# An OAuth2.0-Based Unified Authentication System for Secure Services in the Smart Campus Environment

Baozhong Gao [1], Fangai Liu [1,*], Shouyan Du [2,*] and Fansheng Meng [1]

[1] Shandong Normal University, Shandong, China
[2] Computer Network Information Center, Chinese Academy of Science, Beijing, China
dushouyan@hotmail.com

**Abstract.** Based on the construction of Shandong Normal University's smart authentication system, this paper researches the key technologies of Open Authorization(OAuth) protocol, which allows secure authorization in a simple and standardized way from third-party applications accessing online services. Through the analysis of OAuth2.0 standard and the open API details between different applications, and concrete implementation procedure of the smart campus authentication platform, this paper summarizes the research methods of building the smart campus application system with existing educational resources in cloud computing environment. Through the conducting of security experiments and theoretical analysis, this system has been proved to run stably and credibly, flexible, easy to integrate with existing smart campus services, and efficiently improve the security and reliability of campus data acquisition. Also, our work provides a universal reference and significance to the authentication system construction of the smart campus.

**Keywords:** OAuth2.0, authentication and authorization, open API, cloud security, smart campus, open platform

## 1 Introduction

The web was proposed as a hypertext system for sharing data and information among scientists [1], and has grown into an unparalleled platform on which to develop distributed information systems [2].

Cloud computing was defined [24] by the US National Institute of Standards and Technology (NIST). They defined a cloud computing [3] as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. "The data stored in the cloud needs to be confidential, preserving integrity and available" [4]. All cloud computing have to provide a secure way, including personal identification, authorization, confidentiality, and the availability of a certain level in order to cope with the security problem of private information exposed. On cloud-computing platforms, most of the different departments on smart campus have their own independent information systems with separate identity and authentication

[9]. This has created a security issue with the uniformity of information between the different systems [5].

Google is investing in authentication using two-step verification via one-time passwords and public-key-based technology to achieve stronger user and device identification. One disadvantage of this opt in protection is that attackers who have gained access to Google accounts can enable the service with their own mobile number to impede the user restoring their account [6]. From this requirement, the Open Authorization protocol (OAuth) "was introduced as a secure and efficient method of authenticating access to cloud data APIs for authorizing third-party applications without disclosing the user's access credentials" [9].

Based on the current situation at Shandong Normal University, we use OAuth2.0 protocol to design and implement the smart campus authentication system. With this system, users do not have to provide third-party authentication credentials, in the case of using a third-party proxy to log on to a target server. Authorization allows a third-party to obtain the specified information, with a simple, convenient, safe and reliable authentication process. The release of data can be controlled, and there are also many other advantages with the system. Under the support of the unified authentication, all the third-party platforms accessed by the smart campus application are shore an interconnected entity. This further improves the security and reliability of the campus data acquisition. In addition, this promotes better campus information construction, which better serves the school affairs, which includes improvement of the life of teachers and students.

## 2 Related Works

### 2.1 OAuth2.0 Protocol

OAuth is an open license agreement, which provides a secure and reliable framework for third party applications to access HTTP services with certain authority and limitations [10]. It seeks to resolve the shortcomings of propriety authentication protocols by creating a universal and interoperable authorization mechanism between services [7]. Our smart campus system opens OAuth2.0 interface to all teachers and students of Shandong Normal University. The big advantage of using OAuth for connecting devices is that the process is well understood, and numerous libraries and API platforms provide support for API providers and client developers alike [11]. OAuth adds an additional role: the Authentication Service (AS), which is invoked by the SP to verify the identity of a user in order to grant access token [13]. OpenID Connect is an authentication protocol that is based on OAuth and adds an identity layer for applications that need it [12]. There are three main modes of participation entities in OAuth2.0:

- RO (Resource Owner): The entity that grants access control to protected resources, which is equivalent to the user defined in OAuth1.0.
- RS (Resource Server): The server that stores data resources of users and provides access to the protected resources requested.
- AS (Authorization Server): The service that verifies the identity of the resource owner and obtains the authorization, and issues the Access Token.

