Authentication and Authorization for RESTful WEB API in Scientific Computing Environment

Rongqiang Cao [1]
Computer Network Information Center, Chinese Academy of Sciences
P.O. Box 349，Bejing 100190, China
E-mail: `caorq@sccas.cn`

Yangang Wang
Computer Network Information Center, Chinese Academy of Sciences
P.O. Box 349，Bejing 100190, China
E-mail: `wangyg@sccas.cn`

Xuebin Chi
Computer Network Information Center, Chinese Academy of Sciences
P.O. Box 349，Bejing 100190, China
E-mail: `chi@cnic.cn`

Rong he
Computer Network Information Center, Chinese Academy of Sciences
P.O. Box 349，Bejing 100190, China
E-mail: `herong@sccas.cn`

Shasha Lu
Computer Network Information Center, Chinese Academy of Sciences
P.O. Box 349，Bejing 100190, China
E-mail: `lusha721@sccas.cn`

Xiaoning Wang
Computer Network Information Center, Chinese Academy of Sciences
P.O. Box 349，Bejing 100190, China
E-mail: `wxn@sccas.cn`

[1]Speaker

Abstract:

Through grid computing and cloud computing technologies, SCE (Scientific Computing Environment) integrates massive computing, storage and application resources. These resources are packaged as easy-to-use open APIs in RESTful web services. These APIs are used to develop terminal software for multi-disciplinary and cross-scenario. Around authentication and authorization issues among users, several services for open APIs, authentication and authorization are proposed and implemented in this paper. The proposed services provide single sign-on for several WEB communities by SCE accounts, support users to authorize terminal software that could access massive resources and personal private data in proxy mode, and also help administrators determine which open APIs a client could access. Atop the proposed services, all related people consisting of users, developers and administrators, no longer need to worry about and solve complex problems with authentication and authorization. What they need to pay much attention on are specific business logics and application scenarios for their interested areas. The proposed services have been applied to general computing portal, operation and management portal in national high-performance computing environment, and also WEB communities for computational chemistry, bioinformatics, etc. These examples show that the proposed services have achieved good performance and user experience.

International Symposium on Grids & Clouds 2019, ISGC2019
31st March - 5th April, 2019
Academia Sinica, Taipei, Taiwan

1. Introduction

Through grid computing and cloud computing technologies， CNGrid [1] and SCE (Scientific Computing Environment, previously also known as ScGrid) [2] integrate massive computing, storage and application resources. As a general-purpose computing platform started from 2006 in CAS (Chinese Academy of Sciences), SCE is designed as a pyramidal structure. At present, the top layer is a centralized massive computing environment named ERA which is a heterogeneous cluster, a 2.3 petaflops supercomputer for computing and data service platform. The middle layer is distributed among China, in which 9 branch centres are selected and connected. In addition, SCE has 19 sub-branch centres and 11 GPU centres in the bottom layer. Until now, SCE has integrated more than 200 petaflops computing capability and 200 petabytes storage capability [3].

To leverage massive computing capability and hundreds of applications in SCE, a set of easy-to-use RESTful web API has provided for developers since 2013 [4]. Based on these APIs, the developers could pay much attention on designing and implementing business logics of different kinds of scenarios occurred in interesting disciplines and fields. However, computing capability and user's data are protected resources which should be limited to access by certain authentication and authorization service. This paper proposed and implemented simple RESTful web APIs to solve authentication and authorization issues among users, terminal clients and open web APIs in SCE. The remains of the paper are organized as follows. Section 2 discussed background and issues to be solved in this paper. Section 3 described the architecture of simple authentication and authorization service, including design principles, modules and services. Section 4 discussed simple authentication services and multi-granularity authorization services. Several typical use cases were described in section 5 to show availability and simplicity. Section 6 described related works. And section 7 summarized this paper and pointed out future work.

2. Background and issues

In the process of providing computing services for users and developers, we found that SCEAPI-REST [4] reduced the workload of designing and implementing clients such as WEB communities and mobile apps, but also introduced authentication and authorization issues among users, computing resources and clients themselves. Through deeply analysis and discussion, there were 3 problems to be solved as follows.

(1). How an account of SCE securely could login into a client developed by third-party without disclosing sensitive credential information such as password? SCE provides each user an account, ant a user can use that account login into different clients. Different authentication methods were required to satisfy several typical clients such as web gateways, scripts and desktop applications in development and production environments. In addition, it was necessary to provide users single-sign-on service in order to avoid inputting account credentials repeatedly in different clients.

