1. Chapter 1 Introduction
   1. Introduction to E-government Platform
   2. Introduction to the PAAS Cloud Architecture Platform
   3. Introduction to the Existing Work
   4. Main Work
   5. Content Organization
2. 用户特点
3. Chapter 2 Relevant concepts and theories
   1. B/S model and SSO

单点登录的好处

Benefits

Benefits of using single sign-on include:

Reducing password fatigue from different user name and password combinations

Reducing time spent re-entering passwords for the same identity

Reducing IT costs due to lower number of IT help desk calls about passwords

SSO shares centralized authentication servers that all other applications and systems use for authentication purposes and combines this with techniques to ensure that users do not have to actively enter their credentials more than once.

Connected to: Authentication OpenID Login

From Wikipedia, the free encyclopedia

* 1. SAML and Oauth

1. 用户试图访问SG

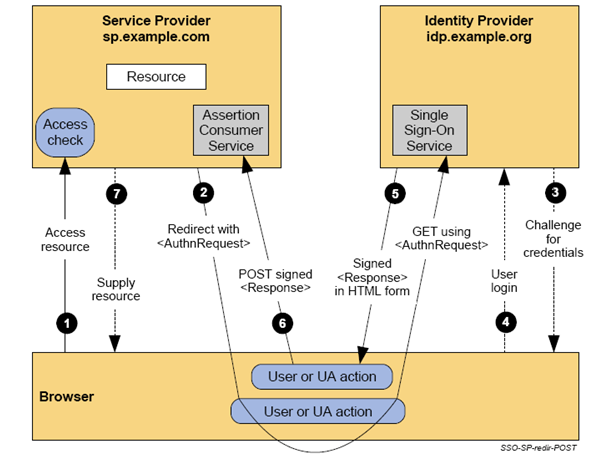
2. SG 发一个AUTHNrequest 到 IDG

3. IDG 要求用户登录

4. 用户登录

5. 登录后把SAML 发到浏览器

6. 浏览器再转发到SG

7. SG 验证SAML 请求后转到SG 应用

SAML 是一种认证协议， 主要用于SAAS的单点登录

SAML 过去， 现在和未来

由OASIS安全委员会制定的SAML, 逐渐成为一个流行的单点登录解决方案。 尽管传统上看SAML很复杂， 要很多花费, 但是有了SSO365 的产品和解决方案，企业不论规模如何， 从大公司到小公司， 都可以瞬间部署SAML 兼容的解决方案。

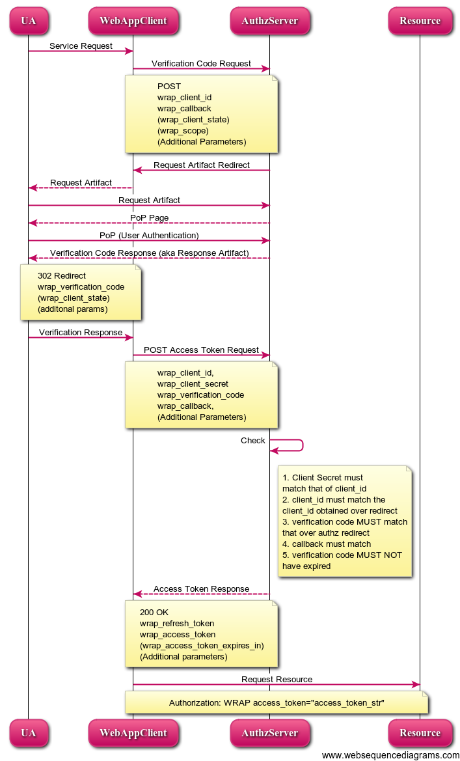
SAML又译作安全断言标记语言， 英文全称是Security Assertion Markup Language， 其实翻译成安全令牌标记语言更合适。 它是一个基于XML的标准，用于在不同的安全域(security domain)之间交换认证和授权数据。与传统的单点登录产品不同， 它主要解决的是跨域的单点登录问题。 比如一个公司的员工可以通过SAML访问saas 应用不需要登录。

如左图所示， SAML标准定义了身份提供者(identity provider)和服务提供者(service provider)，这两者构成了前面所说的不同的安全域。 它主要解决了在IDG 中认证的用户， 可以访问SG 不需要重新验证。 原因很简单， 因为IDG 和SG 有一个互信的关系。

