Journal of Engineering and Applied Sciences 14 (Special Issue 7): 10186-10194, 2019
ISSN: 1816-949X

High-Assurance Security in Swift Component

[1]Hala A. Albaroodi, [2]Aseel Ismael and [3]Mohammed Abomaali
[1, 2]Ministry of Education, Baghdad, Iraq
[3]Deptartment of Computer Techniques Engineering, Alsafwa University College, Karbala, Iraq

Abstract: Cloud platform includes several storage services that can be used to manage the needs of users in both private and public environments. Swift, an object storage component of the OpenStack cloud computing platform is used to store, manage and retrieve data. The CIA Model is key considerations for cloud computing services such as the Swift storage. Due to the simplistic nature of Swift storage, its trust model has a negative implication and this amplifies the potential impairment from compromised node, leaving the cloud platform vulnerable to various attacks. The existing model requires fundamental modification with improvement in security and trust level. The authors discuss in details the steps of the proposed model and how improving the security of cloud computing can be achieved. The results of this research indicate that the proposed model, i.e., CloudSecure have significantly improved and effective the security. Conclude, the cognitive effectiveness of the existing OpenStack Swift has been improved.

Key words: Security, authentication, authorization, access control, analysis, MD5 hash function, RSA, man-in-the-middle attack

INTRODUCTION

As data needs to be immediately available and stored indefinitely on a variety of devices, the demand on storage is rapidly changing (Pearson and Benameur, 2010). This demand requires the construction of storage silos that utilize standard protocols which are restricted to specific applications. Online video, social media and gaming are applications that demand the driving of this change.

Public cloud storage services have strived to satisfy these new storage needs, however, most organisations are not capable of building their own scalable storage solution. Thus, using cloud storage is inevitable (Ren *et al*., 2012). However, to provide these changing needs, a storage model must be able to handle web-scale workloads with many simultaneous readers and writers to a data store.

The possibility of utilising Cloud Computing (CC) based on the Open Source Software (OSS) technology such as OpenStack is promising due to their cost-effectiveness and its fast development cycle (Albaroodi *et al*., 2013). Swift as one of the basic components of the OpenStack project is employed to satisfy and on demands. Swift's usage includes small deployments for storing VM images, mission-critical storage clusters for high-volume websites, custom file-sharing applications, mobile application development and data analytics.

However, security is one of the hindrances in proliferation of CC (Kaufman, 2010; Slipetskyy, 2011). Intention is targeted in a way that Swift needs to be secure whereas other components like nova, cinder, dashboard and others are remained untouched. For the sake of criticizing it, OpenStack does not support password complexity requirements and passwords are stored in plaintext format. Thus, the access to the sensitive data files is not secured and can be hacked. Furthermore, information transformed within the OpenStack is not protected using any encryption and decryption techniques (Albaroodi *et al*., 2013). Therefore, OpenStack has not delivered a promising security level similar to its level of performance as a CC platform.

OpenStack Swift: OpenStack Swift version 1.10.0 is an IaaS delivery model that can be distributed in any deployment model. When OpenStack Swift is deployed in a cloud delivery model, a cloud vendor maintains security control. All security services are available through web services at the application levels. Figure 1 depicted the existing OpenStack Swift.

This study was conducted to identify the problem in the current OpenStack Swift which uses keystone as a principal component for secure user authentication (Khan *et al*., 2011). The authentication process is based on usernames and passwords which are transmitted as cleartext. A successful authentication on Swift must invoke the Keystone server which generates a valid token for using by the end-user. The use of HTTP protocol in all communications is considered to be less secure than the use of HTTPS (Ryan *et al*., 2015) as HTTP increases the

Corresponding Author: Hala A. Albaroodi, Ministry of Education, Baghdad, Iraq

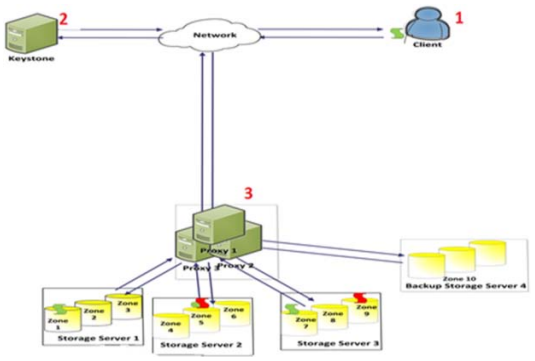

Fig. 1: Gaps in the existing OpenStack Swift (Carrez, 2013)

potential for particular security problems. Existing OpenStack Swift model uses a centralized entity known as proxy server that act as a middleware between the keystone server and the database repository. One of the most critical problems to note here is the centralized entity itself (keystone) that can affect the keystone by the network congestion or fault tolerance. The network congestion takes place when many users are accessing the proxy at the same time. On the other side, the fault tolerance happens, if the proxy goes down then the whole model goes down as well. Moreover, in the data repository the data transfer between the client and storage server take place in the insecure communication; these insecure communication makes it vulnerable for possible attacks (Ristov *et al*., 2014). Security of the OpenStack Swift finds in: identity, access control, password storage and strength password, data protection and lastly authentication and authorisation.