(2). How a user of SCE could authorize a client to access computing resources and private data in limited scope and block illegal actions beyond the approved scope? A user should authorize a client to perform some actions on behalf of him based on Auth2 protocol. It was necessary to implement some component for supporting simple and standard OAuth2 authorization workflow.

(3). How an administrator could manage privileges of open APIs and assign different permissions to any client dynamically? SCEAPI-REST provides different kinds of services such as jobs, data, applications computing resources, in form of easy-to-use RESTful web APIs. Therefore, certain service or tools were needed to implement authorization based on resource and role models.

3. Microservice architecture of Simple authentication and authorization

Based on SCEAPI-REST, microservice architecture was designed to provide simple authentication (AuthN) and authorization (AuthZ) services according to low coupling and high cohesion design principles. In the process of designing, we paid much attention on determining core functions of each microservice, isolating different microservice, and maintaining API compatibility between backward and forward versions. The microservice architecture was shown in figure 1. In this section, each layer and microservice were discussed as follows.

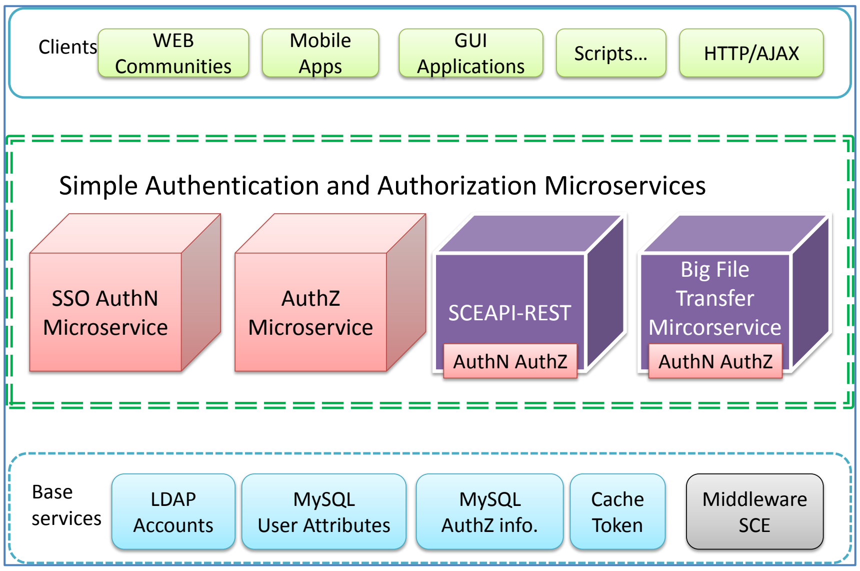

Figure 1 Microservice architecture of Simple authentication and authorization

3.1 Client layer

Clients are softwares that provide command lines, graphical desktop applications, and web gateways for users and administrators to perform lots of actions in SCE. But it is more accurate to say that clients are certain kinds of terminal softwares developed on SCEAPI-REST by developers coming from many disciplines and research areas. Due to the cross-platform and cross-language features of SCEAPI-REST, developers can choose a familiar and extensible integrated development environment (IDE) to create an application community that reflects individual needs, as long as the IDE supports HTTP protocol and Ajax asynchronous request-response mode. Based on the authentication and authorization services proposed in this paper, a client can perform authentication by different methods of username and password strings, SSO and so on. Further, a client can login to SCEAPI-REST in backend in different authentication methods on behalf of a user who has authorized the specific client. Then the client can access massive computing resources and user's private data, perform actions emitted by a user.

3.2 Microservices layer

This layer is composed of 4 mircoservices as shown in the middle of figure 1. For developers, these microservices provide RESTful APIs that supports cross-platform and cross-language features, or Software Development Kits (SDKs) in popular programming languages such as Java, Python and so on. Based on the RESTful APIs or SDKs, developers can spend less time for how to implement authentication and authorization on their own clients and more time on business logics originated from the specific scenarios. For common users, these mircoservices provide standard OAuth2 workflow and several kinds of methods to authenticate or authorize a client in order to avoid disclosing sensitive information and inputting credentials repeatedly in different clients. For administrators, these mircoservices provide simple interface to define roles, classify APIs and mange authorization scopes. Each mircoservices is discussed in detail as follows.