- UA (User-Agent): It’s suitable for all non-server-side client, like Chrome, Firefox, and Safari web browsers.
- AT (Access Token): A piece of data the authorization server created that lets the client request access from the resource server [8].
- AC (Authorization Code): A piece of data that the authorization server can check, used during the transaction’s request stage [8].

The standard authorization process of OAuth2.0 protocol with specified API is generally divided into six steps:

1) User authorization: The client sends an authorization request includes the type, Client Id, and Redirect URL to RO, which begins the user authorization process.

2) Return authorization code: RO agrees to the authorization, and returns an AC to UA.

3) Request access token: After the user completes the authorization, the client uses AC and Redirection URL to request AT from AS.

4) Return access token: AS authenticates client, and validates AC. If the client user's private certificate and access permissions are verified and qualified, the client will gain access to the AT from AS.

5) Request protected resources: The client requests for the protected resources to RS through an AT.

6) Acquire user’s information: After the RS verify the validity of the AT, in response to this resource request, the client can obtain the permission to acquire protected resource using the specified information interface.

### 2.2 Open API Access Control with Oauth

An API is whatever the designer of the software module has specified it to be in the published API document, without any judgment regarding what functionality the API offers vis-à-vis the functionality it ought to offer [14]. The information owned by online services is made available to third-party applications in the form of public Application Programming Interfaces (APIs), typically using HTTP [15] as communication protocol and relying on the RE presentational State Transfer (REST) architectural style [12]. Our smart campus authentication system API works as following:

1) Prerequisite description

1. Our smart campus application system has opened the use of the Open API permissions. From the interface list of the API, it’s known that some interfaces are completely open, and some interfaces need to submit applications in advance to gain access.

2. The resources to be accessed are accessible by the user's authorization. While the site calls the Open API to read and write information about an OpenID (user), the user must have already authorized the OpenID on smart campus application system.

3. Access Token has been successfully obtained, and is within the validity period.

2) Call the Open API

Web sites need to send requests to a specific Open API interface to access or modify user data. When all Open API are invoked, in addition to the private parameters of each

DOI: 10.1007/978-3-319-93713-7_73

interface, all Open API need to pass generic parameters based on the OAuth2.0 protocol as following shown in Table 1 [25]:

**Table 1.** OAuth 2.0 generic parameters

| Parameters | Description |
|---|---|
| access_token | Access_Token can be obtained by using Authorization_Code and has a validity period of 3 months |
| oauth_consumer_key | The AppID which assigned to the app after the third-party application is successfully signed in |
| openid | The user ID, corresponding with Smart SDNU account |

We take the library borrowing information acquisition interface as an example, the interface authority is read_personal_library, and it aims to return the number of borrowed items and the collection of borrowed information of currently logged-in user's library in the specified period.

Request: GET http://i.sdnu.edu.cn/oauth/rest/library/getborrowlist

Request parameters are as shown in Table 2:

**Table 2.** /library/getborrowlist request parameters

| Field | Required | Type | Description |
|---|---|---|---|
| start | false | string | Start date (formal as yyyy-MM-dd HH:mm:ss) |
| end | false | string | End date (formal as above) |
| count | false | int32 | Return number (1-50, default as 10) |
| index | false | int32 | Return page numbers (default as 1) |

After successful return, we can get the user borrowing data shown as following:

```
[
  {
    "identityNumber":"2013001001",
    "bookName":"Title",
    "borrowDate":"2014-01-01T08:00:00",
    "mustReturnDate":"2014-01-31T08:00:00",
    "isRenew":false
  },
  {
    If there are any other books, the same as above.
  }
]
```

In our OAuth2.0-based smart campus authentication system, we can also get needed information through personal basic information interface, personal campus card information interface, personal library information interface, lost and found information interface, school public information interface, school news information interface, and weather information interface. Through three years' analysis chart of interface access times, we know that the three most frequently accessed interfaces are the library, card and user personal information API. During the period 2014-2016, the card and the user's personal information API vary more stable, however, the number of library API visits in 2015 increased significantly compared with 2014, and then stabilized, indicating that the construction of the new library of our school effectively improve the learning enthusiasm of teachers and students in our school, which provides a remarkable promotion significance to the study style construction of our school.

