to the SAML-based operations. This facility of OAuth made it more popular among the SAML community and the universal open standard.

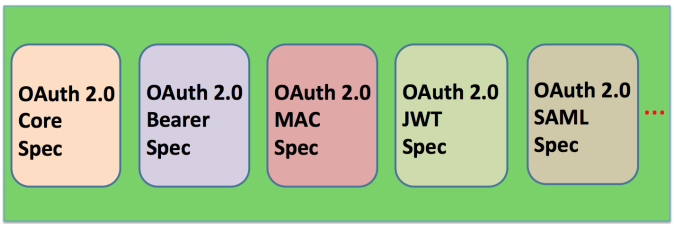

Fig. 3. OAuth 2.0 commonly used specifications and profiles

OAuth assumes four key roles in any authorization process Resource Server (RS), Resource Owner (RO)/User, OAuth Consumer/Client (OC) and Authorization Server (AS) [15]:

- **Resource Server (RS)** hosts user data that is protected by OAuth.
- **Resource Owner (RO)/User** is the user of the application and owner of data.
- **OAuth Consumer/Client (OC)** is the application which makes an API request to get protected resources on behalf of the resource owner.
- **Authorization Server (AS)** authorises the consumer after getting permission from resource owner and issues access token to the consumer for accessing protected resources available on the resource server.

OAuth offers the flexibility and leaves it up to server implementers to decide how the actual authentication and authorisation are to be done [22]. A typical OAuth use case example is illustrated in Fig. 4 and its corresponding steps are described below:

1) RO logs into an application and requires to access resources from a different organisation
2) OC requests for a Request Token and Secret Key
3) AS issues the Request Token and Secret Key
4) OC sends a URL link containing the Request Token to User and asks for an authorization
5) RO has logged into OP's system and clicks on the URL containing the Request Token
6) AS asks User to allow or deny OC
7) RO authorizes to access resources
8) AS generates an Access Token and forwards it to OC

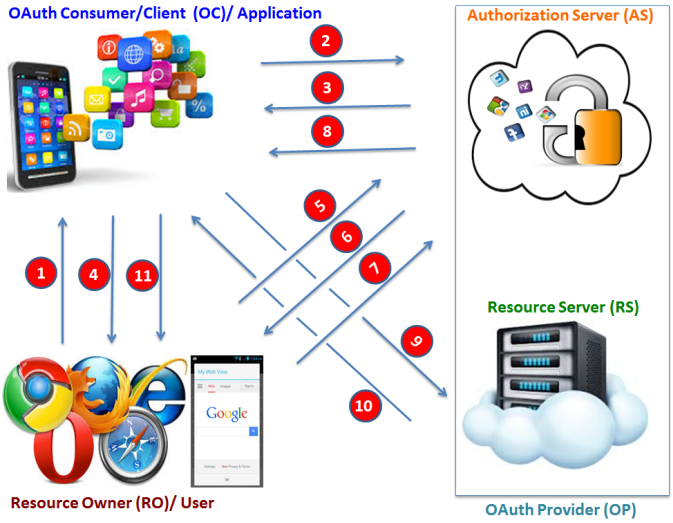

Fig. 4. A typical OAuth use case example

9) OC sends the Access Token and acquires resources for User
10) RS sends resources to OC
11) OC delivers resources to User

C. OpenID Connect (OIDC)

OpenID Connect is a group of lightweight specifications that afford a framework for transmitting digital identity via RESTful APIs [11]. The final OpenID Connect specifications were launched on February 26, 2014 [23]. OpenID Connect is seen as the evolution of OpenID 2.0, and is built as a profile of OAuth 2.0 rather than a completely distinct protocol foundation [11]. OpenID Connect 1.0 is just another identity layer on the top of the OAuth 2.0 protocol [23], as shown in Fig. 5. It facilitates clients to confirm the identity of the user depending on the authentication made by an Authorization Server, in addition to acquire simple profile information about the user [3]. OpenID Connect uses two main types of tokens: an access token and an ID token. The ID contains information about the authenticated user and it is a JWT (JSON Web Token). This token is signed by the identity provider and can be read and verified without accessing the identity provider [24].

OIDC assumes five key roles in any authentication and authorization process End User, Relying Party (RP), Authorization Endpoint (AE), Token Endpoint (TE) and UserInfo Endpoint (UIE) [22], [24], [25], [26]:

- **End User** is the user of the application and owner of the information.
- **Relying Party (RP)** is the application which makes API request to get protected resources on behalf of the end user.
- **Authorization Endpoint (AE)** is the only endpoint where the end-user needs to interact if they are not already logged in. It validates the identity of the end-user and obtains the consent and authorization from the end-user if the client has not been pre-authorized. It returns an authorization grant to the end-user or client depending on the use case. Sometimes, this authorization grant can then be passed in a request by the client to the token endpoint in exchange for an ID token, access token, and refresh token [26].
- **Token Endpoint (TE)** handles requests for retrieving and refreshing access tokens, ID tokens, refresh tokens, and other variables. It accepts a request from the client that includes an authorization code that is issued to the client by the authorization endpoint directly or via end user. When the authorization code is validated, the appropriate tokens are returned in response to the client [22].
- **UserInfo Endpoint (UIE)** is an OAuth 2.0 protected resource that the client application can retrieve consented claims, or assertions, about the authenticated end user. The client should present a valid access token to retrieve only those UserInfo claims that are scoped by the presented token.