3.2.1 SSO Authentication microservice

As shown in figure2, the SSO Authentication microservice provides general authentication services for different clients by means of SDKs and RESTful web APIs, which is described in detail in section 4.1. This microservice palyed as a gateway to all different clients in SCE. All users would be redirected to this microservice, complete authentication workflow in a security environment, and then access at least one or more clients in which they are interested.

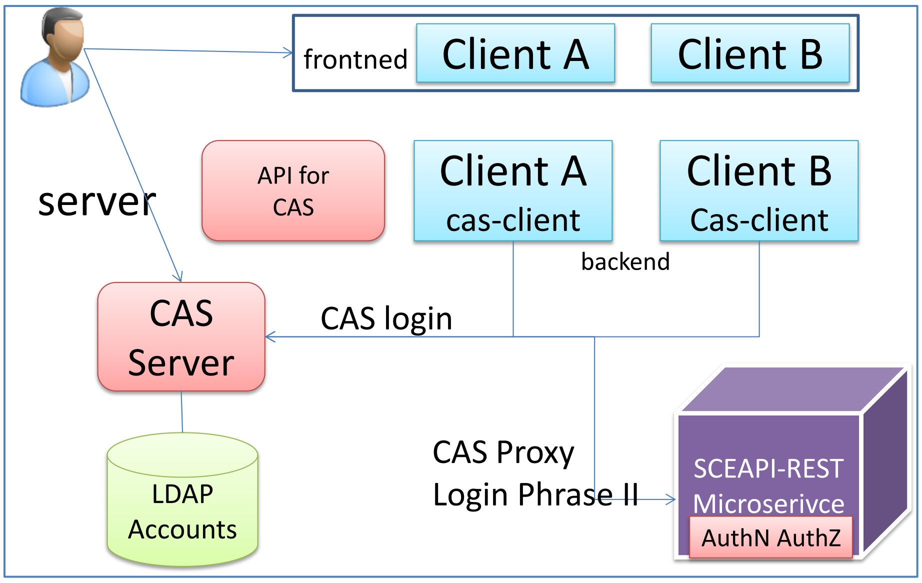

Figure 2 the context and modules of SSO Authentication microservice

This microservice is built on Apereo CAS, a famous and enterprise-class open source framework for SSO [5]. All accounts are managed and stored in a light weight LDAP database in SCE. It is easy to connect Apereo CAS to the LDAP database and add CAPTCHA to the Apereo CAS to avoid malicious attacks. Besides, attributes of user principle are modified to satisfy requirements of account mapping in SCE. At last, SSO service is provided for different clients developed by 3rd-party and ourselves developers.

Generally, a client is composed of a frontend displayed as certain web pages or a mobile application for any mobile device, and a backend that receives requests from the corresponding frontend and perform actions on behalf of a user by calling APIs provided SCEAPI-REST, discussed in section 3.2.3 in detail. In order to provide SSO, any backend of the clients needs to insert a client of Apereo CAS into its workflow and configure parameters to protect certain sensitive actions. If a developer can't find a client of Apereo CAS in specific programming language or platform, the developer could implement a customized client easily and quickly based on RESTful APIs provided in this paper and protocols [6] provided by Apereo CAS.

3.2.2 Authorization microservice

The authorization microservice is designed on principle of resources and roles. According RESTful rules, everything is abstracted as a resource and all actions could be mapped into several operations in HTTP protocol, such as GET, POST, DELETE, and so on. SCEAPI-REST provides many actions on different resources including but not limited to jobs, supercomputers, applications. Besides, there are 3 important roles in SCE. A user performs actions on certain clients in SCE. A developer designs and implements a client for the specific scenarios. An administrator determines what APIs could a client has permissions to access. From the above consideration, the authorization microservice provides resource and role based authorization service for computing users, clients and administrators, as show in figure 3.

