网络出版时间：2016-06-14 11:26:34
网络出版地址：http://www.cnki.net/kcms/detail/51.1196.TP.20160614.1126.078.html

基于 Shibboleth 的在线实验平台多资源访问认证

张 禹[1]，陆慧梅[1]，向 勇[2]

(1. 北京理工大学 计算机学院，北京 100081；2. 清华大学 计算机学院，北京 100084)

摘 要：基于联盟认证的单点登录能够认证不同组织的用户，但资源的多样化导致了管理上的困难。针对该问题，采用 Shibboleth 联盟认证方式，规范多种资源访问的 REST API 接口，并借助 Shibboleth 的属性筛选策略发布访问资源的授权码，实现多组织用户访问多种复杂资源的统一认证。以基于 OpenEdX 的在线实验平台为例，解决了平台中统一认证以及复杂资源授权的问题，验证了采用 Shibboleth 进行用户统一认证，通过 REST API 接口以及授权码的发布可实现复杂资源共享，并在 OpenEdX 上以 XBlock 的方式实现与其余系统的数据交互。

关键词：Shibboleth；联盟认证；在线实验平台；OpenEdX

中图分类号：TP393.08

Identity of Shibboleth-based multi-resource on online experimental platform

Zhang Yu[1], Lu Huimei[1], Xiang Yong[2]

(1. School of Computer Science & Technology, Beijing Institute of Technology, Beijing 100081, China; 2. School of Computer Science &Technology, Tsinghua University, Beijing 100084, China)

Abstract: Federated identity is applied to achieve Single-Sign-On for the situation in which users are from different organizations, However, the diversity of resources brings about trouble of management. To solve the problem, this paper selected Shibboleth as a method of federated identity. System in federation provided REST API of their own resources, and the authorization code which identified the access rights of resource was released by the attribute publishing policy in Shibboleth, thus multi-resouce was accessed by users from different organazations after unified authentication. Take online experimental platform based on OpenEdX as an example, unified authentication and authorization of complex resources was implemented. Shibboleth was applied to authenticate users, coupled with REST API and authorization code, and complex resources were shared during several systems. Moreover, some XBlocks were developed on OpenEdX to share data with other systems.

Key Words: Shibboleth;federated identity; online experimental platform; OpenEdX

0 引言

随着网络资源的多样化，各类系统的规模不断增大，资源的管理愈发困难。例如，慕课学习和实验环境中包含了视频播放、作业发布和提交、在线答疑和交流、在线实验等多个系统，每个系统都拥有各自不同类型的资源，利用 REST API[1]可实现各系统间的信息交互，但无法保证资源访问的安全。并且由于用户来源于多个不同的组织，规模较大，普通的单点登录无法满足需求。

为了实现单点登录，目前常见的解决方案包括 CAS[2]，OpenID[3]和 OAuth[4]。这 3 种实现单点登录的方式有一个共同的缺陷，所有的验证都是交由同一机构(CAS 的服务器端，OpenID 的服务网站，以及 Oauth 的第三方验证系统)完成的，账号信息也统一存放(尽管可以是分布式的)；在多组织的情况下，随着用户规模扩大，认证端的负载以及账号的管理难度会不断增加。而 Shibboleth[6, 7] 作为联盟认证[5]的一个实现，它不负责实际的验证，用户验证提交给 Shibboleth 后被转发至用户所属的组织，用户的管理也由各自的组织完成，用户规模的扩大对 Shibboleth 影响很小。目前 Moodle 在线教学平台，Web of Service 科研资料库等一些由多个组织协作支持的系统都已经集成了 Shibboleth 登录。

Shibboleth 虽然能够对多组织用户进行统一的认证，但验证完成后只能返回用户的属性字段，无法对复杂资源授权。本文在 Shibboleth 的统一认证基础上，通过授权码+REST API 的方式解决该问题，并借助属性发布策略限制授权码的发布，使系统在通过 Shibboleth 认证后能安全地获取所需资源。为了进行验证，我们应用该方案解决基于 OpenEdX 的在线实验平台上统一认证以及资源授权的问题。项目仓库：https://github.com/xyongcn/online_experiment_platform.