MATERIALS AND METHODS

Proposed model; CloudSecure: In order to address the shortcomings of the existing security model of OpenStack Swift, we proposed CloudSecure model. The important components of the CloudSecure model are client-side, server-side and database repository centre. The client-side component is cross platform and design to run on laptop, mobile smart phone and tablets. Figure 2 depicted the CloudSecure.

It provides a window-based Graphical User Interface (GUI) by which a new user can create a cloud service account or facilitate existing user to use CloudSecure services. For existing users, the username, password and CAPTCHA information will be displayed. Upon the opening of the application, the user credential module will connect with online cloud server to get the new CAPTCHA text and will be displayed on the user screen. The motivation to use

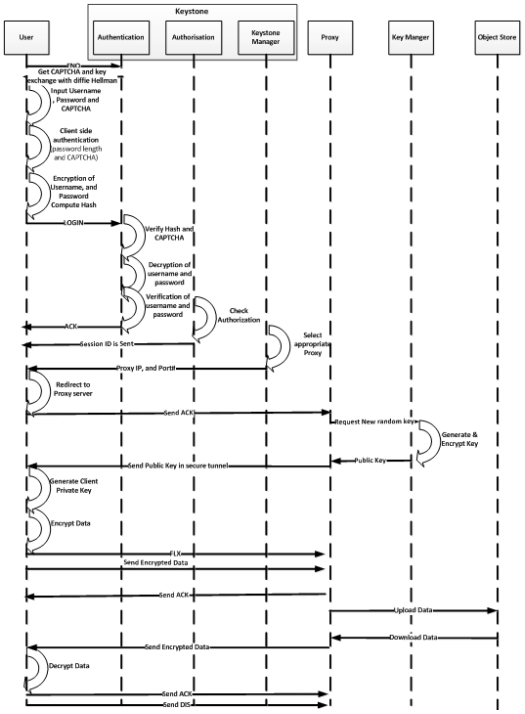

Fig. 2: Flow diagram of the CloudSecure model

CAPTCHA is to differentiate between a normal user and the robot to avoid keystone server exhaustion (Saxena *et al*., 2012).

By the users storing their data in the CC platform, users lose physical control of their data which requires each user to encrypt her/his data prior to uploading data to the cloud servers as mentioned earlier (Ristov *et al*., 2014). The use of cryptographic and hash function algorithms for data security will enable secure and efficient access control to user data. The CloudSecure model incorporates cryptographic algorithms at the different level such as (authentication/authorisation) and MD5 hash function along with CAPTCHA is used to introduce a more secure access method for user credential module. The motivation to use MD5 hash function is mainly to know whether the user credential information has been tampered by intruder or not (Grobauer *et al*., 2011). Moreover, users must be identified by keystone before they are allowed to use any of the OpenStack Swift services; this step guarantees a unique point of entry. Keystone server verifies the hash code along with CAPTCHA and later decrypts usernames and passwords and provides each user with a unique user credential that enables access to the services for which they are authorised.

The challenge in the Blowfish symmetric cryptography algorithm is that they key used to encrypt and decrypt should be same on client and server side, a

study conducted by Mohamed *et al*. (2012) shows that the time consumption metrics of the Blowfish symmetric algorithm needs less time to perform which could indicates its preference in comparison with other encryption algorithms. Practically, the generation is based on big random number and will be taken place on the keystone server side. One of the most problematic and crucial process in the Blowfish symmetric cryptography is the key exchange mechanism; key exchange using Diffie-Hellman key exchange algorithm (Diffie and Hellman, 1976). The motivation to use Diffie-Hellman as the key exchange algorithm is due to its vast applicability in many security protocols such as SSL, SSH, IPSec etc. (Carts, 2001). The key exchange will take place in the insecure channel between client and keystone server. The method of key exchange has been presented as below.

The user credential module (username, password and the CAPTCHA) has two authentication processes on client side and server side. The client-side authentication will check the length of the password text and validation of CAPTCHA text locally, the validation of the CAPTCHA will be done after the server side (keystone) sends the generated CAPTCHA to the client side. Once the client-side authentication process has been completed then the username along with password will be encrypted using Blowfish symmetric cryptographic algorithm. After the username and password has been encrypted using Blowfish symmetric cryptography algorithm, MD5 hash function value will compute the encrypted data (username and password) and the data will be sent to the Keystone server in the cloud.

