Security Solution Based on WSO2 IS

Yuanyuan Xu [1, a], He Yang[2, b]
[1] Department of computer science, HuBei University of education, Wuhan, 430205, China
[2] Department of computer science, HuBei University of education, Wuhan, 430205, China
[a]email: xyy_xuxu@163.com, [b]email: yanghe@whu.edu.cn

Keywords: API Security;WSO2; OAuth

Abstract. this document introduces the requirement of identity management in business and the enterprise product WSO2 Identity Server, as well as the key components: Inbound authentication, Authentication framework, Local authenticators, Federated authenticators, Provisioning framework and IdP and SP configurations. After introduce the architecture, this document detailed depicts the typical usage Securing APIs with OAuth & SOAP Security using Username Token. This document also list sample code to show the usage of Securing APIs with OAuth & SOAP Security using Username Token. Finally, this document introduces the feature Identity Federation and SSO. By using these features, WSO2 Identity Serve can cover most security scenario in SOA.

Introduction

Authentication, Identity, controlling access, security and provisioning requirements are rapidly evolving over the years. Authentication and authorization mechanisms cannot only span a single boundary of trust. Hence, more and more services are provided across trust boundaries, which may include partners, subsidiaries, customers or suppliers and may span across global organizations. Security becomes important in SOA.

Wso2 Identity Server

WSO2 provide enterprise SOA products, it's 100% open source and support cloud. WSO2 identity server provides sophisticated security and identity management for web application (include cloud application), web service and APIs.

Access Control: WSO2 IS uses XACML (eXtensible Access Control Markup Language) to support Role-based access control (RBAC) and Attribute-based access control (ABAC).

API Security: WSO2 Identity Server supports 2-legged and 3-legged OAuth to access third party service providers.

Architechture Introduction

WSO2 IS supports authentication request include SAML SSO, OAuth, OpenID and Passive STS. WSO2 IS uses Inbound Authentication component to handle request and send on to the authentication framework. The provisioning framework of WSO2 IS can be integrated with the User Store Manager component to access LDAP, JDBC etc.

This Server Model see Figure 1

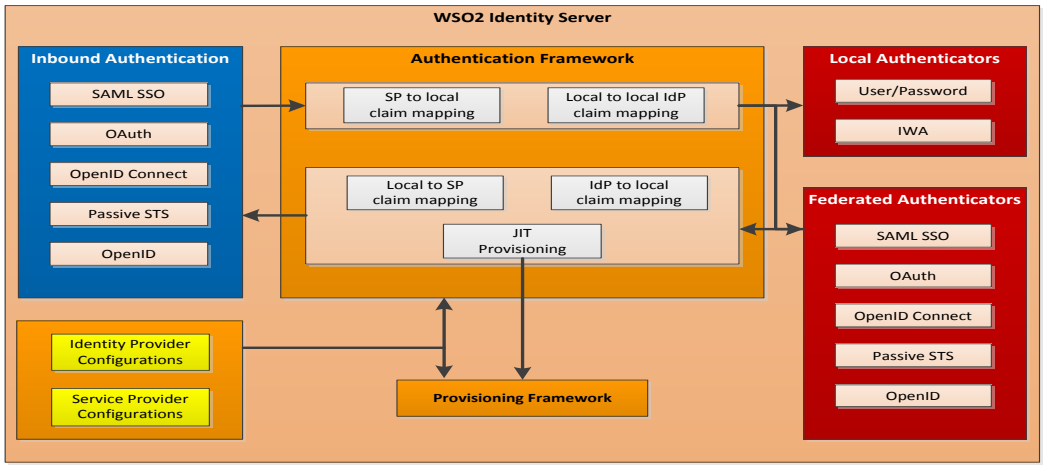

Fig.1. WSO2 Identity Server

Inbound Authentication

Inbound authentication component supports the request includes:

SAML SSO: Security Assertion Markup Language (SAML) is an OASIS open standard. SAML provides a web-based Single-Sign-On capability by exchanging user identity and authentication data between systems.