1 统一认证与资源管理方案

1.1 系统框架

本文选取 Shibboleth 作为认证框架，实现多组织用户的统一认证并向认证系统发布用户属性。对于复杂资源的访问授权，Shibboleth 无法直接通过属性发布实现，需要以间接的方式控制。在 Shibboleth 用户属性中添加 REST API 的访问授权码字段，该授权码用于标识某一资源的访问权限，经用户所属组织同意(通过修改属性发布策略)，系统通过 Shibboleth 认证能够获取外部共享资源的授权码。系统携带授权码访问资源的 REST API 接口，以用户身份对资源进行操作。通过上述方式，Shibboleth 能够对共享的复杂资源的访问进行授权。

基于 Shibboleth 的系统框架如图 1。系统包含以下 5 个组件，其中 IDP 与 SP 已集成在 Shibboleth 中：

a)SP(server provider)，服务提供端。用于过滤用户的资源请求，WAYF(where are you from)提供 SP 信任的 IDP 列表，用户选择自己所属的组织后跳转至对应 IDP 进行验证，验证完成后 SP 向 IDP 查询用户属性，根据属性值判断用户是否有权限访问资源。本文描述的 SP 作为 Apache 的模块运行。

b)IDP(Identity Provider)，认证提供端。IDP 位于用户所属的本地机构，在不将用户身份泄露给 SP 的前提下，IDP 能向 SP 保证用户是合法的。在请求时，IDP 向 SP 返回用户的属性列表，这些属性已经事先被用户批准只用于授权，即经过了属性发布策略的筛选。

c)openLDAP, Shibboleth 不负责实际的认证，需要额外部署认证源，本文选取 OpenLDAP[8]作为实际的认证源存储用户信息，IDP 有权查询 OpenLDAP 中的用户属性并进行发布。OpenLDAP 是 LDAP(Lightweight Directory Access Protocol，轻量目录访问协议)的一个实现，能够支持客户端大量的读操作，目前在线实验平台包括两个 OpenLDAP 认证源。

d)待认证系统。SP 运行在待认证系统的 Apache 服务器上，服务器从 SP 获取用户基础属性用于在本地创建帐号，同时获取用户的资源授权码，待认证系统通过该授权码能以用户身份访问其余系统上的用户资源。

e)用户资源。系统管理各自的资源，种类不作限制，对外提供需验证的 REST API 接口。

待认证系统获得的用户属性先后经过 IDP，SP 的筛选，IDP 端的属性筛选表明对待认证系统的信任程度，SP 端的筛选用于过滤不需要的冗余信息。图 1 中的用户属性 3 包括用户基础属性以及授权码(如果需要的话)。用户基础属性交由待认证系统识别用户身份，判断用户是否有权限访问本地资源。授权码用于系统 A 访问用户的外部共享资源。系统 A 需要与用户所属的 IDP 协商，获得允许后通过修改 IDP 端的属性发布策略，向 A 发布资源 B 的授权码。系统 A 从 SP 获取授权码后即能访问资源 B 的 REST API 接口，对资源进行操作(操作权限由资源 B 所在的系统控制)。

待认证系统与 SP 通过 Apache 服务器进行交互。SP 能较好的作为 Apache 模块运行，对资源进行保护的配置也较为简便，任何访问指定 Apache 目录的 url 请求都会被转发至 SP 处理，浏览器端也会跳转至用户选择的 IDP 进行认证。对于其他非 Apache 服务器的系统，为避免大量的修改调试工作，一种通用的方法是:无论具体系统使用何种服务器，都将 SP 部署在 Apache 上，并在原系统服务器中为 Apache 保留某一端口，将 Shibboleth 登录的 url 请求转发至 Apache，由 Apache 负责后续的处理，验证完成后从该端口取回用户属性。

