IJRAR.ORG E-ISSN: 2348-1269, P-ISSN: 2349-5138

INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS (IJRAR) | IJRAR.ORG

An International Open Access, Peer-reviewed, Refereed Journal

Single Sign on in Java

Niravkumar K Patel

University of the Cumberlands

Abstract

Single Sign-on is important to use in the Distributed Environment System. There are several authentications required to authenticate the other Tenant as part of the Single sign-on and It is hard to modify the old legacy application into to Single sign-on application. Which are using a username and password. The old authentication is just authenticated in a single application. We need credentials to store usernames and passwords in the database and authenticate via the Internet. It will try to verify the username and password from the database and verify the authentication. In single sign-on, there are all tenants to be provided to authenticate like Amazon, Azure, Google Cloud, and Oracle, if they have an account on these providers then They will authenticate with the same username and password to any application, so They don't need to create an account on the destination web application. Single sign-on is a secure authentication mechanism to authenticate without a client username and password. There are several authentications in Single sign-on like SAML, and OIDC base authentication. The single sign-on is easy to maintain and add other providers in the same application. It is very easy to maintain the user and providers.

Web applications to store the username and password are critical for any company with secure protocols and confidential processes. It is the most common security weakness to store information in the database. Single sign-on is the problem enabling companies to enable the federal environment in which clients sign in the environment once and They can use the services provided by the different organizations.

INTRODUCTION

The security assertion markup language (SAML) 2.0 web browser SSO profile is one of the standard technologies. It will define the XML-based format for user information per their tenant, like First Name, Last name, Middle name, Address, Email, and Phone number. All this information is to be encoded format in XML. It's contained in the number of protocol security assertions and a healthy number of protocol binding that will show how the report should be exchanged in the application and deployment criteria. SAML-based SSO is the core solution for authentication and authorization. Each SSO provider has the services available to provide SSO-based authentication as per their providers.

Three leading roles are part of the protocol—a client, an Identity provider, and a service provider. Web browsers provide the user's information, and the service provider and Identity provider aim to authenticate as per their corresponding authentication.

Background

Identity management is a concept that provides authentication and access authorization for the user in the internet provider. An external mechanism has performed the user access control. The security aspects are supplied by OAuth protocol and protocol based on the message component between internet infrastructure.

There are six major components: customer services, API gateway, Appliance technology, Authentication server, and Access Authorization. We can protect outside users who belong to another environment by prohibiting them from entering into infrastructure. The application has identifier attributes like number and symmetric key, provider by service provider. The symmetric key and unique ID are stored as customer service accounts, which the service provider can provide. The authentication will perform the service provider and identify the provider via secure protocol communication. It will share the token exchange and authenticate the application. The API gateway redirects the request as per the environment and targeted web application.

Users will be using the provided service, which does not have access to full service. It uses only the bridging elements for such applications and services, which are provided as per their roles and responsibilities. The authentication and access authentication servers will validate their application credentials, providing SSO for those services. We can use two key-based encryption algorithm processes to provide secure end-to-end

communication between the client and the application. First, the Customer and application use secret encryption based on the symmetric master key to provide the session key. Secondly, the application and gateway use public key encryption, and gateway and apps use that secret key encryption to message to protect and transmit the data, including application share.