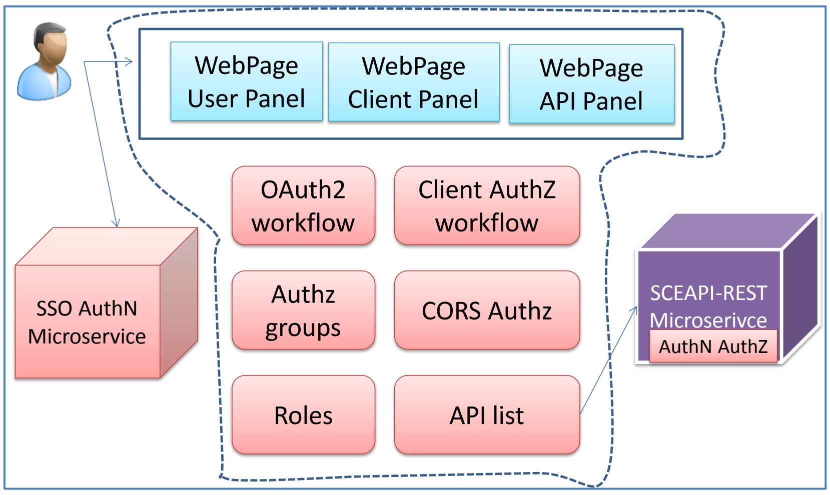

Figure 3 the context and modules of authorization microservice

According to features of resource and derived RESTful APIs, several super groups and common groups are defined for each API in SCEAPI-REST. Each API could belong to one or more authorization groups. An API that does not belong to any group cannot be accessed by a client. The granularity of authorization is a group that could be composed of empty, one or more APIs. By adding or removing APIs in the specific group, multi-level granularity of authorization could be implemented to meet the requirements of complex application scenarios. For super-privileged APIs, the action couldn't be performed unless the user and the client both have advanced permissions.

The Cross-origin Resources Sharing (CORS) module is used to determine whether a client can call SCEAPI-REST from the browser to better support single web page applications. When a request is received by SCEAPI-REST, the microservice gets CORS information, and then checks whether a request from different domains has permissions for performing certain actions. In addition, this microservice provides a standard OAuth2 workflow for users to authorize a client accessing computing resources and private data on behalf of him, and provides a simple procedure for developers to apply permissions which a client needs to implement workflows for complicated scenarios.

3.2.3 SCEAPI-REST microservice

SCEAPI-REST is a unified RESTful web API for HPC and enables access to massive HPC resources aggregated by SCE middleware in ScGrid and CNGrid. SCEAPI provides developers a set of easy-to-use programming capabilities for accessing HPC resources, such as jobs, data, supercomputers, and applications. Full discussion of SCEAPI-REST is out of the scope of this paper. Only authentication and authorization service is discussed in this section. In order to support different clients accessing HPC resources integrated into SCE, several authentication modes were added to SCEAPI-REST microservice, as shown in figure 4.

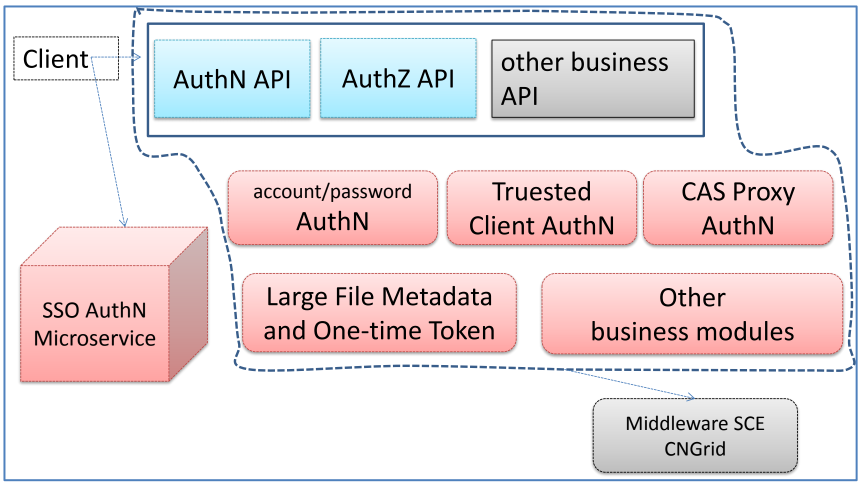

Figure 4 the context and modules of SCEAPI-REST microservice

For simplicity, it is common to the SCEAPI-REST microservice providing username and password authentication method for testing and scripts written by professional computing users. It is not recommended that developers adopt this simple authentication method in production environment for security reasons. Trusted client authentication belongs to super-privileged authentication method, which requires a client firstly obtains a certificate from SCEAPI-REST. A client proves its identity to the SCEAPI-REST microservice with RSA or DSA certificates when the backend of the client starts. In this method, SCEAPI-REST only verifies whether the username provided by a trusted client exists on the account database LDAP and is available. For security, only the clients implemented and maintained by our own team could be permitted to use this authentication method.