在大多数计算机连入互联网之前，各个系统中像身份验证和授权这类安全服务的实现完全是独立的。因此，执行身份验证所需的全部代码，以及密钥、口令，供授权决 策所用的用户信息，以及授权策略本身均存放于使用这些信息的系统上。最初，系统连接到网络上时情况变化不大。每个系统都是一个孤岛，各系统都要求用户拥有 一个账户才能访问该系统， 我们统称为系统为主（domain centric）的用户认证系统。随着互联网的出现，多台机器作为一个Web站点的主机成为一种普遍现象。但仅仅因为用户要使用不同的机器处理不同的请求而强迫他们多次在网络上进行登录，显然是令人无法接受的。在SAML 产生之前， 人们一直在寻找类似的解决方案。 因为越来越多的应用迁移到云中， 所以这个问题变的日益迫切。为了解决这个问题， 2001年， Sun 等公司联系了一些同行, 如Oracle, CA, Fedelity, Intel, AOL等， 成立了自由联盟（Liberty Alliance）, 试图制定一个标准。2003年， Liberty Alliance将他们制定的标准推荐个OASIS组织安全服务技术委员会(Security Services Technical Committee)， 从而使SAML 成为一个OASIS的标准。 2007年，Gartner认为SAML 已经成为事实上的工业标准。 到目前为止， 全球有70多家支持saml 的产品， 包括微软， IBM， Oracle, CA等。因为大公司关注的是如何整合自己的产品线， 相对兼容性较差。 这样就出现了象SSO365等专注SAML来解决用户身份安全 ， 同时有良好兼容性的公司。

综上所述， 开放成为SAML 流行的一个根本原因。 之前ibm和微软曾经主推过WS-federation， 但是因为它的封闭性，并没有得到众多厂商的支持， 巨人们最终也选择了二者兼而有之。 因为SAML 安全性高， 甚至被选择作为支付的凭证， 在金融，保险和电商得到了很好的应用。

随着移动设备的发展，SAML 也受到了不少的挑战。 因为整个协议是基于XML 的， 对WEB-SERVICE 有很好的支持， 但对基于JSON 的REST 支持较差。 未来的SAML ， 讲充分发挥其安全的一面， 和OAUTH 结合， 衍生出新的应用案例。



1. UA(用户)试图访问WebAppClient

2. WebAppClient 要求AS 给予授权

3. AS 返回一个Authorization Code

4. 用户用这Authorization Code通过UA 传递给WebAppClient

5. WebAppClient换回一个Access Token

6. WebAppClient使用Access Token 来访问RS

7. RS 通过AS 再次验Access Token

8. RS 开放应用给Web

OAuth 是一种授权协议， 主要是针对移动应用

OAuth 过去， 现在和未来

OAuth得到众多大的社交网络的支持， 如谷歌， 雅虎， 脸书等， OAuth 协议采用了业界最好的Web 认证技术，统一形成一个开放的标准。 SSO365 的各种社交网络接口， 支持最终用户不间断地访问国内和国外的应用。

OAuth是IETF互联网工程任务组2010年制定的一个标准。 同SAML 产生的背景类似， 随着各种各样网站的不断增多, 人们在寻找减少用户名和密码, 同时解决访问授权问题的的解决方案。每个网站都有自己的用户名和码显然是件让人沮丧的事情， 但是， 整个用户信息的共享应该是受控的， 因为隐私的关系， 显然你不会把你的身份信息共享给一个给你的宠物找伴侣的网站， 但是给家庭朋友聚会的组织者就不同了。 用户觉得，比如一个登录到一个大的网站， 如google, yahoo等， 如果能把它作为身份认证机构， 在访问一些小的网站时， 使用同样的身份就好了。 于此同时， 用户应该自己可以决定信息共享的程度， 这就提出了另外一个概念以用户为主(user centric)的认证授权方式。

如左图所示,产生了两个新的概念AS (认证服务器， authorization server), 和 RS (资源服务器, resource server). 用户首先通过应用程序访问AS, 登录， 并询问用户是否允许把上面的用户信息开放给第三方网站RS。 如果用户选择了同意，那么就变得简单了。 AS会先向browser发放一个临时的token， 叫authorization code, 然后传递给应用程序。 应用程序用这个authorization code来换取一个access token， 然后放在header中去访问RS。 RS 收到后， 可以和AS 来重新验证， 也可以直接信任， 从而开放服务给用户。