OpenID also offers some flexibility in the implementation, however, it standardised many parameters such as instance scopes, endpoint discovery, and dynamic registration of clients, which were left up to implementers in the OAuth 2.0 implementation [22]. A typical OIDC use case example is illustrated in Fig. 6, and its corresponding steps are described below:

1) EU provides their OpenID login details
2) RP discovers OP and forwards login details to OP
3) AE authenticates User after verifying login credentials
4) EU has logged in to the OP's system
5) EU sends an authorization code
6) RP sends the authorization code and secret information
7) TE sends an ID Token and Access Token
8) RP validates the ID Token
9) RP sends the Access Token to UIE
10) UIE sends detailed information containing user's attributes
11) RP delivers services to User

III. Comparison of Federated Identity Management (FIdM) Standards based on the Proposed Evaluation Criteria

This section presents the comparative analysis of the three FIdM standard SAML, OAuth and OIDC based on the proposed evaluation criteria as shown in Table I [1], [3]. They have been critically analysed and compared on the basis of proposed criteria to demonstrate their strengths and limitations. From this critical analysis, it suggests that SAML has some issues in some of the evaluation criteria related to mobile devices and IoT, and requires an overhaul [3]. OAuth is an effective protocol for authorization. Nonetheless, as it is a delegation protocol, consequently, it has not been developed for authentication and offer a complete FIdM solution. OpenID Connect offers slightly better functionality, as it has been developed to deliver services for the web, cloud, mobile

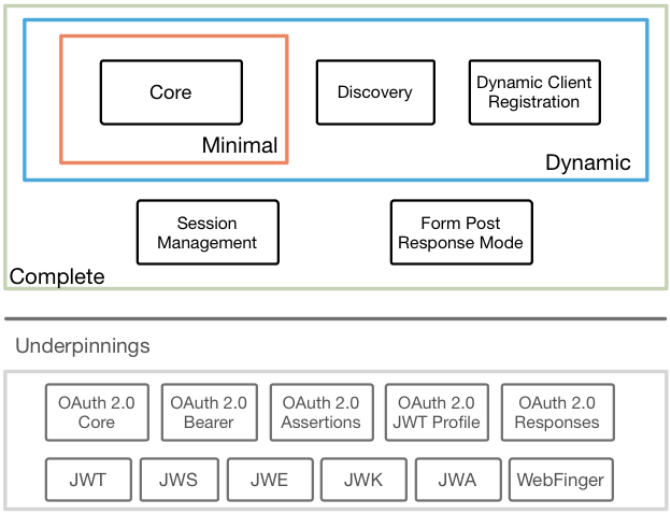

Fig. 5. OIDC Protocol Suite [23]

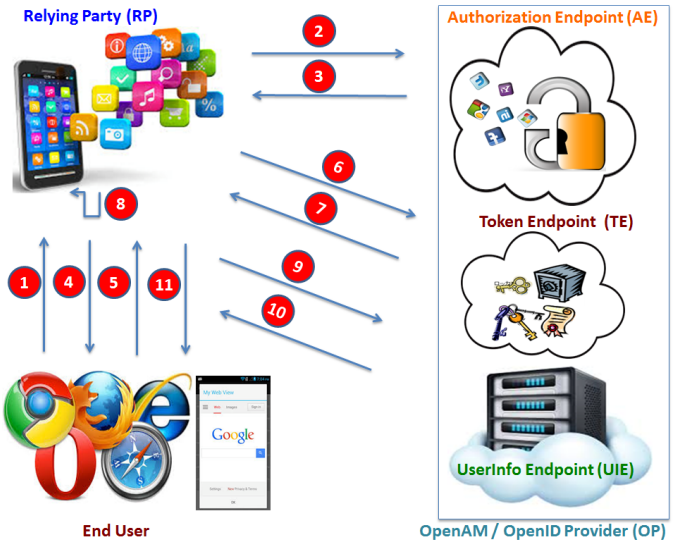

Fig. 6. A typical OIDC use case example

devices and IoT. Nevertheless, it is a developing standard, and the OpenID Connect 1.0 specifications were instigated on February 26, 2014 [23]. Moreover, many prominent companies such as Facebook and Twitter have been using their own version of OpenID Connect, known as Facebook and Twitter Connect based on OAuth 2.0 [3]. Thus, OIDC requires more time and enterprise acceptance to become established standard.

IV. Security Vulnerability Analysis of Federated Identity Management (FIdM) Standards

This security vulnerability analysis focuses on SAML and OIDC because OAuth is already an implicit standard in OIDC. SAML and OIDC both describe the security and privacy considerations for using them. They are powerful SSO frameworks but their method of deployment and implementation may leave some vulnerability which could lead to potential attacks. Here some of the working procedures of SAML and

```
<samlp:AuthnRequest
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
ID="identifier_1"
Version="2.0"
IssueInstant="2016-12-05T09:21:59Z"
AssertionConsumerServiceIndex="1">
<saml:Issuer>https:// saml-sp.com/SAML2</saml:Issuer>
<samlp:NameIDPolicy
AllowCreate="true"
Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/>
</samlp:AuthnRequest>
```

Fig. 7. SP-to-IdP Authentication Request in SP-Initiated SSO: Redirect/POST Bindings

OIDC are discussed which can be referred throughout the vulnerability analysis to demonstrate how their improper or erroneous deployment may be exploited for attack.