OAuth/OpenID Connect: OAuth 2.0 has 3 phases: requesting an Authorization Grant, exchanging the Authorization Grant for an Access Token and accessing the resources using this Access Token. OpenID Connect is another identity layer base on OAuth 2.0.

Passive STS: A Security Token Service (STS) is a software based identity provider.

OpenID: OpenID is a URL issued by an OpenID Provider. The OpenID Provider maintains user profile.

Authentication Framework

The key feature is claim management, it maintains the mapping between local claims and service provider claims. Just-in-Time provisioning can automatically create users.

Local authenticators

Local authenticators support authentication processes by checking the username and password or Windows-based authentication (IWA).

Federated authenticators

Federated authenticators support authentication processes with external applications.

Provisioning framework

The provisioning framework support integration with User Store Manager Component.

IdP and SP configurations

The identity provider and service provider configurations for Service Provider and Identity Provider.

Process

Web service is wide used, so below processes describes the steps for securing of RESTful and SOAP usage:

Securing APIs with OAuth

This sample process is based on WSO2 IS and WSO2 ESB. RESTful service is wide used, so below processes describes the steps for securing a RESTful service RESTful service:

Step 1: Register a user with WSO2 Identity Server：WSO2 IS provides web console to manage user profile, and import feature to load user via domain or JDBC.

Step 2: Register consumer secret on WSO2 IS.
Step 3: Set up the consumer key as user name.
Step 4: Generate OAuth Authorization header.
Step 5: Invoke the proxy service.
Step 6: OAuth mediator in ESB invokes the service on WSO2 IS to verify consumer..
Step 7: Verify consumer key and verify the oauth_signature value.
Step 8: If request is authenticated, it will sent to the RESTful service.
SOAP Security using UsernameToken
Step 1: Configuration key store.
Step 2: Securing target web service on ESB..
Step 3: Add target web service as trusted service of Security Token Service (STS).
Step 4: Secure STS with Non-repudiation.
Step 5: Add claims to the user.
Step 6: Client test.

Core Code

Two legged OAuth
Below is a code segment for OAuth :

```
..
//User invoking the service
String USER_NAME = "xxx";
//User password
String PASSWORD = "xxx";
//Create a configuration context. A configuration context contains information
configContext = ConfigurationContextFactory.createConfigurationContextFromFileSystem( null, null);
// The server certificate is looked up in the trust store.
//Following code sets what trust-store to look for and its JKs
//password.  Note : The trust store should have server's
//certificate.
System.setProperty("javax.net.ssl.trustStore", path + File.separator +"wso2carbon.jks"); System.setProperty("javax.net.ssl.trustStorePassword", "wso2carbon");
//Here we are authenticating the given user name and
//password with IS
client = new AuthenticationServiceClient(IDENTITY_SERVER + "services/", configContext);
…
authStub = new AuthenticationAdminStub(configCtx, serviceURL); authStub._getServiceClient().getOptions().setManageSession(true);
authenticate = authStub.login(userName, password, remoteIp);
// Setting user name as consumer key oauthParameters.setOAuthConsumerKey(CONSUMER_KEY);
// Setting above assigned consumer secret oauthParameters.setOAuthConsumerSecret(CONSUMER_SECRET);

// setting 2-legged OAuth flag
        oauthParameters.setOAuthType(OAuthType.TWO_LEGGED_OAUTH);
// We will be using HMAC-SHA1 signature. Google API has a class to do that
OAuthHmacSha1Signer signer = new OAuthHmacSha1Signer();

// Create a sample service. The name of the current application given here
  // Names are only for reference purpose
SampleService service = new SampleService ("oauthclient", "sampleapp");
```

```
    service.setOAuthCredentials(oauthParameters, signer);
…
    SOAP Security using UsernameToken
    Below is the core code of client:
    // set the trust store as a system property
    System.setProperty("javax.net.ssl.trustStore", keystorePath);
    System.setProperty("javax.net.ssl.trustStorePassword", keystorePwd);
    // create STS client
    STSClient stsClient = new STSClient(configCtx);
    …
     // request the security token from STS.
    Token responseToken;
    Policy stsPolicy = loadPolicy(stsPolicyPath);
     …
    responseToken = stsClient.requestSecurityToken(null, stsEPR, stsPolicy, relyingPartyEPR);
    // store the obtained token in token store
    TokenStorage store = TrustUtil.getTokenStore(configCtx);
    responseTokenID = responseToken.getId();
    store.add(responseToken);
    // create service client
    ServiceClient serClient = new ServiceClient(configCtx, null);
    // load policy of secured service
    serClient.getOptions().setProperty(RampartMessageData.KEY_RAMPART_POLICY, sec_polic
y);
      // Set the token id as a property in the Axis2 client scope
    serClient.getOptions().setProperty(RampartMessageData.KEY_CUSTOM_ISSUED_TOKEN,re
sponseTokenID);
     // set action of the Target Service to be invoked.
     serClient.getOptions().setAction("urn:helloWorld");
    serClient.getOptions().setTo(new EndpointReference(relyingPartyEPR));
      // invoke the service
    responseElem = serClient.sendReceive(getPayload(echoRequestMsg));
```