For clients implemented and maintained by 3rd-party, it is recommended that developer adopts CAS proxy method to access APIs provided by SCEAPI-REST microservice in their clients. It is well known that Apereo CAS provides a simple method to implement proxy authentication. A developer could configure his client easily to support proxy authentication according to guide documents provided by Apereo CAS. The SCEAPI-REST microservice will check proxy authentication information and determine whether permit or deny a request comes from a client. In addition, SCEAPI-REST microservice also generates metadata and one-time token of a large file for the large file transfer microservice discussed in section 3.2.4 in detail.

3.2.4 Large file transfer microservice

It is common to a job that has large input or output files which are hundreds of megabytes or even tens of gigabytes. It is not suitable for SCEAPI-REST to perform large files transfer service because the transfer service often spends lots of time and consumes much CPU capability so that the response time might be much longer for the transient actions that only need a little CPU capability. Therefore, a dedicated microservice for transferring big files is implemented to enhance data transfer capability and support high throughout jobs, as shown in Figure 5.

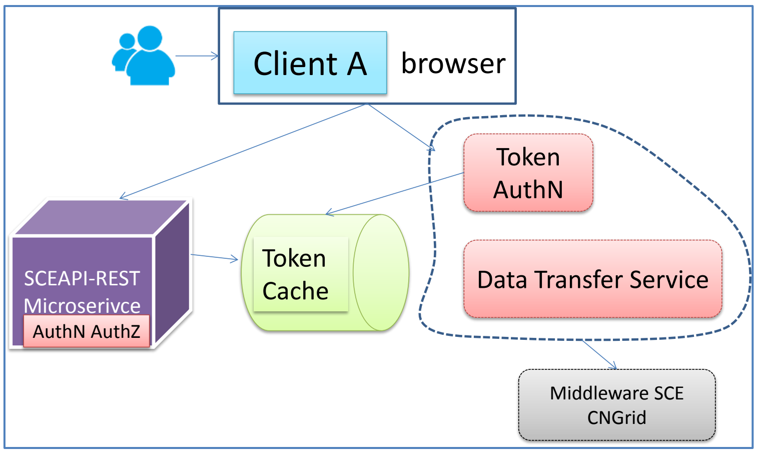

Figure 5 the context and modules of Big file transfer microservice

If a user wants to transfer a large file, he firstly accesses a client and declares that a large file needs to be transferred. After authentication and authorization check passed, SCEAPI-REST accepts a transfer request coming from the client, gets metadata of the large file, generates one-token, saves information into shared storage, then returns a redirected response to the client. The client accesses the big file transfer microservice via the URL that carries one-time token. Upon receiving a request, the token authentication module is responsible for check whether the one-time token is available and being used for the first time. Then the data transfer service begins to transfer the specific file as the URL requested. If there is something wrong caused by network or other reasons, the transfer could be restarted and continues from last position if the request from the same client that defines by User-Agent in browser, original IP and other information.

3.3 Base services

This layer is composed of storage and open source software. The MySQL database provides persistent storage of user information and authorization information. The Memcached, open source distributed caching system, provides caching storage capability storing meta-information of large files, temporary tokens and other environmental data for high-speed access. SCE middleware provides massive computing resources and a large number of application softwares, models and algorithms to complete user's computing tasks.

4. Simple authentication and authorization services

Based on the architecture and microservices proposed in this paper, authentication and authorization services are provided for solving issues among computing users, clients and

SCEAPI-REST. In this section, RESTful API, authentication methods and workflows, and authorization service are discussed as follws.

4.1 RESTful API for authentication and authorization

The authentication services provided in this paper are REST API and an SDK libraries. All RESTful API shares the same access entry point described as a second-level domain name. Each API is defined by relative path, HTTP methods and parameters, as shown in Table 1. The SSO service is built on Apereo CAS, which also provides different kinds of client SDK in popular programming languages. Compared with modifying and customizing authentication in a client, RESTful API provided in this paper provides a more flexible authentication method, which is convenient for developers to design and implement different authentication workflows that meet requirements of application scenarios.