A. Denial-of-Service (DoS) Attack

1) DoS Attack in SAML: To assimilate the possibility of DoS attack in SAML, it is necessary to understand the SAML message flow. SAML supports two general message flows, namely SP-initiated and IdP-initiated. For the web SSO profile, two common SAML messages are: an *Authentication Request* message sent from an SP to an IdP, and a *Response* message, containing a SAML assertion, sent from the IdP to the SP. The SAML Conformance and Profiles specifications ascertain the SAML bindings, which can validly be used with the above two messages. In particular, an Authentication Request message can be sent from an SP to an IdP via the HTTP Redirect Binding, HTTP POST Binding, or HTTP Artefact Binding [27], [28], [29]. Moreover, the Response message can be sent from an IdP to a SP via the HTTP POST Binding or the HTTP Artefact Binding [27], [28], [29]. Furthermore, SAML permits asymmetry in the message pair, allowing a different binding on the return message to that of the initiating message. The decision of which binding to use, is made according to the configuration settings at the SP and the IdP sides [30].

One possible scenario of DoS attack in SAML is when the SP-Initiated SSO (Redirect/POST Bindings) message flow is implemented. In this type of SAML message flow, the user tries to access a resource on the SP (e.g., saml-sp.com). Though the user does not hold a valid/current logon session on this site and the corresponding federated identity is governed by their IdP (e.g. saml-idp.com). Thus, the user is sent to the IdP to log on and the IdP delivers a SAML web SSO assertion for the user's federated identity to the SP. This exchange uses a Redirect Binding for the SP-to-IdP *AuthnRequest* message (see Fig. 7) and a POST Binding for the IdP-to-SP *Response* message (see Fig. 8).

In this scenario, an attacker can target the IdP for DoS attack by exploiting any vulnerability related to erroneous deployment or due to the vulnerabilities of other supporting tools [31]. The IdP can potentially be flooded with requests by compromising valid users or a honest SP because the SAML request requires substantial processing overheads (including parsing of request and assertion construction). Consequently,

TABLE I
COMPARATIVE ANALYSIS OF FIDM STANDARDS SAML, OAUTH AND OIDC BASED ON THE PROPOSED EVALUATION CRITERIA

Criteria	SAML	OAuth	OIDC
1. Current Version	SAML 2.0	OAuth 2.0	OpenID Connect 1.0
2. Introduction Year	2005	2012	2014
3. Main Usages	Federated Identity Management (FIdM), Single Sign-On (SSO) for enterprise users.	API authorization between Applications.	Federated Identity Management (FIdM), Single Sign-On (SSO) for consumers.
4. Authentication and Authorization	It is a standard for authentication and authorization.	It is a standard for authorization of resources.	It is a standard for authentication and authorization.
5. Token Format	XML	XML, JSON, JWT	JSON, JWT
6. Token Content	Token contains user identity information but not credentials.	Token contains user identity information but not credentials.	Token contains user identity information but not credentials.
7. Protocol Used	XML, HTTP, SOAP	JSON, HTTP, REST	JSON, HTTP, REST
8. Schemas and Deployments	SPML, SCIM	SCIM	SCIM
9. Roles/Actors	Identity Provider (IDP), Service Provider (SP) and User.	Resource Server (RS), Resource Owner (RO)/User, OAuth Consumer/Client (OC) and Authorization Server (AS).	End User (EU), Relying Party (RP), Authorization Endpoint (AE), Token Endpoint (TE) and UserInfo Endpoint (UIE).
10. Transaction Initiation	SP and IDP initiation.	Consumer/Client (OC) initiation.	Relying Party (RP)/ End User initiation.
11. User Consent	It is not responsible for collecting users consent. However, ECP allows for the exchange of SAML attributes outside the context of a web browser.	It collects users consent before sharing attributes.	It collects users consent before sharing attributes.
12. Claims	No distributed and aggregated claims.	No distributed and aggregated claims.	Distributed and aggregated claims.
13. Client Discovery and On-Boarding	No dynamic introductions.	No dynamic introductions.	Dynamic introductions.
14. Immediate Revocation of Access	It supports revocation. However, in some cases, if you remove a user from your identity provider, you must also manually suspend them. Otherwise, they will continue to be able to authenticate using access tokens or SSH keys.	It supports revocation. Token revocation is used to revoke a specified OAuth 2.0 access or refresh token. A revoke token request causes the removal of the client permissions associated with the specified token used to access the user's protected resources.	It supports revocation. Similar to OAuth. However, OIDC has additional ID token that is a cryptographically signed, self-contained token. It allows resource owners to authorize access without a call to the authorization server and it cannot be explicitly revoked.