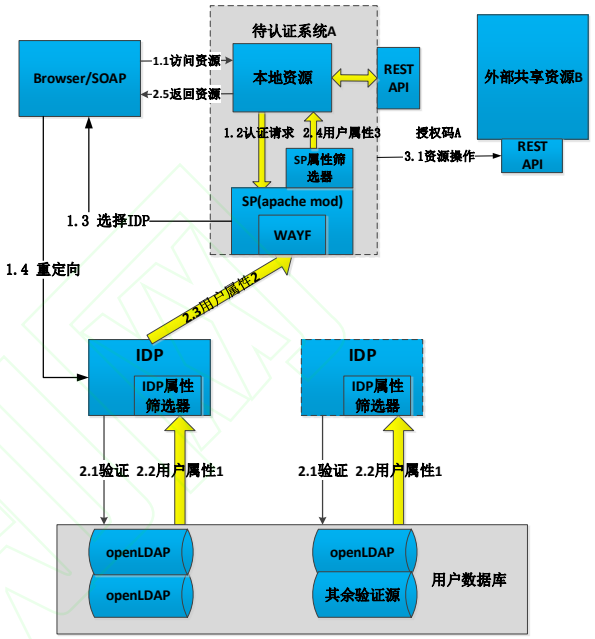

图 1 基于 Shibboleth 的系统框架

Shibboleth 完整实现了 SAML 协议[9]中的配置文件和查询/响应协议，配置文件用于用户与 SP，WAYF 之间通过表单交互并重定向至 IDP，而查询/响应协议用于 SP 端与 IDP 端交互用户属性，包括 AQM(属性查询消息)和 ARM(属性应答消息)，这两种消息都通过 XML 实现，并使用 SOAP 负载。

1.2 Shibboleth 属性发布

Shibboleth 发布的属性包含两类，用户身份属性以及共享资源的授权码。通过用户身份属性的发布，Shibboleth 完成对各系统上帐号的统一认证与本地资源的管理，通过发布授权码 Shibboleth 对所有访问共享资源的请求进行统一授权。

各系统在认证过程中需要的用户信息是不同的，向所有 SP 发布相同的属性是多余且不安全的。IDP 成功验证用户后，收集数据库返回的用户属性，对于某一 SP 筛选出允许向其发布的属性，SP 收到属性后再一次筛选出待认证的系统所需要的属性。通过两层筛选，Shibboleth 能灵活地发布满足待认证方需求且最小化的用户信息。Shibboleth 有灵活的属性筛选策略，通过 xml 文件进行配置，针对用户属性，请求参数等值进行判断，支持基础条件，正则表达式的属性筛选，通过组合几乎能满足所有筛选需求。如以下一段 xml 配置代码将 Affiliation 属性的值限定为“student”或“staff”，且只向 url 为“service.example.edu/shibboleth-sp”的 SP 发布该属性。

```
<AttributeRule attributeID="Affiliation">
```

```
<PolicyRequirementRule
xsi:type="basic:AttributeRequesterString"
vaue="https://service.example.edu/shibboleth-sp"
/>
<PermitValueRule xsi:type="basic:OR">
<basic:Rule
xsi:type="basic:AttributeValueString" val-
ue="student" />
<basic:Rule
xsi:type="basic:AttributeValueString" val-
ue="staff"/>
</PermitValueRule>
</AttributeRule>
```

2 在线实验平台多资源访问认证实例

在线实验平台在 OpenEdX(一个开源的网络公开课程平台[10])上为用户提供实验所需的开发环境(linux 系统),并提供内部 GitLab 存储实验代码,以免去用户前期繁琐的环境配置工作,避免环境差异带来的问题,最快地进行实验。

本文基于上述问题开发了一个在线实验平台,该平台包含三部分:OpenEdX,GitLab(代码存储平台)以及 Docker[11]服务器(提供实验 Docker)。平台用户来自多个学校,用户信息由不同组织管理,用户能够独立访问这三个系统,需要进行统一认证。平台资源包括用户帐号,GitLab 用户仓库以及实验 Docker,用户能在 OpenEdX 上通过插件访问其余两个系统。对于共享资源的需求,OpenEdX 与 Docker 需要访问用户仓库,OpenEdX 同时需要控制 Docker。因此,需要利用图 1 描述的框架实现统一认证,并对这些复杂资源统一管理,授权平台间共享资源。

