Security analysis of OpenStack keystone

Baojiang cui Tao xi
Beijing University of Posts and Telecommunications, National Engineering Laboratory for Mobile Network Security
Beijing, china
cuibj@bupt.edu.cn xitao_kobe@163.com

***Abstract*—As a base platform of cloud computing, OpenStack's has getting more and more attention. Keystone is the key component of OpenStack, we analyze the security issues of Keystone and find some vulnerabilities of it and then, we propose a new authentication model using both symmetric encryption and asymmetric encryption. Through the security test of new model, it is proved that the new model is much safer than the original one.**

Keywords—cloudcomputing; OpenStack; security; keystone;authentication model

I. FORWORD

OpenStack is a free and open-source cloud computing software platform [1]. User primarily deploy it as an infrastructure as a service solution [2]. Until now, OpenStack has released 10 versions from A to J, functions of the system has been continuously improve during the evolution of the version [3]. As an open-source cloud platform, its safety is relatively high, but there are still bugs and vulnerabilities which have become one of the most important points need to be considered in the evolution of its development. Keystone is an important component of OpenStack responsible for authentication [4]. This paper mainly studies the security issues of OpenStack authentication protocols, and implements a series of in-depth analysis of system architecture in order to understand the working principle of Keystone [5]. From the analysis of Keystone we find some defects and problems, and for those defects and problems we proposed appropriate measures and a new authentication model.

II. RELATED WORK

OpenStack security researchers have actively studied the safety aspect of the platform and found a variety of security issues [6], but did not directly solve the vulnerabilities of OpenStack. Sasko Ristov [7] tested four different platforms and analyzed the security assessment of OpenStack and found some vulnerabilities. They discussed details about these vulnerabilities and showed how they can be solved by appropriate patches and other solutions. Rostyslav Slipetskyy [8] implements a study of security issues pertinent to cloud computing by examining evolving standards and formed a basis for analyzing how identified security issues are mitigated in OpenStack. OpenStack's safety-related work mainly focuses on the perspective of the overall security of the cloud platform, but did not implement in-depth research on a special functional component. This paper will implement in-depth research on Keystone components and proposed some solutions of its problems.

III. INTRODUCIONT TO KEYSTONE

Keystone is an OpenStack identity service that manages user databases and OpenStack service catalogs and their API endpoints. It integrates with existing backend directory services like

978-1-4799-8873-0/11 $31.00 © 20115 IEEE
DOI 10.1109/IMIS.2015.44

LDAP [9] and supports multiple authentication mechanisms, such as username-and-password and token-based systems. OpenStack identity management is based on standard PKI token, allows the client to get offline token authentication without identity services.

There are two authentication mechanisms in Keystone, UUID token used in versions before G, and PKI [10] used after G which may improve the security of Keystone.

UUDI Token

The Fig.2 below shows how tokens were originally generated by Keystone and then used by the client to "sign" every subsequent API request.