TABLE I
COMPARATIVE ANALYSIS OF FIDM STANDARDS SAML, OAUTH AND OIDC BASED ON THE PROPOSED EVALUATION CRITERIA

Criteria	SAML	OAuth	OIDC
15. Data Integrity/ Non-repudiation	XML Signature - X.509; SAML tokens are almost always signed with a private key, as it is a trusted relationship between IDP and SP.	Default bearer token has no proof of possession. However, token contents can be protected by using a DS or a MAC.	JSON Web Signature (JWS)-HMAC SHA-256; [Additional Support -RSA SHA-256 and ECDSA P-256 SHA-256].
16. Data Confidentiality/ Privacy	XML Encryption-Triple-DES-CBC with 192-bit key and a 64-bit initialization Vector (IV), AES-CBC with a 128-bit initialization vector (IV); [TLS-SSL, Web Services Security (WSS)].	TLS is mandatory to implement with OAuth for token confidentiality. However, token encryption must be applied in addition to the usage of TLS protection.	JSON Web Encryption (JWE)-RSA-PKCS1-1.5 with 2048-bit key, AES-128-CBC, and AES-256-CBC; [Additional Support- ECDH-ES with 256-bit key, AES-128-GCM, and AES-256-GCM].
17. Web and Native Mobile Apps Support	It is specially designed for Web apps. However, HTTP artefact binding can be used to reduce the flow of SAML messages through the browser.	It supports both Web and native mobile apps.	It supports both Web and native mobile apps.
18. Consumer and Enterprise Support	It mainly supports enterprise users because it involves SP and IdP.	It supports enterprise users, and consumer apps and services.	It supports enterprise users, and consumer apps and services.
19. Lightweight Standard/Protocol	It is not a lightweight standard. XML states trees in a verbose form. Every element in the tree has a name (the element type name), and the element must be enclosed in a matching pair of tags.	It is a lightweight standard. JSON states trees in a nested array type of notation similar to that of Javascript. Indeed, a JSON document can exactly be parsed as Javascript to result in the corresponding array.	It is a lightweight standard. Similar to OAuth. JSON has a much smaller grammar and maps.
20. Platform Independent Vendor-Neutral and Open Standard	It is a platform independent, vendor-neutral and open standard. However, flexibility in the implementation leads to the different design models.	It is a platform independent, vendor-neutral and open standard. However, flexibility in the implementation leads to the different design models.	It is a platform independent, vendor-neutral and open standard. It also standardised many parameters such as instance scopes, endpoint discovery, and dynamic registration of clients, which were left up to implementers in the OAuth 2.0 implementation.
21. Scalable Standard	It requires the implementation of a complex broker service in order to support multi-SP and multi-IDP use cases.	It greatly reduces the work required to act as a client of a service, which is very vital for mobile community. Also ScalableOAuth extension can be used. Also ScalableOAuth extension can be used.	It is highly scalable, as it has been designed from the inception to provide services for the web, cloud, mobile devices and things.
22. Mobile Standard	It is limited in its ability to support mobile and smart-TV devices.	It has been designed for the mobile API and therefore it is also known as a token in your mobile.	It has been working towards standardising a GSMA Mobile Connect standard for mobile devices.
23. Service Examples	Google, Salaseforce, Amazon, OneLogin, Shibboleth, AOL, Go Daddy	Facebook, Twitter, Linkedin, Google, Salaseforce, Yahoo, AOL, Orange, Deutsche, Telekom	Google, Salaseforce, Yahoo, AOL, Orange, Deutsche, Telekom
24. Vendor Examples	Microsoft, Ping Identity, IBM, Oracle, Centrify, Okta, VeriSign	IBM, Microsoft, NRI, Ping Identity, Layer 7, ForgeRock, Gluu, MITRE	IBM, Microsoft, NRI, Ping Identity, Layer 7, ForgeRock, Gluu, MITRE

```
<samlp:Response
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
ID="identifier_2"
InResponseTo="identifier_1"
Version="2.0"
IssueInstant="2016-12-05T09:22:05Z"
Destination="https://saml-sp.com/SAML2/SSO/POST">
<saml:Issuer>https://saml-idp.com/SAML2</saml:Issuer>
<samlp:Status>
<samlp:StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
</samlp:Status>
<saml:Assertion
.........
.........
</saml:Assertion>
</samlp:Response>
```

Fig. 8. IdP-to-SP Response in SP-Initiated SSO-Redirect/POST Bindings

```
GET /.well-known/openid-configuration HTTP/1.1
  Host: openid-provider.com
```

Fig. 9. OIDC Discovery Configuration Request

```
HTTP/1.1 200 OK
  Content-Type: application/json

  {
   "issuer":
     "https:// server.openid-provider.com ",
   "authorization_endpoint":
     "https:// server.openid-provider.com/connect/authorize",
   "token_endpoint":
     "https:// server.openid-provider.com/connect/token",
   "token_endpoint_auth_methods_supported":
     ["client_secret_basic", "private_key_jwt"],
   "token_endpoint_auth_signing_alg_values_supported":
     ["RS256", "ES256"],
   "userinfo_endpoint":
     "https:// server.openid-provider.com/userinfo",
   .........
   .........
   }
```

Fig. 10. OIDC Discovery Configuration Response

```
<form method="post" action="https://saml-idp.com/SAML2/SSO/POST" ...>
<input type="hidden" name="SAMLRequest" value="request" />
<input type="hidden" name="RelayState" value="token" />
...
<input type="submit" value="Submit" />
</form>
```

Fig. 11. A SAML *AuthnRequest* message encoded as the value of a hidden form control named *SAMLRequest* in SP-Initiated SSO: POST/Artefact Bindings

the effort required for processing of each Response assertion (see Fig. 8) is proportionally much greater than the effort required by an attacker to generate the request [32]. This is confirmed by the draft on Security and Privacy Considerations for SAML document [33] that SAML is vulnerable to DoS attacks.