Table1 RESTful open API for simplified authentication and authorization

	HTTP methods	API path	remark
Simple authentication	POST	/users/login	Username/password
	POST	/users/login/platform	Trusted client and username
	GET	/users/logout	logout
SSO authentication	POST	/casv4-service/v1/ticket	authenticate and return a TGT
	POST	/casv4-service/v1/ticket/{TGT}	A TGT changes to a ST
	GET	/{protected-client}/?ticket=ST-*	Access a protected client
	DELETE	/casv4-service/v1/ticket/{TGT}	Logout SSO

4.2 Multiple authentication methods and two-phrase authentication workflow

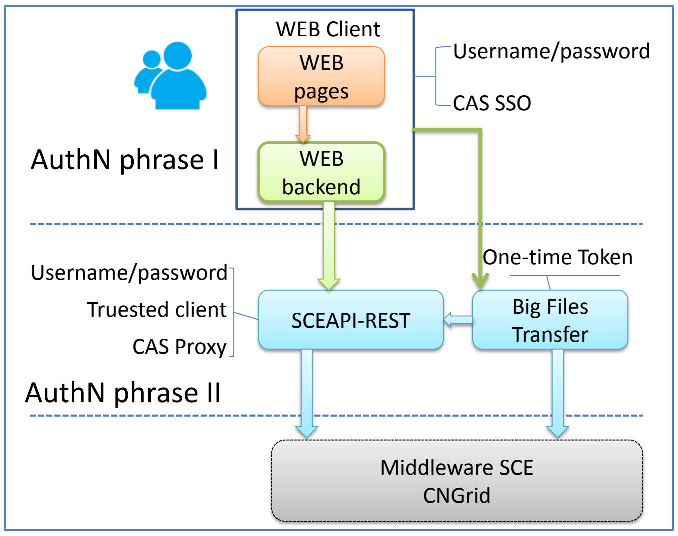

Figure 6 multiple authentication methods and tow-phrase authentication workflow

The simple authentication service proposed in this paper is divided into two phases, as shown in Figure 6. The first phrase is used to login into a client via an account of SCE. Clients such as WEB gateways and mobile applications supporting HTTP protocol can perform authentication workflow through SSO which provides good experience and avoids risk of clients disclosing account sensitive information. The username\password authentication method could be used only in developing or test environments.

The second phrase is used to login into SCEAPI-REST microservice for a client on behalf of a user in SCE. The username\password authentication method is the easiest way to authenticate. For security reason, it is generally used in the development and testing phase of clients and Linux script. The trusted client authentication method requires check the identity of a client by means of RSA or DSA certificates. The SSO authentication method is more secure and reliable. However, this method requires the WEB gateways to develop an additional proxy authentication workflow. Then the SCEAPI-REST microservice will check proxy authentication information and determine whether permit or deny a request comes from a client.