## 3 OAuth2.0-based Smart Campus Authorization System

### 3.1 Overall Structure

In order to comply with the security and privacy requirements, based on OAuth2.0 protocol, we use APIs to put together highly distributed applications with many interconnected dependencies [16], and propose the smart campus open authentication system, which adopts the hierarchical architecture design concept. Cloud-based services, the social Web, and rapidly expanding mobile platforms will depend on identity management to provide a seamless user experience [17]. Moreover, the proposed architecture aims at minimizing the effort required by service developers to secure their services by providing a standard, configurable, and highly interoperable authorization framework [13].

The overall architecture of the system including three following layers:

- Infrastructure layer: The main work of Infrastructure Layer is to collect and process various information timely and accurately through RFID identification, infrared sensors, video capture, GPS and other technologies and equipment on the campus information collection and dynamic monitoring, and send the information collected from the hardware device to Data Layer in security.
- Application service layer: Application Service Layer's main work is to effectively integrate and management of various information, and achieve unified management of information. Based on the existing management systems, such as financial management system, student management system, staff management system, scientific research management system, equipment management system, logistics management system, through the use of cloud computing and cloud storage technologies to provide a unified management platform, which facilitates third-party applications development, the application service architecture is shown in Figure 1.

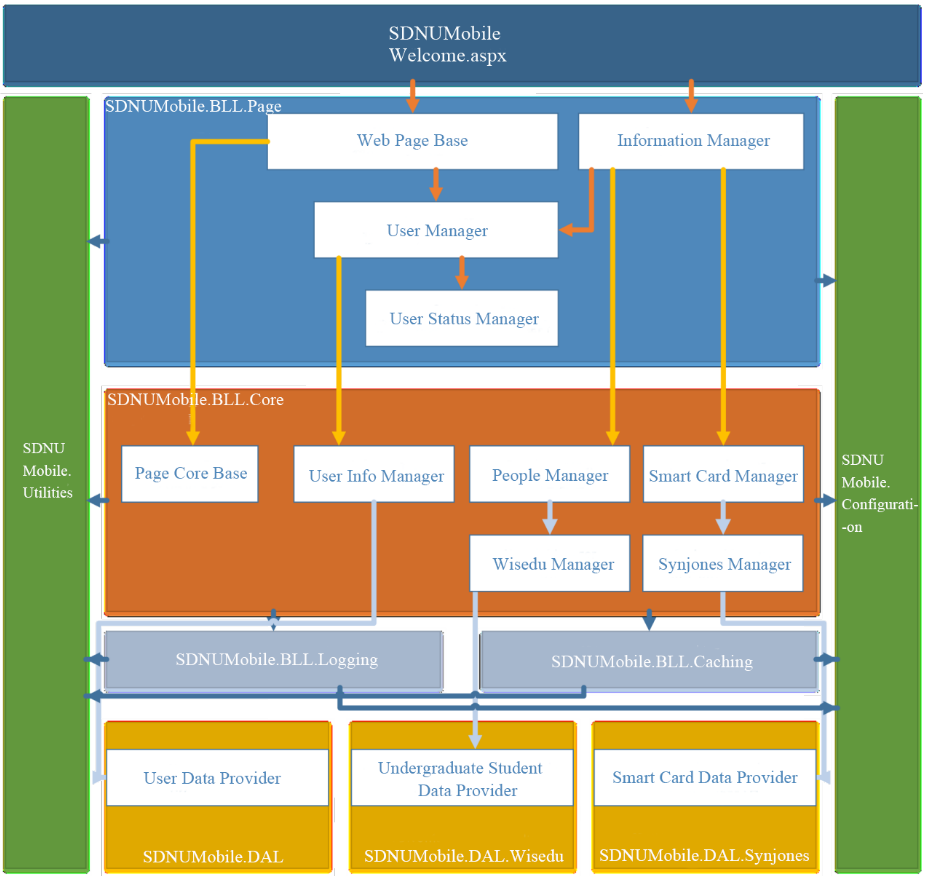


**Fig. 1.** Smart campus application service architecture diagram

- Information provision layer: The main work of information provision layer is to provide teachers and students with specific and effective service platform. In this platform, in accordance with the unified data specification standards, through the identity of the open platform of the school, teachers and students' information, users can use the shared platform of teaching resources and research resources to form an interconnected shared entity.