2) DoS Attack in OIDC: To assimilate the possibility of DoS attack in OIDC, it is necessary to understand the *discovery* process for obtaining OIDC identity provider's configuration information. OIDC identity provider (e.g., openid-provider.com) supports metadata discovery and therefore, it hosts its configuration information at the endpoint (/.well-known/opened-configuration). In most of the implementation, endpoint is accessible by any client/relying party who is wishing to send registration request and thus, it is publicly open and possibly non-secure. Subsequently, OIDC client/relying party sends an HTTP GET request (see Fig. 9) to this metadata endpoint to obtain the configuration information of OIDC identity provider.

In response to this request for the configuration information, the OIDC identity provider (openid-provider.com) sends a response which is a set of *Claims* about the OIDC provider's configuration, including all necessary endpoints and public key location information as shown in Fig. 10. This information is necessary for client/relying party to further communicate with the OIDC identity provider or the OAuth authorization server.

Assuming this common implementation model when the endpoint is publicly open and non-secure, and dynamic discovery process is allowed without any authentication. If not properly implemented, this vulnerability can be easily exploited for DoS attack on an OIDC identity provider and flooded by countless dynamic discovery requests, which could easily overwhelm the OIDC identity provider [34]. Furthermore, this dynamic discovery process may also be exploited for DoS attack on client/relying party. Where, an attacker may try to spoof an OpenID identity provider by publishing a discovery information that contains an issuer *Claims* using the Issuer URL of the OIDC identity provider being impersonated, but with its own endpoints and signing keys. Thus, the client/relying party can be flooded with information by attacker.

B. Man-In-The-Middle (MITM) Attack

1) MITM Attack in SAML: One of the many possibilities of a MITM attack in SAML is when the SP-Initiated SSO (POST/Artefact Bindings) message flow is implemented. This exchange uses a POST Binding for the SP-to-IdP *AuthnRequest* message and a Artefact Binding for the IdP-to-SP *Response* message [30]. In this type of SAML flow, the user tries to access a resource on the SP (e.g., saml-sp.com). Though the user does not hold a valid/current logon session on this site and the corresponding federated identity is governed by their IdP (e.g. saml-idp.com). The SP saves the requested resource URL in local state information, which can be saved across the web SSO exchange and sends an HTML form to the requested browser in the HTTP response (HTTP status 200) [27]. The HTML form contains a SAML *AuthnRequest* message (see Fig. 12) encoded as the value of a hidden form control named *SAMLRequest* as shown in Fig. 11.

The user enters correct credentials and a local logon related security setting is generated for the user at the IdP. Later, the IdP creates an artefact containing the source ID for its website (saml-idp.com) and a reference to the *Response* message (the

```
<samlp:AuthnRequest
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
ID="identifier_1"
Version="2.0"
IssueInstant="2016-12-05T09:22:05Z"
AssertionConsumerServiceIndex="1">
<saml:Issuer>https://saml-sp.com/SAML2</saml:Issuer>
<samlp:NameIDPolicy
AllowCreate="true"
Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/>
</samlp:AuthnRequest>
```

Fig. 12. SP-to-IdP-Authentication Request in SP-Initiated SSO: POST/Artefact Bindings

```
<samlp:ArtifactResolve
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
ID="identifier_2"
Version="2.0"
IssueInstant="2016-12-05T09:22:05Z"
Destination="https://saml-idp.com/SAML2/ArtifactResolution">
<saml:Issuer>https://saml-sp.com/SAML2</saml:Issuer>
<!-- an ArtifactResolve message SHOULD be signed -->
<ds:Signature
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
<samlp:Artifact>artifact</samlp:Artifact>
</samlp:ArtifactResolve>
```

Fig. 13. SP's Assertion Consumer Service sends a SAML *ArtifactResolve* message to IdP using the synchronous SOAP binding in SP-Initiated SSO: POST/Artefact Bindings

```
<samlp:ArtifactResponse
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
ID="identifier_3"
InResponseTo="identifier_2"
Version="2.0"
IssueInstant="2016-12-05T09:22:05Z">
<!-- an ArtifactResponse message SHOULD be signed -->
<ds:Signature
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
<samlp:Status>
<samlp:StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
</samlp:Status>
<samlp:Response
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
ID="identifier_4"
InResponseTo="identifier_1"
Version="2.0"
IssueInstant="2016-12-05T09:22:05Z"
Destination="https://saml-sp.com/SAML2/SSO/Artifact">
<saml:Issuer>https://saml-idp.com/SAML2</saml:Issuer>
<ds:Signature
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
<samlp:Status>
<samlp:StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
</samlp:Status>
<saml:Assertion
.....
.....
</saml:Assertion>
</samlp:Response>
</samlp:ArtifactResponse>
```

Fig. 14. IdP's Artefact Resolution Service sends back a SAML *ArtifactResponse* message to SP using the synchronous SOAP binding in SP-Initiated SSO: POST/Artefact Bindings

MessageHandle). The HTTP Artefact binding permits the choice of either HTTP redirection or an HTML form POST as a way to deliver the artefact to the SP [27].

The SP's Assertion Consumer Service sends a SAML *ArtifactResolve* message (see Fig. 13), which contains the artefact to the IdP's Artefact Resolution Service endpoint using the synchronous SOAP binding. The IdP's Artefact Resolution Service extracts the MessageHandle from the artefact and finds the original SAML *Response* message accompanying with it [27]. The retrieved message is placed in a SAML *ArtifactResponse* message (see Fig. 14) that is returned to the SP using the synchronous SOAP binding. The SP extracts and processes the *Response* message and processes the embedded assertion for creating a local logon security setting for the user at the SP [27].