4.3 Multi-granularity authorization service

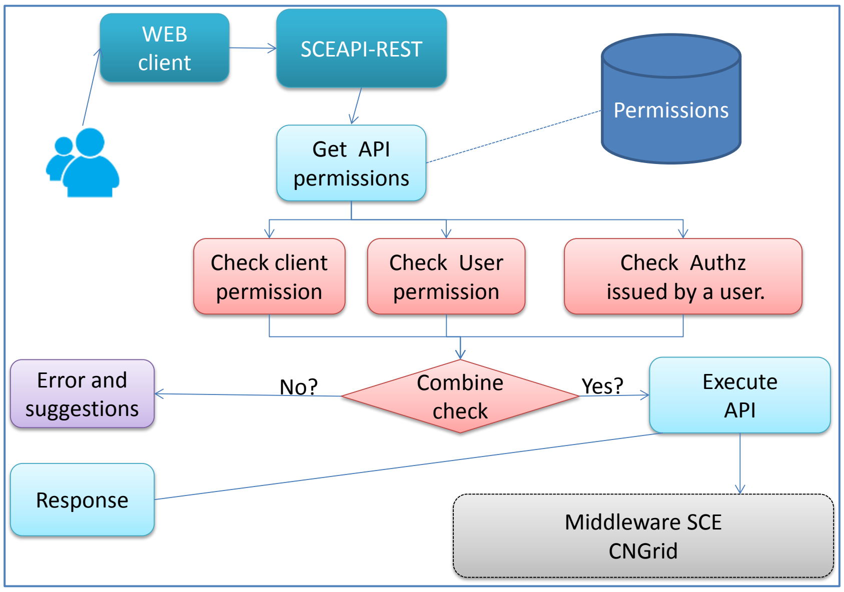

Figure 7 authorization workflow of processing an request from a client

The authorization service proposed in this paper enables multi-granularity authorization service and simple audit service in SCE. Each requests for an API goes through the entire authorization check and logging workflow, as shown in figure 7. After login to a WEB client, a user can initiate a request, such as submitting job metadata to to create a computing job. First, if the request is sent from the browser directly, the client must pass the CORS authorization check. Otherwise the call is aborted and an error message is returned. Secondly, SCEAPI-REST microservice will check whether the request has permission for calling the API. The result depends on the intersection of client permission, user's permission and authorization granted by the user to the current client. If the intersection is not empty, the check is passed. At last, the SCEAPI-REST microservice checks the signature information about the API calculated based on an HTTP method, a request URL, a timestamp, and a secret key returned when login successfully. If not, the request is aborted and an error message is returned. If passed, the request is processed and a response is returned.

Each microservice, which takes part in dealing with a request, always records metadata and status information about the request identified by a global unique and random ID. Then the log written by different microservices is collected by a log server. With help of an offline log analyzation tool, an administrator could audit entire workflow for a request marked by a global ID by analysing logs separated from different microservices.

5. Typical use cases

The simple authentication and authorization services proposed in this paper has been applied to the WEB clients such as general computing Portal2.0[7], SCE status display gateway[8] and other gateways or web communities in different disciplines and areas, such meteorology, chemistry, biology, genetics, material and so on. The typical use cases show that the simple authentication and authorization services proposed in this paper has good performance and user experience.

Portal2.0 provides users with an one-stop console for performing computing tasks, which supports easy-to-use graphical interface and supports full lifecycle management of jobs [7]. The first phrase authentication used in Portal2.0 is the SSO authentication method so that all users could access this gateway via global accounts provided by SCE. Because Portal 2.0 is developed and maintained by our team, the trusted client authentication method is used in second phrase authentication. From the view of a user, what all he needs to do is to login into portal2.0 with an account provided by SCE, and then he can perform actions against jobs, data, supercomputers, and applications in SCE.

The SCE status display gateway gathers status information about different resources, and displays status information in multiple dimensions, such as queues, nodes, jobs, accounts, utilization of CPU, memory, storage, and so on [8]. The gateway is authenticated in first phrase by SSO authentication method provided by China Science and Technology Network. For security, only users in the white list can access this gateway. The trusted client authentication method is used in the second phrase authentication because the platform is developed and maintained by ourselves. What’s more, the gateway accesses SCEAPI-REST from the front-end browser in a read-only manner so that each request is checked by the CORS module. If passed, the SCEAPI-REST queries information on backend databases and returns a response that includes status data ordered by timestamp.

6. Related works

Based on grid computing and cloud computing technologies, sscientific computing environment aggregates massive computing resources to provide users with high-performance, high-throughput, and artificial intelligence computing services. XSEDE (Extreme Science and Engineering Discovery Environment) provides resources and services in the form of a single virtual supercomputer [9].WLCG (World Wide LHC Computing Grid) is the world's largest high-energy physics computing and storage facility [10], which provides computing and data services for high energy physics. The national high-performance computing environment (CNGrid) aggregates massive computing resources including five national supercomputing canters in Tianjin, Jinan, Shenzhen, Changsha and Guangzhou in China [11]. Globus Online [12] and Globus Platform-as-a-Service [13] can hosted service that lets you use powerful grid and cloud computing capabilities without installing softwares. Globus also provides foundational identity and access management platform services designed toaddress unique needs of the science and engineering community [14].

In authentication, a pair of username and password is the most widely used authentication method, despite the high security risks. MyProxy is an authentication method based on x.509

certificates [15], which is widely used in grid computing and is also the main authentication method of XSEDE [16]. Compared with the one-time authentication method, the multi-phrase authentication method is more complicated and strict, which can effectively prevent attacks of hijacking accounts [17]. In this paper, multiply authentication methods and two phrases authentication are provided to meet different and complicated requirements.

In authorization, RBAC is based on roles and forms a many-to-many mapping relationship between users and permissions [18] Attribute-Based Access Control (ABAC) can deal with complex context information, support massive resource entities and users, and perform rapid authorization verification [19], but authorization strategies are often too complex to evaluate effects caused by even a little change in policies. To deal with complex and dynamic features required in SCE, OAuth2.0 and resource-role authorization is used to enhance the RBAC capability and provide multi-granaulity authorization services.