### 3.2 System Workflow

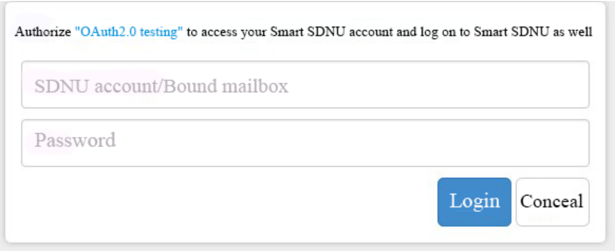
1) Third-party application access to temporary token process:

a) Attach the request information to the request packet: Attach the client credentials, the subsequent callback address, and the requested control information (including the value of the replay prevention single value and the timestamp of the request) to the request packet in accordance with authentication system requirements.

b) Request parameter signature: The HTTP request mode, HTTP protocol request packet URI, the specified request key value pairs arranged as the base string, then, through the HMAC-SHA1 or RSA-SHA1 algorithm to generate messages digest (signature), and the key that participates in the signature algorithm is the key that corresponds to the client credentials.

c) Encapsulation: Add the signature to the HTTP request header, and finally encapsulate it as a legitimate request packet to obtain temporary token.

d) Verify the legitimacy of the request: The third-party application sends to the OAuth2.0-based authentication system a request packet for temporary token through the client credentials. The first step of the authorization procedure is to check the validity of request packet when it receives the request, and then compare with the signature in the request body. If they are exactly the same, the request is identified as legitimate and credible. Hence, the system generates temporary token and the corresponding key, and returns to third-party application, the user login interface as shown in Figure 2.



Authorize "OAuth2.0 testing" to access your Smart SDNU account and log on to Smart SDNU as well

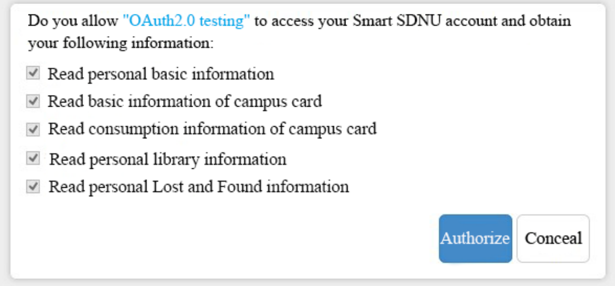
**Fig. 2.** User login interface

2) Third-party user authorization application process:

a) Temporary token authentication page: Third-party application encapsulates the request packet contains callback address through temporary token, and guide the user to authorize authentication. Meanwhile, the page will jump to authentication information page with temporary token, as shown in Figure 3, in which the user credentials of the authentication system can be avoided for public to third-party applications.

b) The user agrees or denies authorization: If the user licenses, authentication system will activate the temporary token to access token, and jumps to the callback address that has previously passed to the authentication system. Then, the third-party application exchanges the temporary token for access token by triggering the callback event in the address. The corresponding temporary token is cleared if the user rejects the authorization. Therefore, the third-party application will be noticed that user is denied authorization, and the authentication system will not issue the access token.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_73


Do you allow "OAuth2.0 testing" to access your Smart SDNU account and obtain your following information:

☑ Read personal basic information
☑ Read basic information of campus card
☑ Read consumption information of campus card
☑ Read personal library information
☑ Read personal Lost and Found information

**Fig. 3.** User authorization interface

3) User information acquisition process:

a) Third-party application request: After the third-party application obtains the access token and the corresponding keys, make a request with request signature which ensures that the request is credible. In order to get the user's protected information, the keys that participate in the signature algorithm must contain not only the keys that match the client's credentials, but also the keys that match the token's credentials.

b) Server authentication: While server accepts the request for user information, it’s need to verify a series of verification process as whether the request packet is legitimate, whether the requested web interface exists, whether the third-party applications have the right to call the interface, and whether the called web interface parameters are legal. After all those process passed, it can implement the business process within the server to obtain the corresponding protected resources. Among them, the protected resources are listed in the user license authority of the project.