The above explained SAML SP-Initiated SSO (POST/Artefact Bindings) process utilises the SOAP binding which is the weak link and vulnerable to the MITM attack [35]. The *RelayState* token is not a transparent reference to state information which is maintained at the SP. This *RelayState* mechanism can leak information about the user's activities at the SP to the IdP if the SP deployment is erroneous or some other kind of existing vulnerabilities which may also lead to the MITM attack [31]. Since the HTTP Artefact binding will be used to deliver the SAML *Response* message, it is not compulsory that this assertion be digitally signed which is also a great security risk and increases the chances of the MITM attack in SAML.

2) MITM Attack in OIDC: One of the many possibilities of a MITM attack in OIDC is in the process of OIDC *dynamic client registration*. After obtaining the OIDC configuration information, an OIDC client/relying party has to register with the OpenID provider in order to utilise OIDC services for an End-User. For registering a new OIDC client/relying party at the Authorization Server, the client/relying party (e.g., openid-app.com) sends an HTTP POST message including its metadata to the Client Registration Endpoint (OAuth 2.0 Protected Resource) with a content type of application/JSON, and the parameters represented as top-level elements of the root JSON object as shown in Fig. 15. The subsequent response may carry a Registration Access Token which can be used by the client/relying party to accomplish required tasks upon the resulting client/relying party registration. This response should use the HTTP 201 Created status code and return a JSON document [RFC4627] using the application/JSON content type with the corresponding fields and the client/relying party Metadata parameters as top-level elements of the root JSON object as shown in Fig. 16.

The OIDC identity provider may require an Initial Access Token to limit registration requests to only authorized clients or developers [34]. However, to support an open dynamic registration, the Client Registration Endpoint should accept registration requests without OAuth 2.0 Access Tokens. Therefore, the dynamic client registration could be the potential source of many attacks including the MITM attack. This MITM attack may be caused by a logical flaw in the OAuth 2.0 protocol or the presence of a malicious OIDC identity provider or malicious client/relying party [36], [37].

```
POST /connect/register HTTP/1.1
  Content-Type: application/json
  Accept: application/json
  Host: server.openid-provider.com
  Authorization: Bearer eyJhbGciOiJSUzI1NiJ9.eyJ ...

  {
   "application_type": "web",
   "redirect_uris":
     ["https://client.openid-app.com/callback",
      "https://client.openid-app.com/callback2"],
   "client_name": "My App",
  ......
  ......
  }
```

Fig. 15. OIDC Dynamic Client Registration Request

```
HTTP/1.1 201 Created
  Content-Type: application/json
  Cache-Control: no-store
  Pragma: no-cache

  {
   "client_id": "s6BhdRkqt3",
   "client_secret":
     "ZJYCqe3GGRvdrudKyZS0XhGv_Z45DuKhCUk0gBR1vZk",
   "client_secret_expires_at": 1577858400,
   "registration_access_token":
     "this.is.an.access.token.value.ffx83",
   "registration_client_uri":
     "https://server.openid-provider.com/connect/register?client_id=s6BhdRkqt3",
   "token_endpoint_auth_method":
     "client_secret_basic",
   "application_type": "web",
   "redirect_uris":
     ["https:// client.openid-app.com/callback",
      "https:// client.openid-app.com/callback2"],
   "client_name": "My App",
  ......
  ......
  }
```

Fig. 16. OIDC Dynamic Client Registration Response

A malicious OIDC identity provider can trick the client/relying party into sending an authorization code to the attacker's Token Endpoint. Once a code is stolen, an attack that involves cutting and pasting values and state in authorization requests and responses can be used to confuse the relying party into binding an authorization to the wrong user [38].

The MITM attacker can confuse a relying party in relation to the selection of an appropriate IdP at the start of the login or authorization procedure for obtaining an authentication code or access token that can be utilised to impersonate the user or for acquiring user data. [36], [37]. It permits a hacker to modify user data and fool the relying party into treating it as the IdP the user wants [36], [37]. As a consequence, the relying party sends the authorisation code or the access token issued by the honest IdP to the attacker depending on the OAuth mode employed. Eventually, an attacker can utilise this information for login at the client/relying party under the user's identity (managed by the honest IdP) or accessing the user's protected resources at the honest IdP [36], [37].

C. Cross-Site Scripting (XSS) Attack

1) XSS Attack in SAML: In ordinary XSS attacks, the attacker uses social engineering methods to trap a user by clicking on a malicious link, whereas XSS attacks in SAML, an exploitation of the vulnerability of the erroneous deployment of SAML framework makes it easy for systematically trapping a user by visiting URIs that may be vulnerable to XSS attacks [39]. This is a more serious XSS attack because here, the client is not suspicious in receiving an altered resource. Furthermore, a *Response* used in SAML process could possibly contain unencoded data delivered by a source which is not a trustworthy source. Therefore, an attacker could use this to initiate an XSS attack by diverting to a maliciously-crafted URL. In addition to the issue with SAML *Response*, a plain deployment of SAML exposes the *RelayState* field (see Fig. 11) to a probable injection of malicious code which may be executed at the honest SP side.