2.1 系统框架

如图 2 所示,OpenEdX 在线实验平台框架主要包括以下五大部分:

- OpenEdX:用于在线课程的平台。能够直接在 OpenEdX 上访问 GitLab 仓库代码以及实验 Docker。
- GitLab:具有版本控制功能的代码托管平台,用于放置实验代码。携带授权码访问 REST API 可以上传公钥,并在本地通过私钥对仓库进行 git 操作。
- Docker 服务器:为用户分配已配置好实验环境的 Docker,通过网页终端访问。
- XBlock: OpenEdX 上的插件模块,负责与其余系统进行交互。代码编辑,浏览以及 Docker 控制都以 XBlock 插件的形式运行在 OpenEdX 上。
- Shibboleth:用于各平台的单点登录以及资源的统一授权管理,内部结构与图 1 相同,在此省略。

为了对系统进行测试,用户的初始信息从学堂在线获取,验证用户可靠性后注册接口根据获取的信息为用户在 LDAP 中创建账号。OpenEdX 作为在线实验平台的核心,负责初始化用户在其余系统上的资源,包括创建用户仓库,初始化实验代码,上传公钥以及创建 Docker 实验容器。

各系统向 Shibboleth 提交用户的认证请求,由 Shibboleth 完成用户认证。由于 Docker 基于 ssh 认证,直接部署 Shibboleth 较为困难,因此将 Docker 的用户账号直接与 LDAP 数据库连接。Docker 只需要对代码操作,无需对其发布 GitLab 资源授权码,由 OpenEdX 的控制模块在创建 Docker 时负责提供用户私钥即可。代码编辑,浏览插件,以及用户的 Docker 都借助私钥获取用户在 GitLab 上的仓库代码,并在本地进行处理。用户登录 OpenEdX 后可以直接使用 XBlock 访问 GitLab 以及 Docker 资源。

在线实验平台中的三个系统都使用 REST API 接口对外提供资源,通过 HTTP 协议提供的 GET、POST、PUT 和 DELETE 方法,客户端能够实现对资源的获取、创建、修改和删除。借助 Shibboleth 的资源授权以及 REST API 接口,各平台无需任何数据的迁移即能实现平台间的数据共享。

2.2 系统组件

2.2.1 OpenEdX 与 XBlock

OpenEdX 上课程所需的组件以 XBlock 的形式封装,课程开发者能够自行编写 XBlock 插件。在线实验平台的主要功能集中在 OpenEdX 上,OpenEdX 以 XBlock 的形式提供各类插件与其余平台进行交互,用户完成 Shibboleth 认证并登录 OpenEdX 后,XBlock 能够获取 Shibboleth 的授权,以用户身份对 GitLab 仓库以及 Docker 进行操作。OpenEdX 运行在 Nginx 服务器上,需要将认证请求转发至 Apache 中的 SP 模块,SP 完成认证后从该端口返回用户属性至 Nginx。