The security mechanism in CloudSecure model in the user credential module in the server side involves authentication as well as authorization of the user. In the server-side authentication process, the hash code of the MD5 hash function received from client side will be matched with the hash code on the server side. If the hash code of the MD5 hash function is different, then the user authentication data (username, password, along with CAPTCHA) will be discarded (Saxena *et al*., 2012). The authentication process includes the decryption of username and password then matching with the entry from user authentication database. If the user credential module (usernames, password along with CAPTCHA) does not match then the server side will send NACK (Not Acknowledgement) to the client side and added to the blacklist; if the user failed three times consecutively, the user will also be added to the blacklist database. This specific user will be in the black list database for certain time period (1 month) that is customizable (Mitchell and Jones, 2005).

The authorisation process is executed once the authentication process is successful. This process is used to determine the type of user either a normal user or an administrator user (admin). The difference between both types of users is the number of features in terms of functionality as well as security.

One of the most critical processes in the keystone module is the keystone manager process which deals with the selection of appropriate proxy server based on the available hardware resources such as the availability of CPU and memory resource and the number of active users in its database. The current state of proxy server information will be updated in the database periodically to obtain the best available proxy server credential information for the new user. The Keystone manager will select the least busy proxy server and send its credential information to the client to ensure that it will be redirected to the specified proxy server for communication.

In order to ensure security when transferring data (uploading and downloading) and to reduce the possibility of the attack between the client and the data repository, the user should use the asymmetric cryptography algorithm. With the implementation of a RSA based PKI mechanism, CloudSecure model gives secure data transmission with strong authentication key appliance which supports the services of PKI and provides asymmetric key management libraries (Kalpana and Singaraju, 2012).

The key change is one of the operations of asymmetric encryption. The key change can be done by the help of random asymmetric key generation and later encrypting this with receiver of the public key. The RSA relies on the integer factorization difficulty with RSA encryption is done by the public key is substantially needs less computation cycle whereas RSA decryption is done by the private key. The RSA public key is known to everyone and decrypted with the private key.

However, in the OpenStack Swift Model, the security library does not feature secure key management at the cloud platform. In addition, a separate server for key storage is required to keep the key in a secure location that is only be accessed by proxy server. These keys are randomly generated and will be discarded after specific time interval. After the time interval passed, the new keys will be generated. Previously in the authentication process between client and keystone server, Blowfish symmetric cryptography along with Diffie-Hellman key exchange mechanism were used. After the authentication and authorisation the client will be redirected to proxy server and secure communication between the proxy and client. In this research, RSA asymmetric cryptography is used for secure data transfer between client and proxy server. Before the secure data transfer been initiated, the public key will be transferred from proxy server to the client. Once the public key is transferred to client through secure channel, then the private key will be computed using the receiver public key. Data will be retrieved from the