## 4 System Security Testing

In order to demonstrate the effectiveness and performance of the proposed authorization system, we have conducted security experimental tests as design and injection security aspects. The operation flow is the following:

### 4.1 Security Testing of Design

1) CSRF attack security testing:

Take the card data acquisition interface as an example. It obtains data normally through the website, and load JSON data from js. Then, visit http//i.sdnu.edu.cn/card/get/info interface, and check the cookie login status as well as return JSON data.

After completed the preparing work, we begin commence a CSRF attack. Try to access the web page as http//i.sdnu.edu.cn/card/get/info under the same cookie condition. If the background check request header does not contain option of X-Requested-With, it will return an error as following to prevent data leakage caused by the attack.

Error: {"status": "error", "result": "Invalid_Referer"}

DOI: 10.1007/978-3-319-93713-7_73

If there is an attack from third-party sites, cause the site has been prevented from cross-border attacks, it can't obtain the data as well, and will return an error as: No 'Access-Control-Allow-Origin' header is present on the requested resource.

2) XSS attack security testing:

We process this test under the condition that there is a request tries to get data from a third-party site.

Request:>$.getJSON("http://i.sdnu.edu.cn/card/get/info?0.3132845529366785");

Response: XMLHttpRequest cannot load http://i.sdnu.edu.cn/card/get/info?Link?ur1=f6LtFstZ1zpQAAbp9HC4eZtRcoFq-07mpvRYEnDQ0VcSWIm5I9qnd6HM1hhiv7Xmh2008Nd6vExMW9krveVWL:10.31324845529366785.

No 'Access-Control-Allow-Origin' header is present on the requested resource. Therefore, origin website 'http://baike.baidu.com' is not allowed to access.

It proves that the web site prevents cross-domain requests, thus, XSS attacks are unable to obtain needed data, which helps reducing the risk of user data leakage.

### 4.2 Security Testing of Injection

1) Scope parameters injection testing:

We process this test under the condition that there is a process attempts to inject Scope parameters when requesting user authorization.

Request:https://i.sdnu.edu.cn/oauth/authorize_code?response_type=code&scope='or1=1'&client_id=00000000000000000000000000000000&oauth_version=2.0

Response: Client no permission.

The reason is that when scope in the background checking the call, there is a middle layer of the cache makes it not directly connected to the database. The injected string is simply regarded as a string rather than SQL statement processing, and the injection process is failed.

2) Camouflage the client testing

1. User login select authorization

The attacker knows the client ID and tries to fake the client to defraud the user authorization in order to get needed data, and go on the process of user login and user authorization, the needed parameters are as shown in Table 3.

**Table 3.** Table captions should be placed above the tables.

| Parameters | Description |
|---|---|
| response_type | Designated as "code" in the first step |
| client_id | The certificate ID issued by the server to the client |
| redirect_uri | The redirection URI needed to return Authorize Code |
| scope | Scope of authority for this certification application |

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_73

| | |
|---|---|
| oauth_version | Requested authentication version (2.0) |

Request:https://i.sdnu.edu.cn/oauth/authorize_code?response_type=code&scope=BasicAuth,None&client_id=00000000000000000000000000000000&oauth_version=2.0

2. Return authorize code

Use the default callback address: https://i.sdnu.edu.cn/oauth/?authorize_code=YoCRIJWg932Y1NioNeO2uleyHtVN4Ps5

3. Use the authorize code to exchange for access token

The attacker doesn't know the Client Secret while in the process of using the authorization code in exchange for Token. Needed parameters are as shown in Table 4.

**Table 4.** Authorization parameters

| Parameters | Description |
|---|---|
| grant_type | Designated as “authorization_code” in this step |
| authorize_code | The authorization code obtained in the previous step |
| client_id | The client ID |

Request:https://i.sdnu.edu.cn/oauth2/access_token?grant_type=authorization_code&authorize_code=YoCRIJWg932Y1NioNeO2uleyHtVN4Ps5&client_id=00000000000000000000000000000000

Response: Client no permission.