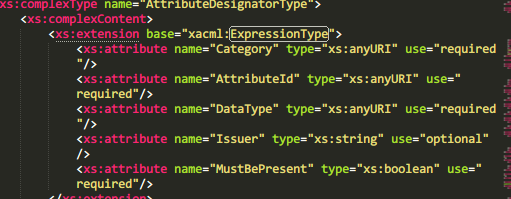
之前， 人们有Flickr Auth、Google's AuthSub、Yahoo's BBAuth、 Facebook Auth等。2006年， Twitter 提出的 Twitter OpenID标准。 2007年， 人们开始坐下来讨论一个标准， 这样， OAuth core 1.0 就诞生了。 在2008 年，人们把它提交给了 IETF(Internet工程任务组 InternetEngineeringTaskForce )。 IETF又叫互联网工程任务组，成立于1985年底，是全球互联网最具权威的技术标准化组织，主要任务是负责互联网相关技术规范的研发和制定，当前绝大多数国际互联网技术标准出自IETF。这样在， 2010年， OAuth 作为RFC 5849 号标准发布了。2012 年， OAuth 进化成RFC 6749. 2.0 发布了。 OAuth 2.0是OAuth协议的下一版本，但不向后兼容OAuth 1.0. OAuth 2.0关注客户端开发者的简易性，同时为Web应用，桌面应用和手机，和智能家居提供专门的认证流程。

OAuth协议为用户资源的授权提供了一个安全的、开放而又简易的标准。任何第三方都可以使用OAuth认证服务(AS)，任何服务提供商(RS)都可以实现自身的OAuth认证服务，因而OAuth是开放的。业界提供了OAuth的多种实现如PHP、JavaScript，Java，Ruby等各种语言开发包，大大节约了程序员的时间，因而OAuth是简易的。

目前， OAuth 得到了众多社交网站的推崇， 包括国内的QQ, 人人， 新浪微博等。 SSO365的OAuth 解决方案， 可以在认证用户后， 充当AS, 给RS 提供授权令牌Access Token.

随着协议的不断推广， OAuth 一个缺点也越来越明显。 比如对于ACCESS TOKEN， OAuth 并没有严格的定义。 导致各个厂商生成的ACCESS token难以互换使用。 随着时间的推移， 人们在呼唤新的协议如OpenID Connect (注：非OpenID) 等的出现。

* 1. ABAC and XACML





* 1. REST and SOAP API
  2. Java Web Application
* Java
* MyBatis
* Spring
* MVC
* Bootstrap
* Axis
  1. Clustering and Encryption algorithm

1. Chapter 3 System Analysis

Identity Must Meet the Cloud’s Needs

Five areas of the identity stack must evolve to realize a cloud-scale identity fabric: access control and autho­rization; authentication, federation, and single sign-on (SSO); user account management and provisioning; auditing and compliance; and cloud platform architectural requirements.

* Access control and authorization

Managing access control to premises-based and cloud apps is a complex undertaking. In the cloud, outside the firewall, perimeter controls cannot be relied upon to con­trol even binary access. Today, many users are outside the private network and access SaaS apps over the Internet with no need to go through the company network. Autho­rization, therefore, must evolve to a distributed model to support users outside the network firewall.

The authorization in depth concept captures the vary­ing granularity of authorization policies across three levels. Level 1 is a coarse-grained access control policy that governs users’ access to an application or resource. Level 2 is more fine-grained, controlling access to the data level - typically the URL. Level 3 is the most fine-grained level, controlling access to functions or views, sometimes referred to as “entitlements.” Any scalable authorization model must reflect the need to address multiple degrees of granularity or depth. Tying this back to scalability - the greater the granularity, the greater the volume of authori­zation transactions. Another key to scaling access control is grouping access. The earliest forms of access control on the mainframe were based on manually maintained access control lists (ACLs). ACLs initially worked well because few people used the mainframe, but as user bases grew, they became un­wieldy, leading to the use of groups. Group access control scaled well but still required manually managing group memberships. This led to dynamic group management using rules to determine membership and therefore access. Today, organizations use role-based access control (RBAC) instead of groups to reflect the enterprise’s member­ship and attribute-based access control (ABAC) to handle dynamic permissions. In the cloud, these attributes and role memberships are decoupled from the operating system and can be distributed across systems via federation.

Authorization can be scaled by using a distributed, federated model. By breaking the authorization pro­cess down into its core policy elements - management, decision, and enforcement - it is possible to federate authorization across technical and organizational do­mains. Policy management points, policy decision points, and policy enforcement points must run in distributed locations, especially across the cloud.

Identity decision points provide identity data for autho­rization rule evaluation and can utilize Security Assertion Markup Language (SAML) assertions or HTTP headers today and OAuth 2.0 in the future. For example, OAuth leverages a delegated trust model to realize the benefits of abstracting user identity data from user credentials and supports tokenization of authorization. It does require an OAuth-aware architecture of entitlements enforcement. Regardless of the technology used, these authorization decisions must happen quickly and support high volumes of traffic.