2) XSS Attack in OIDC: Some of the popular OIDC identity providers support an automatic authorization granting feature, which creates an authorization response automatically if a user has an existing session with the OIDC identity provider and previously granted permission for the same client/relying party [40]. Using this automatic authorization granting feature, an attacker may be able to steal a user access token by exploiting an XSS vulnerability in the client/relying party. Currently, this vulnerability revealed in Android's built-in browser has been exploited for this XSS attack. Where, an attacker utilises a browser *window.open* event for sending a counterfeit authorization request to OIDC authorization server, in which *response type=code* is altered to *response type=code token id token*.

V. Conclusion

This paper presented an assessment of the three popular FIdM standards Security Assertion Markup Language (SAML), Open Authentication (OAuth), and OpenID Connect (OIDC) considering their architectural design, working, security strength and security vulnerability, to cognise and ascertain effective usages to protect digital identities and credentials. Firstly, it explained the architectural design and working of these FIdM standards. Secondly, it proposed several assessment criteria and compared functionalities of these FIdM standards based on the proposed criteria. Finally, it presented a comprehensive analysis of their security vulnerabilities to aid in selecting an apposite FIdM. This analysis of security vulnerabilities is of great significance for their correct implementation because the improper or erroneous deployment of these standards may be exploited for several attacks. This in-depth assessment would be helpful for other FIdM users and researchers to select apposite FIdM based on their characteristics and application areas. In the future, it would be worthwhile to perform a further investigation of security vulnerabilities of SAML, OAuth and OIDC.

List of Acronyms

AE	Authorization Endpoint
AES	Advanced Encryption Standard
API	Application Program Interface
AS	Authorization Server
CBC	Cipher Block Chaining

CSRF	Cross-Site Request Forgery
DES	Data Encryption Standard
DoS	Denial-of-Service
DS	Digital Signature
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ES	Ephemeral Static
EU	End User
FIdM	Federated Identity Management
GSMA	Groupe Speciale Mobile Association
HMAC	Hash-based Message Authentication Code
HTTP	Hyper Text Transfer Protocol
IDM	IDentity Management
IdP/IDP	IDentity Provider
IETF	Internet Engineering Task Force
JSON	JavaScript Object Notation
JWE	JSON Web Encryption
JWS	JSON Web Signature
JWT	JSON Web Token
MAC	Message Authentication Code
MITM	Man-In-The-Middle
OC	OAuth Consumer/Client
OAuth	Open Authorization
OASIS	Organization for the Advancement of Structured Information Standards
OIDC	OpenID Connect
OP	OAuth/OpenID Provider
PKCS	Public-Key Cryptography Standards
REST	REpresentational State Transfer
RFC	Requests For Comments
RO	Resource Owner
RP	Relying Part
RS	Resource Server
RSA	Rivest-Shamir-Adleman
SCIM	Simple Cloud Identity Management
SAML	Security Assertion Markup Language
SP	Service Provider
SPML	Services Provisioning Markup Language
Spec	Specification
SSL	Secure Sockets Layer
SSO	Single Sign-On
TE	Token Endpoint
TLS	Transport Layer Security
UIE	UserInfo Endpoint
URL	Uniform Resource Locator
WSS	Web Services Security
XML	eXtensible Markup Language
XSS	Cross-Site Scripting

REFERENCES

[1] N. Naik and P. Jenkins, "A secure mobile cloud identity: Criteria for effective identity and access management standards," in *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud 2016)*. IEEE, 2016, pp. 89–90.

[2] N. Naik, "Connecting Google cloud system with organizational systems for effortless data analysis by anyone, anytime, anywhere," in *IEEE International Symposium on Systems Engineering (ISSE 2016)*. IEEE, 2016.

[3] N. Naik and P. Jenkins, "An analysis of open standard identity protocols in cloud computing security paradigm," in *14th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2016)*. IEEE, 2016.

[4] C. A. Gunter, D. Liebovitz, and B. Malin, "Experience-based access management: A life-cycle framework for identity and access management systems," *IEEE Security & Privacy*, vol. 9, no. 5, p. 48, 2011.

[5] N. Naik, "Migrating from virtualization to dockerization in the cloud: Simulation and evaluation of distributed systems," in *IEEE 10th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments, (MESOCA 2016)*. IEEE, 2016, pp. 1–8.

[6] A. Gopalakrishnan, "Cloud computing identity management," *SETLabs briefings*, vol. 7, no. 7, pp. 45–55, 2009.

[7] N. Naik, "Building a virtual system of systems using Docker Swarm in multiple clouds," in *IEEE International Symposium on Systems Engineering (ISSE 2016)*. IEEE, 2016.

[8] ——, "Applying computational intelligence for enhancing the dependability of multi-cloud systems using docker swarm," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016.

[9] N. Klingenstein, T. Hardjono, H. Lockhart, and S. Cantor. (2012) OASIS Security Services (SAML) TC. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[10] N. Ranjbar and M. Abdinejadi, "Authentication and Authorization for Mobile Devices," B.Sc. Dissertation, Department of Computer Science and and Engineering Goteborg, Sweden, 2012.