It proves that the third-party application should ensure that the process of accessing to Access Token should be carried out in the server, which will not be crawled to Secret. So that by implementing this system we can provide more security of data and also provide efficient user authentication [18].

## 5 Performance Analysis

After a period of the OAuth2.0-based system security on-line testing, we find that the Token within the validity period is stored in the cache database as expected. From the process of user Authentication to obtain Token, it's certified that the whole operation follows OAuth2.0 protocol, as well as the https protocol ensures the security of communication process, on the whole, the smart campus unified authentication system is running stable. In this paper, we mainly enhanced the smart campus authentication system performance include following four aspects.

### 5.1 Basic user data management

The main characteristics of user authentication are collected through a unified data specification standard. This forms a basic database including teacher information, student information, school information, and class information, which helps to achieve the independent management and verification of users between different applications. The database is used for both customization and authentication [19]. Then, the third-party application access providers can obtain relevant information through the cross-platform, and compatible interface with permission [20]. It facilitates in solving problems of user data synchronization, and updating between application systems.

### 5.2 Security issues

OAuth2.0 is a protocol with authorization function to control and manage access to web services [21]. In the process of user authentication, in OAuth2.0 protocol, the user does not have to disclose user name, password, and other information to a third-party platform. Authorized HTTP communication uses digital signatures, and access tokens to replace user information. This helps to avoid leakage of sensitive user information. In user center, the user information is transmitted to the application system through the message bus with unified account management and operation no longer without an individual's explicit consent [22]. In addition, OAuth2.0 can be combined with PKI (public key infrastructure) to enrich the authentication function. It solves the problem of identity authentication, authorization problems, and user resource security under the open architecture of a cloud environment.

### 5.3 Cross-platform resource sharing

The resource sharing platform stores only the address information of the resource and the resource token. It does not store the resource itself. The OAuth protocol has been designed to handle computation and storage overhead while providing a standard and simple authorization mechanism for resource access by third-party applications [13]. Through the security API support protocol, access can be facilitated with other applications. This is convenient in providing relevant data for their own applications, and will enhance the usability of the smart campus system. In this aspect, it solves the problem of resource-dispersants and duplicated storage.

### 5.4 Application management

Through the combination of OAuth2.0 protocol and the authority control system, this system can easily access classification of applications. The administrator can allow or deny application access according to the actual situation [23]. If the application only needs simple resources, such as a user's basic information, it can be completed by the application developer self-application, and administrators post-audit. If the requested information contains sensitive resources, the developer needs to submit the main functions of the application, and the scenes that need these user resources. This will increase the application access flexibility, and solve the problem of system management cost in the process of user authentication of the smart campus system.

## 6 Conclusion

This paper, based on an actual situation at Shandong Normal University, proposes a smart campus unified authentication system using OAuth2.0 protocol to achieve third-party applications access and authorization. This allows users who use third-party campus applications, no longer need to provide authentication information. OAuth2.0 is the authentication method of choice by our smart campus to protect users' APIs and federate identification across domains [24]. So, it is possible for users to have safely access to the smart campus system based on cloud environment. This saves the system development costs and development cycle, which will help in successfully accessing the smart campus application system security; in addition to unified management of user information [20]. Through the testing process, it was indicated that a secure OAuth2.0 environment can be built when implemented correctly. It can also achieve the purpose of the initial design and campus information construction. As a result, this paper has a universal reference and significance to the authentication and authorization system construction of the smart campus.

## 7 Acknowledgment

This research is financially supported by the National Natural Science Foundation of China (90612003, 61602282, No.61572301), the Natural Science Foundation of Shandong province ((No. ZR2013FM008, No. ZR2016FP07), the Open Research Fund from Shandong provincial Key Laboratory of Computer Network, Grant No.: SDKLCN-2016-01, the Postdoctoral Science Foundation of China (2016M602181), and Science and technology development projects of Shandong province (2011GGH20123).