Ideally, developers will use the SAML equivalent for provisioning, the Service Provisioning Markup Language (SPML), but there are only a handful of real-world SPML implementations. Without federated provisioning APIs to enable automated synching of local accounts, SAML adoption will remain limited. There is also a lack of sup­port for integrating SAML attributes for personalization, session context, or just-in-time provisioning. The absence of universal user schemas for directories makes building general-purpose management tools difficult.



* Authentication, federation, and SSO

The federation concept is familiar inside the firewall, perhaps best exemplified by the ubiquitous Microsoft Windows domain model. Enterprises can link multiple Windows domains by defining the trust model between different organizations within the firewall and allowing authentication to be delegated to the “local” domain and trusted by a remote domain using Kerberos, making login transparent to the end user.

Modern federation takes this model beyond the pro­prietary Microsoft approach to make seamless SSO work across the Internet using SAML, an XML-based open stan­dard for exchanging authentication and authorization data between security domains - that is, between an identity provider and a service provider - instead of Kerberos.

The problem with federation and SSO is that, after more than a decade, SAML adoption has not risen above 10 percent of enterprise apps - apparently due to the exces­sive costs of infrastructure software. There simply is not enough return on investment for most service providers to implement, expand, and manage a complex federation network. The industry must therefore go beyond SAML and focus on the core HTTP authentication standard. It requires no change to the target app and no coordination between users and the application. HTTP is the gold stan­dard in authentication, with nearly 100 percent of Web apps supporting it.

* User account management and provisioning
* Auditing and compliance

A key challenge in auditing in the cloud is overcoming the lack of visibility into user access of SaaS apps. Using the public Internet rather than connecting to a company network puts users beyond the scope of network monitor­ing tools.

* Unlike most enterprise networks, the cloud is globally accessible. However, regulatory compliance require­ments vary by jurisdiction and are complex and often contradictory. The industry needs a framework to meet global jurisdictional challenges. Identity is central to such a framework because many regulations center on user privacy and access.
* Cloud platform architectural requirements

The cloud has brought with it new architectures and platforms that service providers have yet to make identity-aware. Specifically, many cloud service providers offer storage- or database-as-a-service via hosted hypervi­sors like KVM or those from VMware and Xen, but such IaaSs currently lack identity and access management as a service.

With their high utilization rates, virtualized platforms cannot handle the overhead associated with precloud Web access management (WAM) use of webserver plug-ins and agents. The tight coupling of WAM with plug-ins has proven to be brittle, and the “burstable,” elastic nature of virtualized cloud platforms makes the plug-in model infeasible. The industry requires a proxy-based approach that does not place load on the virtualized Web and ap­plication servers.

In the case of SaaS apps, the identity integration challenge of enforcing access control and supporting audits stems from multitenancy and the fact that the SaaS provider owns and operates the underlying infrastructure, making it impossible to install dedicated agents or plug-ins for each application instance. Also, with most SaaS apps, collecting audit logs is problematic because they are often comingled with other tenant data. In some cases, the audit details are insufficient for answering key forensic ques­tions. What is needed is a loosely coupled, noninvasive identity management platform that can enforce policy upstream from the SaaS apps themselves.

* 1. Architecture Summarize
     1. Function Architecture

2.6 认证 API

API（Application Programming Interface,应用程序编程接口）是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节。在统一身份认证系统范畴内，由于需要认证的系统在语言、功能、形式上各有不同，导致接入方式及接入程度也不尽相同。由认证系统提供基本的操作API，如登陆、登出、用户查询等60 余种，被认证方根据自己需求合理整合后接入认证系统。

。

2.9 系统管理与监控

系统管理和监控，需要对统一身份认证系统进行统一的管理与健康，比如支持内置标准的访问和性能统计信息的收集和监控，提供对一些关键度量进行监控和管理，支持企业级日志系统的集成等。

对于在软件系统设计和开发中遇到的重要问题，应该有相关的文档记录

策略存在冲突是怎么，要求提出解决策略冲突算法

* + 1. Description of Module Function
  1. Non-functional Requirements
  2. Development Environment

1. Chapter 4 System Design
   1. Preliminary Design
      1. The Architecture and Module

组件设计

* + 1. 用ABAC实现RBAC

权限控制

策略冲突时访问控制中授权一致性问题：

Jave WBE设计

* + 1. Module II‘

租户管理

* 1. Detail Design
     1. Data Structure Design
     2. Algorithms Design

1. Chapter 5 System Implementation

测试环境

本系统采用B/S架构组成，软硬件配置如下：

* 1. Core Code and Important Flow
     1. Flow I
  2. Modeling the Performance of System
     1. Concept and Definitions
     2. Model Assumptions
     3. Modeling the Performance

1. Chapter 7 Conclusion