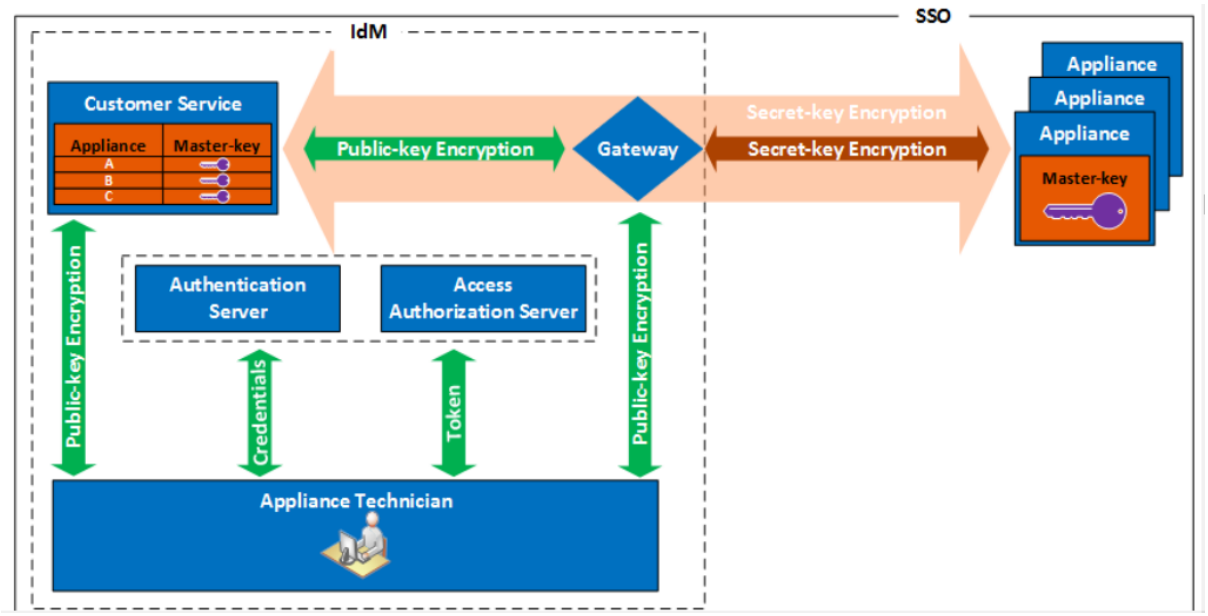


Figure 1

We can consider two end-to-end communications between applications and customers. One is started by the application and responded to by the customer and another client. The application creates communication and performs services requested by customers.

Figure 2

We can see in Figure 2 that end communication started by the application, and the message sequence is the same for all the requests—application service is provided by supplying the requested content. The application generates key encryption and message exchange during requests.

The application also uses key encryption and sends the encrypted value to the gateway with a unique identifier. Gateway parses the message to the internet and forwards the message to customers.