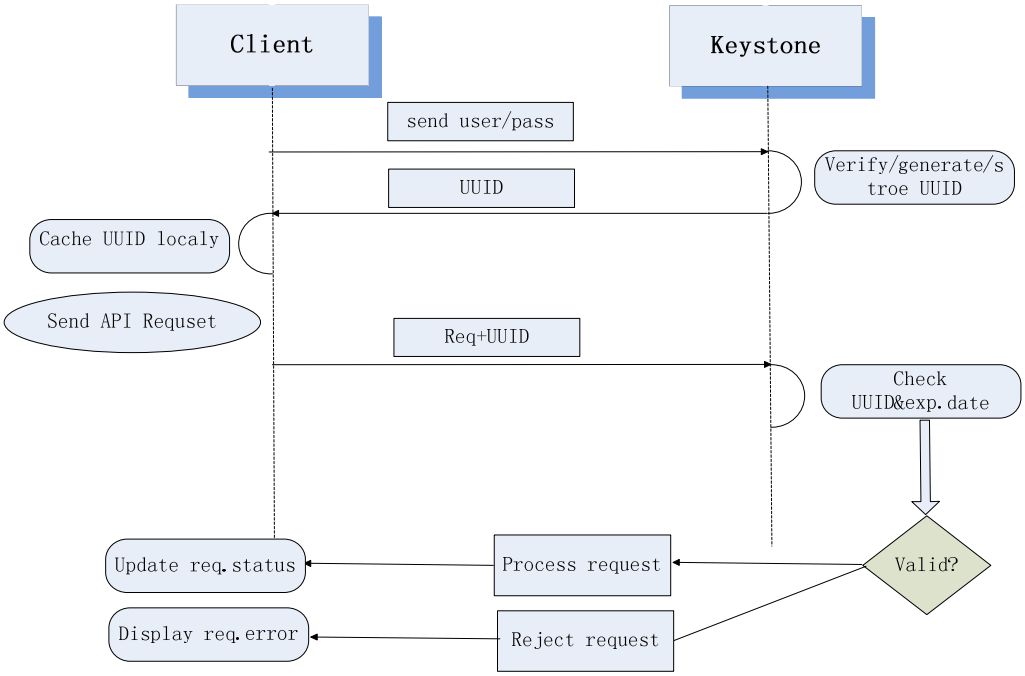

Fig. 1. UUID token process

Based on supplied username/password pair which we assume to be correct, the Keystone server would generate an UUID token and send it to client. This token will be able to prove the identity of the client .While the client needs to send a request to the Keystone, it will add the UUID in the request to prove that the request is sent by certified client.

PKI

The Fig.3 below shows how token validation is performed with the new method introduced in OpenStack's Grizzly release.

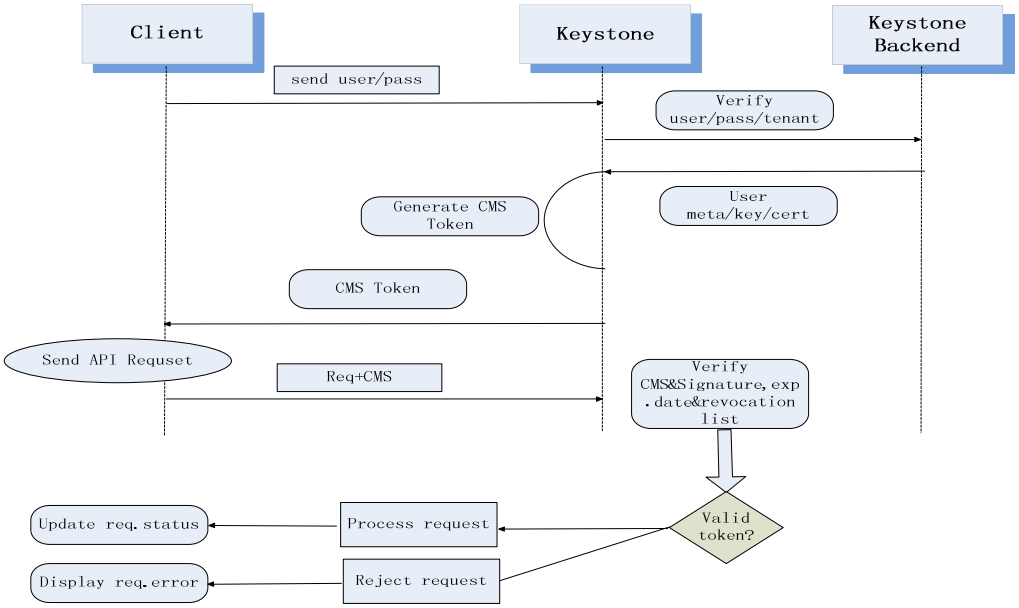

Fig. 2. PKI token process

From Fig.3 we can see that with PKI tokens, Keystone is becoming a Certificate Authority [11]. Different to previous version, Keystone uses its signing key and certificate to sign but not encrypt the user token, and because of that, each API endpoint needs to hold a copy of Keystone's: Signing certificate, Revocation list and CA certificate which will be used to validate the user request. What is verified now is the signature Keystone puts on the user token instead of direct request to Keystone.

IV. SECURITY ANALYSE OF KEYSTONE AUTHENTICATION MECHANISM

Based on analysis of Keystone's authentication mechanism, we found that there are still vulnerable points in current protocol. These vulnerabilities could cause a great impact on the system, such as denial of service, information disclosure, reply attack and so on. The main reason for this is that security comes at a cost, and the designers did not consider the security issues of the system carefully, or maybe the designers consider such security issues can be ignored. Below are some vulnerabilities of the Keystone authentication mechanism:

- As OpenStack's certification center, Keystone use username/password to authenticate client. But during authentication, username and password are sent in clear text. Keystone use the http protocol for authentication which would cause information disclosure, an attacker can get the username/password during the authentication.
- After successful authentication, the Keystone backend would generate a token and send it to client. The token is like a key which will be used when the client need to use other services. This design is not appropriate in security perspective. Once an attacker gets the token, he will get the client's whole privileges which would cause unimaginable disaster to the system.
- The API request send by client is composed by client token and the request message. An attacker can easily implement a reply attack and get the information he want.