7. Summary

All resources in SCE are packaged as easy-to-use open APIs in RESTful web services. Around authentication and authorization issues among users, open APIs and clients, simplified authentication and authorization services were proposed and implemented in this paper. Several micro-services were implemented or deployed to provide easy-to-use authentication and authorization for RESTful WEB API in SCE. Atop the proposed services, all related people, consisting of users, developers and administrators, they no longer need to worry about and solve complex problems with authentication and authorization. What they need to pay much attention on are specific business logics and application scenarios for their interested areas. In future, much more account sources will be added to the authentication microservice proposed in this paper.

ACKNOWLEDGMENTS

This work was partially supported by the National Key Research and Development Program in the Ministry of Science and Technology (MST) under the contact number 2018YFB0204001, by the Youth Fund Project that belongs to National Natural Science Foundation of China under the contact number 61702476, and by Beijing Natural Science Foundation combined with Haidian Original Innovation Foundation under the contact number L182053.

References

[1] Depei, Qian. "CNGrid: A test-bed for grid technologies in China." Distributed Computing Systems, 2004. FTDCS 2004. Proceedings. 10th IEEE International Workshop on Future Trends of. IEEE, 2004.

[2] Xiao H, Wu H, Chi X. SCE: grid environment for scientific computing[C]//International Conference on Networks for Grid Applications. Springer, Berlin, Heidelberg, 2008: 35-42.

[3] Computer Network Information Center, Chinese Academy of Sciences. Super Computing Environment in Chinese Academy of Science. http://cscgrid.cas.cn, 2018-7-29.

[4] Rongqiang, Cao, et al. "SCEAPI: A unified Restful Web API for High-Performance Computing." Journal of Physics: Conference Series. Vol. 898. No. 9. IOP Publishing, 2017.

[5] Apereo Foundation. Enterprise Single Sign-On – CAS. https://www.apereo.org/projects/cas, 2018-7-29.

[6] Apereo Foundation. CAS protocol. https://apereo.github.io/cas/4.2.x/protocol/CAS-Protocol.html, 2018-7-29.

[7] Computer Network Information Center, Chinese Academy of Sciences. Portal2.0 Computing Service. http://test.cngrid.org, 2018-7-29.

[8] Computer Network Information Center, Chinese Academy of Sciences. Monitor and Status Display for Scientific Computing Environment. http://show.cngrid.org/show/cngrid, 2018-7-29.

[9] Towns J, Cockerill T, Dahan M, et al. XSEDE: accelerating scientific discovery[J]. Computing in Science & Engineering, 2014, 16(5): 62-74.

[10] Bird, Ian, et al. Update of the Computing Models of the WLCG and the LHC Experiments. No. CERN-LHCC-2014-014. 2014.

[11] National Computing Environment. Operation and Maintain Center of CNGrid. http://www.cngrid.org, 2018-7-29.

[12] Foster I. Globus Online: Accelerating and democratizing science through cloud-based services[J]. IEEE Internet Computing, 2011, 15(3): 70-73.

[13] Ananthakrishnan, Rachana, et al. "Globus platform ‐ as ‐ a ‐ service for collaborative science applications." *Concurrency and Computation: Practice and Experience* 27.2 (2015): 290-305.

[14] Tuecke, Steven, et al. "Globus Auth: A research identity and access management platform." *2016 IEEE 12th International Conference on e-Science (e-Science)*. IEEE, 2016.

[15] Dooley, Rion, Joe Stubbs, and Jim Basney. "The MyProxy Gateway."Science Gateways (IWSG), 2014 6th International Workshop on. IEEE, 2014.

[16] Basney, Jim, et al. "Integrating science gateways with xsede security: A survey of credential management approaches." Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment. ACM, 2014.

[17] Anantula, Pranayanath Reddy, and G. Manoj Someswar. "Authenticating users with multiple levels of validations in a secure Cloud Computing Environment." i-manager's Journal on Cloud Computing 3.3 (2016): 9.

[18] Ferraiolo, David F., et al. "Proposed NIST standard for role-based access control." ACM Transactions on Information and System Security (TISSEC) 4.3 (2001): 224-274.

[19] Jin, Xin, Ram Krishnan, and Ravi Sandhu. "A unified attribute-based access control model covering DAC, MAC and RBAC." IFIP Annual Conference on Data and Applications Security and Privacy. Springer, Berlin, Heidelberg, 2012:41-55