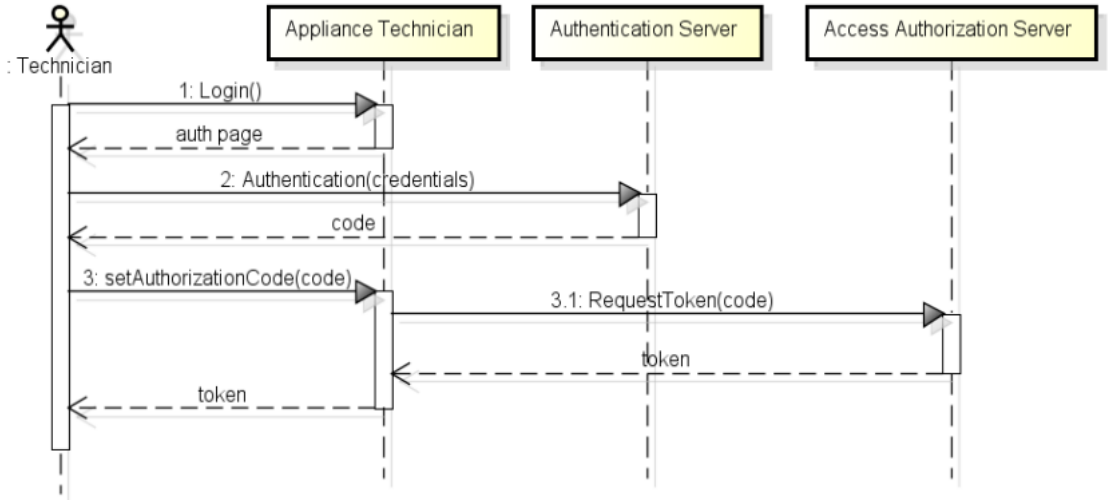


Figure 3

Authentication and access authorization for customers. The application receives an encrypted message with an encrypted key. It will decrypt the message and validate the reply through its value. The application encrypts the request with a new nonce and forwards it to the gateway. It supplies the session jey and index that will be passed for communication to the customer. Client secret retrieves the session key based on the session index and decrypts the message stored in the request to be replied to asynchronously by the customer. Client secret returns the session key index and responds to the nonce. The application receives the response, validates and answers to the nonce, and informs the subjects of the status of that particular request.

We can see in the Figure 2 diagram that the authentication and access authorization process for customers to access the application. Customers request a particular application and are redirected to authenticate the server call with its requested credentials. The authenticate server will validate customers' credentials and reply to a code in a short, valid time. It will be used to request an access token to access the authenticated server. The access authenticate server returns a pass, which the customer can use to reply to an application request. When a user is authenticated and authorized. The user has access to that application and data requests for that application. User can retrieve key encryption and their index and, decrypt the request, process it. It forwarded to an encrypted response along with an access token. The access token has all the information for user identification, service provided, Tenant information, session expiration, and application receipt.

The second type of communication a customer starts is applicable when they want to collect the information and perform the maintenance operation. For an example,

The application software has been upgraded, and this communication follows the call-back procedure. Suppose we see in the figure. The user requests to the client-server some application data in event one and passes the token with a unique number. Then, the Client server will validate the ticket and reply to the application data, including the gateway address according to the application. Application request to the gateway for the application and start the communication with it in event 2. The gateway validates the access token and notifies the application, and the application will create a session following the contact.

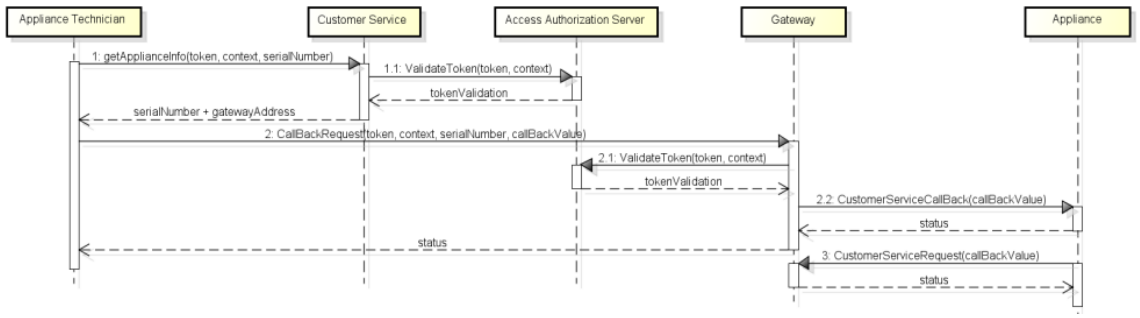
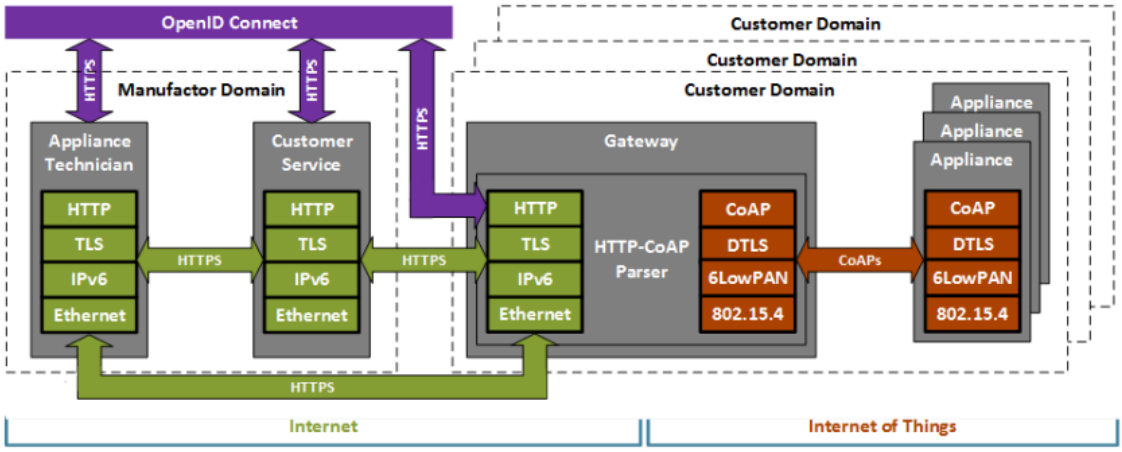