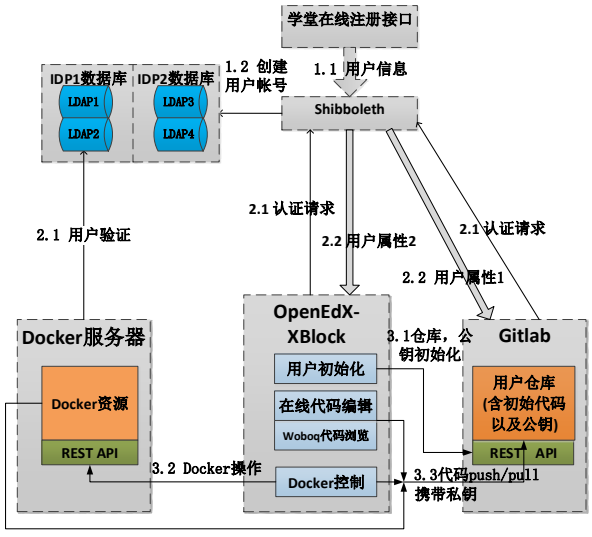

图 2 在线实验平台框架图

2.2.2 Docker

Docker 作为一种虚拟容器技术,相比于 VM 更轻量级,更适合作为实验容器。由于无法在网页端直接访问 Docker(不考

虑间接转发的方式)，Shibboleth 的部署有较大困难，为此本文采取了一种折衷的方式，利用插件式认证模块(nss-pam-ldapd 工具)将 Docker 的 ssh 认证与用户对应组织机构的 OpenLDAP 数据库建立连接，当用户使用 Shibboleth 账号 ssh 连接至 Docker 时，Docker 将自动检测 LDAP 数据库中的用户，若符合则创建账号，供用户登录。这种做法的缺点是 Docker 无法直接获取由 Shibboleth 发布的用户属性，包括授权码，需要由 OpenEdX 在创建 Docker 时提供。

OpenEdX 的 XBlock 插件借助 REST API 接口(携带 Docker 资源的授权码)创建 Docker，可手动删除，否则定时回收。Docker 本身不包含任何用户信息，只通过 DockerFile 配置基础实验环境，XBlock 创建 Docker 时传入用户仓库私钥以及用户基本信息，由 Docker 的启动脚本在本地初始化 GitLab 仓库配置以及仓库代码。为了支持 Docker 的网页端访问，在 DockerFile 中添加了 tty.js(基于 nodejs，通过 WebSocket[12]与 Linux 系统通信)。

2.2.3 GitLab

为存储用户代码，在线实验平台中包含了一个私有化的 Github，即 GitLab。GitLab 对其余系统的资源要求较低，除最基本的用户身份属性外，没有额外要求。OpenEdX 与 Docker 通过私钥访问 GitLab 的用户仓库代码，OpenEdX 还需要通过管理员的授权码获取额外的操作权限，包括创建用户仓库，导入实验代码，以及上传公钥等。GitLab 同样运行在 Nginx 上，SP 的部署与 OpenEdX 相同。

将图 1 中的认证授权框架应用到在线实验平台后，OpenEdX，GitLab，Docker 都不再负责实际的验证，用户账号信息由 IDP 连接的多个 LDAP 管理，只与 Shibboleth 交互，联盟内的系统不会接触到用户的密码信息，SAML 的信息交换协议也能够保证交互的安全性。在复杂资源的管理上，OpenEdX 保存用户的课程相关信息，GitLab 维护用户仓库，Docker 服务器存放用户的实验容器，系统维护各自的资源，通过 Shibboleth 进行统一认证以及外部资源的访问授权，实现了资源维护和认证组织的相互独立。同时 OpenEdX，GitLab，Docker 相互独立，除了 OpenEdX 上的 XBlock 依赖于其余 2 个系统外，只要通过 Shibboleth 验证，每个平台都可以独立使用。

2.3 实验结果

表 1 列出了 Shibboleth 通过发布授权码向各系统授予的访问权限(不考虑系统自身拥有的权限，比如 OpenEdX 对 OpenEdX 的访问权限)。GitLab1 指 Gitlab 的管理员权限，GitLab2 指通过用户私钥获取的对用户仓库的 git 操作权限。

各系统经过 Shibboleth 统一认证后获取发布的授权码。OpenEdX 作为主要的课程平台，为学生创建仓库并上传公钥，Shibboleth 需要向其发布 Gitlab 管理员的授权码，同样，OpenEdX 创建用户的 Docker，需要发布访问 Docker 资源的授权码。Docker 服务器只需要拥有对用户仓库的 git 操作权限，发布用户的私钥即可。GitLab 对其他系统没有访问的需求，只进行统一认证。

访问权限的粒度由提供 REST API 接口的系统决定，与 Shibboleth 无关，例如 GitLab 对管理员以及普通用户有不同的访问接口，因此可以对 Gitlab 的权限进一步划分。授予系统权限的变动只需要修改 IDP 的配置文件。

表 1 Shibboleth 分配的各系统的访问权限

	OpenEdX	*GitLab1*	*GitLab2*	*Docker*
OpenEdX	N	Y	Y	Y
GitLab	N	N	N	N
Docker	N	N	Y	N