[11] Pingidentity.com. (2011) A standards-based mobile application idm architecture. [Online]. Available: http://www.enterprisemanagement360.com/wp-content/files_mf/white_paper/exp_final_wp_mobile-application-idm-arch-8-11-v4.pdf

[12] T. J. Smedinghoff. (2008) Introduction to online identity management. [Online]. Available: https://www.uncitral.org/pdf/english/colloquia/EC/Smedinghoff_Paper_-_Introduction_to_Identity_Management.pdf

[13] C. Forster and N. Readshaw. (2008, April 29) Using SAML security tokens with microsoft web services enhancements: A standards-based approach enabled by tivoli federated identity managers. [Online]. Available: http://www.ibm.com/developerworks/tivoli/library/t-samlwse/

[14] A. Hindle. (2010, May 8) Authentication vs. Authorization - Part 2: SAML and OAuth. [Online]. Available: http://www.axiomatics.com/blog/entry/authentication-vs-authorization-part-2-saml-and-oauth-2.html

[15] R. Boyd, *Getting Started with OAuth 2.0*, 2nd ed. OReilly Media, 2012.

[16] G. Brail and S. Ramji, *OAuth - The Big Picture*. Apigee, 2014. [Online]. Available: http://pages.apigee.com/rs/apigee/images/oauth-ebook-2012-02.pdf

[17] D. Hardt. (2012, October) The OAuth 2.0 authorization framework. [Online]. Available: https://tools.ietf.org/html/rfc6749

[18] M. Jones and D. Hardt. (2012, October) The OAuth 2.0 authorization framework: Bearer token usage. [Online]. Available: https://tools.ietf.org/html/rfc6750

[19] J. Richer and W. Mills. (2012, November 28) OAuth 2.0 message authentication code (MAC) tokens. [Online]. Available: https://tools.ietf.org/html/draft-ietf-oauth-v2-http-mac-02

[20] M. Jones, B. Campbell, and C. Mortimore. (2015, May) JSON web token (JWT) profile. [Online]. Available: https://tools.ietf.org/html/rfc7523

[21] B. Campbell, C. Mortimore, and M. Jones. (2014, November 12) SAML 2.0 profile for OAuth 2.0 client authentication and authorization grants. [Online]. Available: https://tools.ietf.org/html/draft-ietf-oauth-saml2-bearer-23

[22] IBM.com. (2015, September 8) Invoking the authorization endpoint for openid connect. [Online]. Available: http://www-01.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.wlp.core.doc/ae/twlp_oidc_auth_endpoint.html

[23] Openid.com. (2014) What is OpenID Connect? [Online]. Available: http://openid.net/connect/

[24] N. Sakimura. (2014) OpenID Connect Core 1.0 incorporating errata set 1. [Online]. Available: http://openid.net/specs/openid-connect-core-1_0.html

[25] J. Basney, J. Gaynor, and W. Edwards. (2013) OpenID connect for myproxy protocol specification. [Online]. Available: https://redmine.ogf.org/dmsf_files/13113?download=20852

[26] Connect2id.com. (2015) OAuth 2.0 authorisation endpoint. [Online]. Available: http://connect2id.com/products/server/docs/api/authorization
[27] Oasis-open.org. (2008, March 25) Security Assertion Markup Language (SAML) v2.0 technical Overview. [Online]. Available: http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html
[28] N. Naik, P. Jenkins, P. Davies, and D. Newell, "Native web communication protocols and their effects on the performance of web services and systems," in *Computer and Information Technology (CIT), 2016 IEEE International Conference on*. IEEE, 2016, pp. 219–225.
[29] N. Naik and P. Jenkins, "Web protocols and challenges of web latency in the web of things," in *Ubiquitous and Future Networks (ICUFN), 2016 Eighth International Conference on*. IEEE, 2016, pp. 845–850.
[30] J. Somorovsky, A. Mayer, J. Schwenk, M. Kampmann, and M. Jensen, "On breaking saml: Be whoever you want to be." in *USENIX Security Symposium*, 2012, pp. 397–412.
[31] SANS. (2003) Global information assurance certification paper. [Online]. Available: https://www.giac.org/paper/gsec/2876/saml-common-security-language-web-services/104846
[32] J. Naithan, "SAML proposal for securing XML web services.Project paper," *University of St. Thomas, Saint Paul*, 2008.
[33] J. Hodges, O. C. McLaren, P. Mishra, R. Netegrity, T. Moses, E. E. Prodromou, and S. M. Erdos, "Security and privacy considerations for the oasis security assertion markup language (saml)," 2002.
[34] V. Mladenov, C. Mainka, and J. Schwenk, "On the security of modern single sign-on protocols: Second-order vulnerabilities in openid connect," *arXiv preprint arXiv:1508.04324*, 2015.
[35] T. Groß, "Security analysis of the saml single sign-on browser/artifact profile," in *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*. IEEE, 2003, pp. 298–307.
[36] D. Fett, R. Küsters, and G. Schmitz, "A comprehensive formal security analysis of oauth 2.0," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1204–1215.
[37] R. Millman. (2016, January 11) Researchers find two flaws in OAuth 2.0. [Online]. Available: https://www.scmagazine.com/researchers-find-two-flaws-in-oauth-20/article/530038/
[38] G. Curtis. (2016, July 16) Preventing mix-up attacks with OpenID Connect. [Online]. Available: http://openid.net/2016/07/16/preventing-mix-up-attacks-with-openid-connect/
[39] A. Armando, R. Carbone, L. Compagna, J. Cuellar, G. Pellegrino, and A. Sorniotti, "From multiple credentials to browser-based single sign-on: Are we more secure?" in *IFIP International Information Security Conference*. Springer, 2011, pp. 68–79.
[40] W. Li and C. J. Mitchell, "Analysing the Security of Google's implementation of OpenID Connect," in *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2016, pp. 357–376.