Figure 4



Prototype

In this section, we represent a prototype that can be implemented as an authentication scheme based on Identity management and proposed access authorization. The prototype used standard, well-known technology and open-source coding libraries in Java.

Implementation.

The manufacturing domain contains two component applications, and the client-server was implemented using the framework. Client-server is implemented in restful web service using spring java-based API.

The customer domain consists of one gateway and several applications. API gateway interface deployed on HTTP servers implemented in JAVA. Gateway can parse the message between HTTP and COAP protocols and vice-versa.

Based on the Florida project, the live application is implemented in Java and can be executed on ContikiOS. We are using AES 128 but system. We are using 128 algorithms to encrypt the key and decrypt the key.

This is an application working lively in the company right now. The application authenticates 1000 service providers to buy a product into an e-commerce platform. The customer tries to do SSO on the business central, and It will try to out their product to sell in the market as per their company SSO credentials. They will have all the invoice information for their personal use. The application provides the functionality to create a user account, set the password, Update the address, and several other API services to consume as per their needs. The business central has the functionality to handle all the requests efficiently with such secure algorithms as SAML 2.0 and OIDC-based SSO authentication. OIDC SSO

The specification of the attributes-based access control using multi-domain roles to get the standards to enable the model. The OpenID connect, based on OAuth specification, is used to specify the authentication control over the application. The identity token is used to store the authentication control. The OIDC validates the user credentials and returns a nonce to the application as an event. The application uses the nonce to get the access token and identity token. The token contains the user information and validity of the access token, Tenant details, other primary details, etc. The application performs the online validation process of the access token and returns the roles available for that user to choose which role they want to activate. The role activation is performed by sending the access token and the selected parts to start the application and service.

Multi-domain authorization is challenging in this authentication. We have all domains to be authenticated as part of this authentication mechanism. All the multi-domains are to be added in a single environment, and after adding those domains. They have access to this common domain. As part of the multi-domain functionality, we have created algorithms to create role mapping through match operations. Such operations perform an intersection among each role involved when merging a domain to yield the new role permission set. This is a centralized operation as per demand for some processing power. We have all domains to be in a

single place. It is required to re-do operations to role-based operations. Which resource is to be shared as per their role and domain? Which is the resource available to use to that domain user.

Role-based access control is a service provider. It performs the access control, using the role's user attributes. It is implemented attributes evaluation, and policy enforcement attributes-based access control and role-based access control form the core of the implementation.

The user requests the authentication of the application. The application creates a session for that user and requests authentication to the authentication control, providing the user session as an input parameter. The user offers her credentials that, when it is valid to, make an authentication control. The application requests the token to the access authorization server. The access authorization server provides an access token that allows the user to access role-based access control and attribute-based access control to activate the user roles and access the resource.

The access control model was implemented with each component working as a service, requiring an access token that provides online access authorization. Each service has two different scopes are required. One area allows only reading the resource, while the other allows updating the help. This restriction allows access to the application only for authenticated users with valid access tokens. To illustrate, to activate the role application. The user must have an access token in the application scope. To access the resource. The user has read the content and full scope in the token.