V. AN ENHANCED SECURE MECHANISM FOR KEYSTONE

From the analysis of the Keystone authentication mechanism, in order to improve the security of Keystone, we propose a new authentication model based on the original mechanism. The new model combines both symmetric encryption and asymmetric

encryption [12]. Before we introduce the new model, we need to know some notations that will be used. This paper uses the list notations:

TABLE I. NOTATIONS

Notation	Explanation
C	Client
K	Keystone
KU	Public key possessed by client and Keystone. KU_C means KU possessed by client and KU_S means KU possessed by Keystone.
KR	Private key possessed by client and Keystone. KR_C means KR possessed by client and KR_S means KR possessed by Keystone.
$E_K(M)$	Encrypt message M with a key K
$M_A \mid M_B$	Concatenation of the message M_A and M_B
A → B: M	A send a message M to B

Fig.4 shows the secured Keystone authentication model:

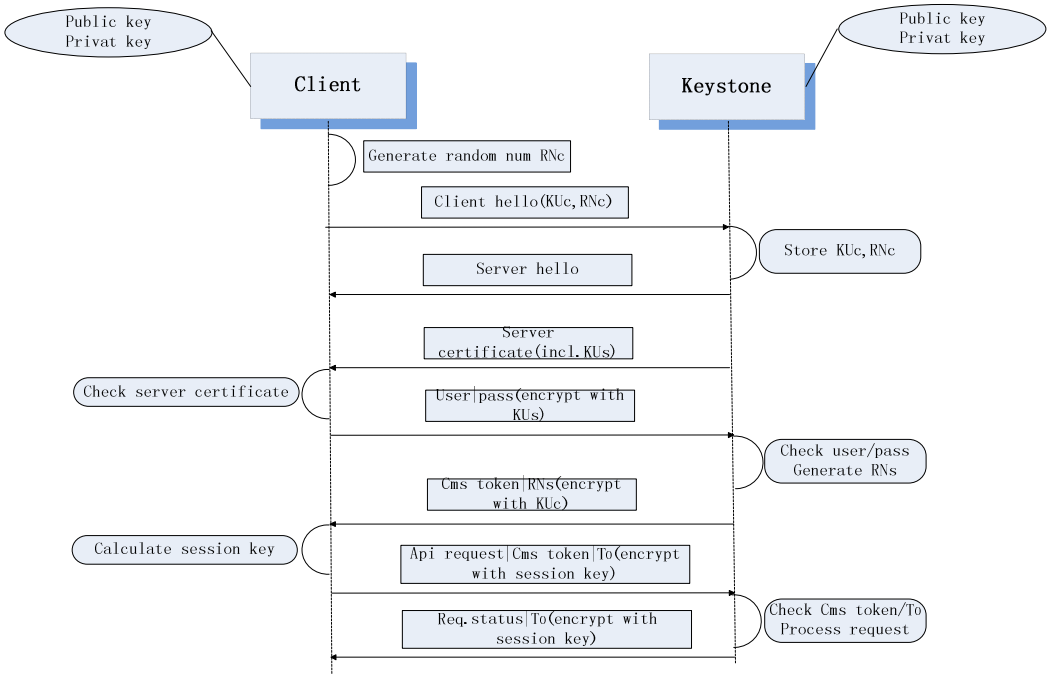

Fig. 3. Secured authentication model

A. Client hello message

Client generates random num RNC and sending the client hello message to Keystone server including client public key and the random num RNC. The random number RNC is used to generate session key which will be used in API request in order to guarantee the security

of the request.

$$C \rightarrow K: client\ hello|KU_C|RN_C$$

B. Server hello message

Keystone server receives the server hello message and store the client public key and client random num RNC locally. Then the Keystone server will send client hello message as a reply.

$$K \rightarrow C: server\ hello$$

C. Server certificate

The Keystone sends its certificate and public key to the client.

$$K \rightarrow C: server\ certificate\ |KU_S$$

D. Client user/pass

The client will verify the validity of the certificate. If the Keystone server's identity has been verified, the client will send its user/pass (encrypted by KU_S) to the keystone.

$$C \rightarrow K: E_{KU_S}(\text{user}|\text{pass})$$

E. Token transmission

The Keystone server decrypts he message using its private key and check the username and password. If the client's identity has been verified, it will send the CMS token to client.

$$K \rightarrow C: E_{KU_C}(\text{CMS token}|\text{RNs})$$

CMS token is generated by Keystone backend, RNs is a random number used to calculate session key.

F. API request with token

The client use DH key agreement algorithms to generate session key and then send an API request message to the Keystone server.

$$C \rightarrow K: E_{SESSION\_KEY}(\text{API Request}|\text{CMS token}|T_0)$$

T_0 is a time stamp used to avoid replay attacks.