### References

1. Berners-Lee, T. J. The world-wide web. Computer Networks and ISDN Systems, 25(4), 454-459(1992).
2. Kopecky, J., Fremantle, P., Boakes, R. & Dr. A History and Future of Web APIs. Information Technology, 56(3), 90-97(2014).
3. Mell, P. & Grance, T. The nist definition of cloud computing. Communications of the Acm, 53(6), 50(2010).
4. Aldossary, S. & Allen, W. Data Security, Privacy, Availability and Integrity in Cloud Computing: Issues and Current Solutions. IJACSA, 7(4), 485-498(2016).
5. Hyeonseung, K., Chunsik & P. Cloud computing and personal authentication service. J. Korea Inst. Inf. Secur. Cryptol, 20(2), 11–19(2010).
6. Grosse, E. & Upadhyay, M. Authentication at Scale. IEEE Security Privacy, Jan/Feb, 15-22(2013).
7. Ferry, E., Raw, J. O. & Curran, K. Security Evalution of the OAuth 2.0 Framework. Information and Computer Security, 23(1), 73-101(2015).

8. Leiba, B. OAuth Web Authorization Protocol. IEEE INTERNET COMPUTING, Jan/Feb, 74-77(2012).
9. Shehab, M. & Marouf, S. Recommendation Models for Open Authorization. IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, 9, 1-13(2012).
10. Huang, R. Y. Smart campus construction program and Implementation. Guangdong: South China University of Technology publishing house(2014).
11. Phillip J. W. API Access Control with OAuth: Coordinating interactions with the Internet of Things. IEEE Consumer Electronics Magazine, 4(3), 52-58(2015).
12. Cirani, S., Picone, M., Gonizzi, P., Veltri, L. & Ferrari, G. IoT-OAS: An OAuth-Based Authorization Service Architecture for Secure Services in IoT Scenarios. IEEE SENSORS JOURNAL, 15(2), 1224-1234(2015).
13. Sakimura, N., Bradley, J., Jones, M., Medeiros, B. D. & Mortimore, C. OpenID Connect Core 1.0 Incorporating Errata. http://openid.net/specs/openid-connect-core-1_0.html(2014).
14. Rama, G. M. & Kak, A. Some structural measures of API usability. Softw. Pract. Exper, 45(1), 75-110(2015).
15. OpenAPI call description_OAuth2.0. from http://wiki.open.qq.com/wiki/website/Open-API%E8%B0%83%E7%94%A8%E8%AF%B4%E6%98%8E_OAuth2.0.
16. Garber, L. The Lowly API Is Ready to Step Front and Center. COMPUTER, 13, 14-17(2013).
17. Lynch, L. Inside the Identity Management Game. IEEE T SERV COMPUTING, 11, 78-82(2011).
18. Jami, S. & Rao, K. S. Providing Multi User Authentication and Anonymous Data sharing in cloud computing. IJETT, 31(1), 50-53(2016).
19. Chess, B. & Arkin, B. Integrating User Customization and Authentication: The Identity Crisis. IEEE Security & Privacy, 82-85 (2012).
20. Li, X. & Wang, L. Digital campus unified authentication platform research and application. JIANGXI EDUCATION, 7-8(2016).
21. Choi, J., Kim, J., Lee, D. K., Jang, K. S. & Kim, D. J. The OAuth2.0 Web Authorization Protocol for the Internet Addiction Bioinformatics (IABio) Database. Genomics & Informatics, 14(1), 20-28 (2016).
22. Zhang, M., Shi, J. Q, Ren, E. & Song, J. OAuth2.0 in the integration of management platform for the sharing of resources research. China Chemical Trade, 182-183 (2015).
23. Wang, X. S. Du, J. B & Wang, Z. Research and Practice of OAuth Authorization System in the Information Environment of Universities. China Higher Education information Academy, Nov(2014).
24. Leiba & B. OAuth Web Authorization Protocol. IEEE INTERNET COMPUTING, 16(1), 74-77 (2012).
25. OpenID Connect Core 1.0 Incorporating Errata Set 1, Vol. 8, [Online]. Available: http://openid.net/specs/openid-connect-core-1_0.html(2014).