3 结束语

传统的单点登录无法满足多组织用户的认证需求，为此需要使用联盟认证框架，而对于联盟中各类无法用属性字段存储的复杂资源，仍需要一种较好的方式进行授权。本文引入 Shibboleth 对包含多组织用户的系统进行统一认证，在此基础上，为了实现复杂资源的授权，令资源所在系统对外提供需验证的 REST API 访问接口，并借助 Shibboleth 的属性发布策略向认证成功的系统发布用于访问共享资源的授权码。用户所属的 IDP 通过对属性的筛选，有权决定是否将用户的资源授权码发布给某一 SP，从而灵活地控制资源的共享。为了进行验证，本文使用该方案对在线实验平台上的多组织用户进行了统一认证，并针对实验平台的资源共享需求，向各个系统发布最小化的用户属性以及授权码，成功对用户的 Docker 以及 GitLab 仓库的访问进行授权管理。

后续的工作主要包括如下几点。第一，目前在线实验平台的 Docker 认证直接与 LDAP 数据库连接，而 Shibboleth 支持 SOAP 访问，可以通过 SOAP 实现 ssh 方式的 Shibboleth 认证，但目前尚不成熟，需要进一步完善。第二是认证源类型的扩展，Shibboleth 为认证源提供了较为灵活的支持，支持 Kerberos，也允许自行编写代码连接 IDP 之外的其他认证系统，为与不同组织的现有数据库连接提供了支持。第三，目前 Shibboleth 对于授权码的安全性保障在于 IDP 对于系统 A 的信任，在本地的属性发布策略中向系统 A 发布资源的授权码，在这种情况下存在授权码被系统 A 泄露或滥用的风险，因此需要对授权码的有效时间进行限制，考虑授权码失效，重新获取的时机以及由哪一方负责对授权码更新。

参考文献:

[1] Pautasso C, Zimmermann O, Leymann F. RESTful Web services versus 'Big' Web services: making the right architectural decision[C]//Proc of the 17th International Conference on World Wide Web. New York: ACM Press, 2008: 805-814.

[2] Documentum content server central authentication service(CAS)SSO[EB/OL]. (2013-11)[2016-2-21]. http: //uk. emc. com/collateral/white-papers/h12009-wp-pdf-documentum-content-server-central-authentication-service-sso-a-detailed-review. pdf.

[3] Recordon D, Reed D. OpenID 2. 0: a platform for user-centric identity

management[C]//Proc of the 2nd ACM Workshop on Digital Identity Management. 2006: 11-16.

[4] Feng Yang, Manoharan S. A security analysis of the OAuth protocol [C]//Proc of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing. 2013: 27-29.

[5] Maler E, Reed D. The Venn of identity: options and issues in federated identity management[J]. IEEE Security and Privacy, 2008: 16-23.

[6] Morgan R L, Cantor S , Carmody S, *et al*. Federated security: the shibboleth approach[J]. **Educause Quarterly**, 2004, **27**(4): 12-17.

[7] Ngo L , Apon A . Using Shibboleth for authorization and authentication to the subversion version control repository system[C]//Proc of International Conference on Information Technology: New Generations. 2007: 760- 765.

[8] Wang X , Schulzrinne H , Kandlur D , *et al*. Measurement and analysis of LDAP performance[J]. **ACM SIGMETRICS Performance Evaluation Review**, 2008, **16**(1): 232-243.

[9] Armando A, Carbone R, Compagna L, *et al*. Formal analysis of SAML 2. 0 Web browser single sign-on: breaking the SAML-based single sign-on for Google Apps[C]//Proc of FMSE. 2008: 1-10.

[10] MOOCs: massive open online coures[EB/OL]. [2016-2-21]. http: //www. eua. be/Libraries/publication/EUA_Occasion-al_papers_MOOCs.

[11] Charles A. Docker[J]. IEEE Software, 2015, **32**(3): 102-c3.

[12] Fette I , Melnikov A . RFC 6455, the WebSocket protocol[S]. 2011.