G. API request reply

The server will check the CMS token and T_0 to decide whether to process request. If the CMS token and T_0 are valid, the server will reply the following message to the client:

$$K \rightarrow C: E_{SESSION\_KEY}(\text{Req. status}|T_0)$$

A complete authentication and request process is finished while the API request gets a reply.

VI. VI.SECURITY ANALYSIS

By comparing to the original authentication mechanism in security and availability, we can see that the new model is relatively safe and reliable. Firstly, we'll analyze the ability of the enhanced Keystone authentication model of defending against several major security threats.

A. Defending against eavesdropping attacks

The enhanced Keystone authentication model use public key and private key to prevent eavesdropping. From Fig.4 we can clearly see that only processes A, B, C is not encrypted. From processes A, B, C a attacker can get KU_C , KU_S , RN_C and server certificate, but even with this information, the attacker can not cause any harm to the system.

B. Defending against replay attacks

Once the communication channel is monitored, attackers can easily get connecting message and then launch a replay attack. In this paper, time stamp T_0 which is recorded when client send API request to the server is used to avoid such attacks. If a message is sent twice form client to the server, the server will check out the exception by comparing the time stamp T_0 with current time.

C. Defending against man-in-the-middle attacks

In this secure mechanism, attacker cannot launch man-in-the-middle attacks, which is because even the attacker gets part of the secure information it cannot reply a reasonable response message. For example, if a fake client send a client hello message

KU_C, RN_C when it receive the reply message server hello and server certificate including server public key, it cannot reply anymore since it does not have the user|pass.

Secondly, we'll analyze the authentication mechanism of the new model.

D. *The Bidirectional entity authentication*

The client uses the certificate to verify the server, and the server uses user|pass to verify the client. Because identity information contained in digital certificate has been signed by an authoritative third party, the client will first confirm the identity of the service and then sends private message. Neither fake client nor fake server can get any useful information of the system.

VII. CONCLUSION

Considering the shortcomings of OpenStack Keystone authentication mechanism, we proposed an enhanced secure mechanism of Keystone authentication system for OpenStack. This paper first makes an introduction of Keystone and the latest related research, then discusses its security vulnerabilities and proposes suggested security services based on the security features of Keystone. This paper describe the details of the design and implementation of the new authentication model, and simply analyzes the security feature of the model with the testing of its ability of defending against several major security threats.

According to the actuality of security-related research, this work is an initial but important considering the widespread use of the OpenStack platform. The future work would include further research of secure mechanism to defend against more potential attacks and research of lightweight key exchange algorithm to ensure the security of information exchange. Besides, the security analysis of OpenStack and its other component would also be our top priority.

ACKNOWLEDGEMENT

This work was supported by National Natural Science Foundation of China (No. 61170268, 61100047, and 61272493).

REFERENCE

[1] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing[J]. Communications of the ACM, 2010, 53(4): 50-58.

[2] Mell P, Grance T. The NIST definition of cloud computing[J]. National Institute of Standards and Technology, 2009, 53(6): 50.

[3] Sefraoui O, Aissaoui M, Eleuldj M. OpenStack: toward an open-source solution for cloud computing[J]. International Journal of Computer Applications, 2012, 55(3): 38-42.

[4] Jackson K. OpenStack Cloud Computing Cookbook[M]. Packt Publishing Ltd, 2012.

[5] Khan R H, Ylitalo J, Ahmed A S. OpenID authentication as a service in OpenStack[C]//Information Assurance and Security (IAS), 2011 7th International Conference on. IEEE, 2011: 372-377.

[6] Ristov S, Gusev M, Kostoska M. Security assessment of OpenStack open source cloud solution[C]//Proceedings of the 7th South East European Doctoral Student Conference (DSC2012). 2012: 577-587.

[7] Ristov S, Gusev M, Donevski A. OpenStack cloud security vulnerabilities from inside and outside[C]//CLOUD COMPUTING 2013, The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization. 2013: 101-107.

[8] Slipetskyy R. Security issues in OpenStack[J]. Master's thesis, Norwegian University of Science and Technology, 2011.

[9] Howes T A, Smith M C, Good G S. Understanding and deploying LDAP directory services[M]. Addison-Wesley Longman Publishing Co., Inc., 2003.

[10] Nash A, Duane W, Joseph C. PKI: Implementing and Managing E-security[M]. McGraw-Hill, Inc., 2001.

[11] Vaeth J S, Walton C S. Virtual certificate authority: U.S. Patent 6,035,402[P]. 2000-3-7.

[12]Fujisaki E,Okamoto T. Secure integration of asymmetric and symmetric encryption schemes[C]