The application is to be created in the distributed environment first for all the users and role-based users. The application is centralized over the network for authentication. The credentials are the same as per the network as the service provider—the application to be created OAuth configuration to authenticate role-based users for the resource and services. The user can try SSO authentication on the web application, and the request will go to the application registration as per session key. The OAuth service tries to authenticate the user as per their credentials and the scope of the user. It will return the JWT token with all the resource and user-level information per access token. The token is a key to use everywhere in the service or application functionality.

Token generation is how service providers forward the end user to the identity provider. This is usually HTTO redirects the registered URL on the identity provider with additional parameters. The user can try logging in to the identity provider, which generates the SSO token. This token is submitted to service providers.

Token redemption is the last step, where service providers receive the SSO token in order. This is a security-critical process. The token contains multiple parameters that must be verified. SSO token Structure is independent of the SSO protocol implementation; Each SSO token has the information that can be structured below.

Identity: The SSO token contains the information about the user authenticating at the service provider. In some protocols, This is an email address like OpenID, OpenID connect, and browser ID such as OpenID, browse id, OAuth, and OpenID connect; the IDP is trusted.

Challenges:

To handle the multi-tenancy feature from service providers and identify providers for different tenants. The API gateway multi tenancy feature transforms the request to different environments. Pass the JWT token to another application so they can use the information after the authenticated user. Try to get the refresh token from the access token to provide the access token. In openId connect, We have faced several challenges, such as consuming the token from the other rest service and using the authentication for that service. We have used ForgeRock as an identity provider to design the GUI application and, internally, call the rest of the service to authenticate. The service will show the Azure Enterprise app registration if the service is established via the service provider. In that app registration, we have configured the scope and access for role-based authentication per their environment. The tenant has to be added to the app registration to authenticate. All app registrations have full Access to all the logs like Audit logs, OAuth logs, access logs, IP logs, etc. When the user is authenticated via the browser, we have an API gateway as Gloo to route the request to the targeted resource URL. Gloo validates all the valid or invalid requests from app registration. This is a complex SSO component for the company because all the customers will use the same element for authentication and authorization. Moreover, they have all the payment API gateway to process the payment, so All the necessary testing has been done as part of the miscellaneous attack during penetration testing.

Conclusion

I presented the authentication method to integrate IDM from the internet context to IOT. An API gateway involves routing all the requests to the targeted machine after authentication and before authentication—the multi-tenant authentication is processed as API gateway and service provided, identity provider. The performance of the matter while SSO, so we have reduced the execution of the application to less than 20 to 30 seconds as per usual. We can conclude that the openId and SAML-based SSO are new learning curves in life, and It's an excellent experiences as an individual. The application is complex, and It was hard to maintain when we started the implementation. Still, We have overcome all the problems and designed robust authentication components efficiently with high performance. We have tested the part from cross-phase, API, token-based attacks, etc. All attributes are based on the multi-domain role activation model, which considers the different semantics roles and allows the Single role activation. We used OID to provide the SSO among the various domains, and Oauth acted as admission control for all the services because each service user separated duty. The component performance was slow while we developed it, but we tried to optimize the performance to avoid some rest calls and optimize effectively in code optimization techniques. We have analyzed all the requests to be forwarded and wanted to optimize them to prevent the rest of the calls.

References

Z. Shelby, K. Hartke, and C. Bormann, The Constrained Application Protocol (CoAP), IETF RFC 7252.

E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2.", IETF RFC 6347.

J. Liu, Y. Xiao, and C. L. P. Chen, "Authentication and Access Control in the Internet of Things," in Proc. of theICDCSW - Intl. Conf. on Distributed Computing

D. Van Thuan, P. Butkus, and D. Van Thanh, "A user centric identity management for Internet of things," in Proc. of the ICITCS - IT Convergence and Security, 2014, pp. 1–4.

P. Fremantle, B. Aziz, J. Kopecky, and P. Scott,"Federated Identity and Access Management for the Internet of Things," in Proc. of the Int. Workshop on Secure Internet of Things, 2014, pp. 10–17.