Identity Federation and Single-Sign-On (SSO)

Identity federation & SSO are able to provide, consume services across trust boundaries. Identity federation enables users to access different applications but using the same access credentials. This makes access easy, and users do not have to remember all the credentials for every application. However, the users have to provide their credentials to each one of the applications separately, this make the maintenance complex. On the other hand, SSO enables users to provide their credentials once and obtain access token to multiple applications. In SSO, the users are not prompted for credentials when accessing each application until the session is terminated.

Conclusion

This document introduces the WSO2 Identity Server, as well as the key component, process and important topics include: controlling access, Identity Federation and Single-Sign-On (SSO). This document also depicts the usage of API security and SOAP security by using sample code. Base on above introduction and sample code, web service security could be simple by using WSO2 IS.

Acknowledgement

In this paper, the research was supported by the natural science foundation of Hubei Province

(Project No. 2014CFB569) and the research project of Hubei Province Department of Education Grant (Project No.XD2014349).

References

[1] James Snell，Doug Tidwell，Pavel Kulchenko. Programming Web Services with SOAP [M]. O'Reilly Media .2011.

[2] Chien-Cheng Lin, Chen-Liang Fang, Deron Liang. A portable interceptor mechanism for SOAP frameworks [J]. Computer Standards and Interfaces, 2013, 36(1) .

[3] Mowery,D., Wiebe,J. , Visweswaran,S. , Harkema,H. , Chapman,W. Building an automated SOAP classifier for emergency department reports [J]. Journal of biomedical informatics, 2012, 45(1).

[4] ZHANG Gong-xuan, SONG Bin, WANG Ping-li. Security Policy of Network Messages Based on SOAP[J]. Journal of Nanjing University of Science and Technology. 2007, 31(1).

[5] Renugambaal Nadarajan, Razali Ismail. Performance and Microstructural Study on Soap Using Different Fatty Acids and Cations [J]. Journal of Surfactants and Detergents. 2011, 14(4).

[6] Xue, K, Hong, P, Ma, C. A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture[J]. Journal of Computer and System Sciences. 2014, 80(1).

[7] Sandeep K. Sood, Anil K. Sarje, Kuldip Singh. A secure dynamic identity based authentication protocol for multi-server architecture[J]. Journal of network and computer applications. 2011, 34(2).

[8] Elena Torroglosa-Garcia, Antonio D. Perez-Morales. Integration of the OAuth and Web Service family security standards [J]. Computer networks. 2013, 57(10).

[9] Gonzalez JF, Rodriguez MC, Llamas Nistal M, Rifon LA. Reverse OAuth: A solution to achieve delegated authorizations in single sign-one-learning systems[J]. Computers & Security. 2009, 28(8).