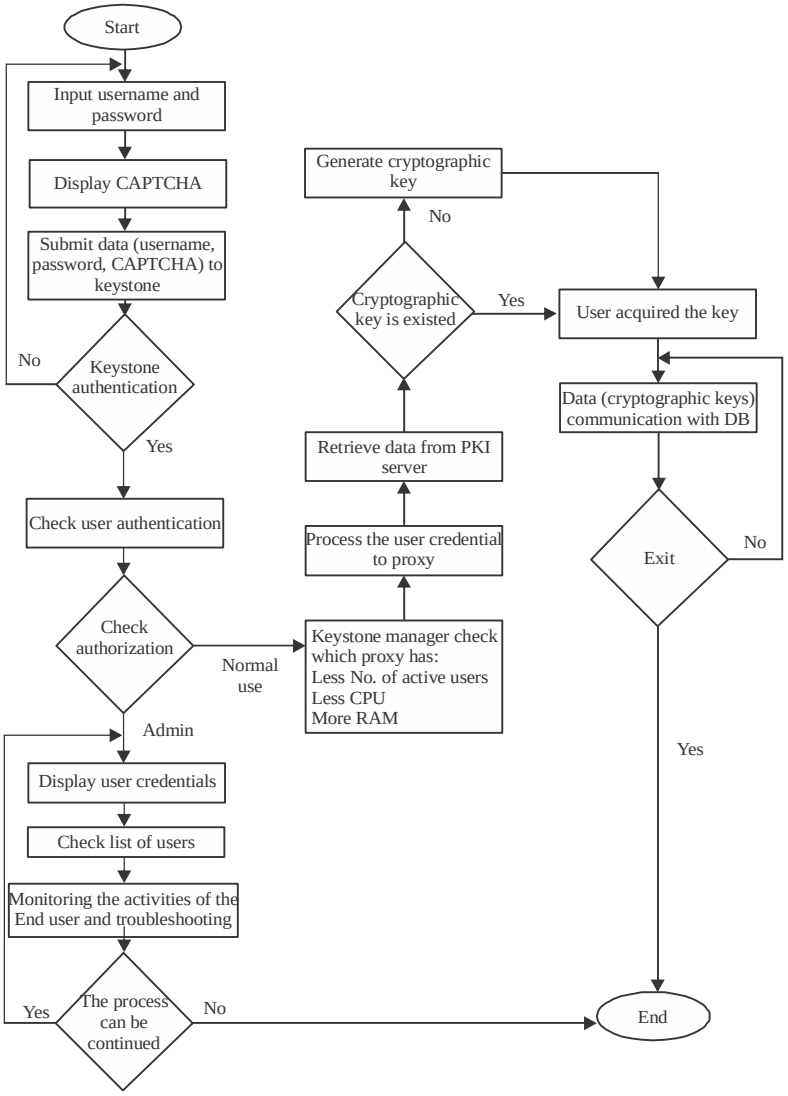

Fig. 3: Sequence diagram of the CloudSecure model

database module and the process will finish if the user wants to exit; the administrator user has the authority to check the list of users, troubleshoot any problems in the model and monitor the activities of the end user for the security purposes (Cigoj and Klobucar, 2012), the processes is depicted in Fig. 3.

RESULTS AND DISCUSSION

The experiments are conducted in real-time environment and enable a comparison between the CloudSecure model and the existing OpenStack Swift Model. Clients are tested the interaction of each entities (keystone, proxy and database repository) in each model. In the current literature, the existing OpenStack Swift Model does not provide a cryptanalysis to analyse and validate the security. Rather than the CloudSecure model have conducted the cryptanalysis to analyse and validate the security; however, the inclusion of this cryptanalysis in the existing OpenStack Swift cloud-based environments would be innovative in the authentication phase, specially, the Blowfish symmetric cryptography, data transfer process on the RSA asymmetric cryptography and data integrity on the MD5 hash function. Using the cryptanalysis will give more justification that CloudSecure security model is robust against different type of attacks. In this research, Brute force attack is conducted on the Blowfish symmetric cryptography used in the authentication process of the CloudSecure model, Birthday attack on the MD5 hash function used for data integrity in the authentication process as well as data transmission process of the

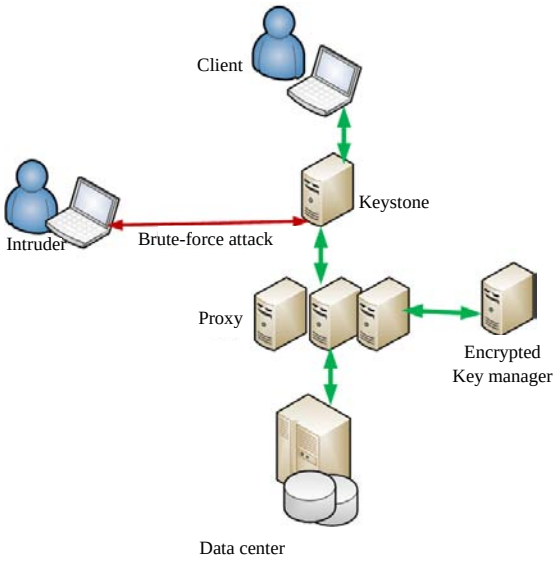

Fig. 4: Brute force attack between client and Keystone

Fig. 5: Brute force attack on the existing OpenStack Swift

Fig. 6: Brute force attack on the CloudSecure model A. birthday attack

CloudSecure model and MITM attack on the RSA asymmetric cryptography used for the secure data transmission between client and cloud proxy module.

Brute force attack: The brute force attack is used to test the strength of user credentials information of the authentication process of any models. The Brute force attack works on the keystone module and is focused only on user credential information (username and password). Then the attacker will split the connection between the client and keystone into two connections. Once the connection is intercepted, the attacker acted as a keystone for the client and acted as the client for the keystonel as illustrated in Fig. 4.

In the first scenario, there is no cryptography involve in the authentication process; in addition, the limitation of the password length is that it can be a maximum of twenty characters. This limitation means that there is no minimum threshold value for the password, if the value of the password is less than seven characters, then it is vulnerable to attack and can be easily obtained by any type of attacks. Moreover, in the existing OpenStack Swift Model token based authentication process has been applied that has been used to prove identity of the person electronically (Slipetskyy, 2011; Grobauer *et al*., 2011; Cigoj and Klobucar, 2012) as shown in Fig. 5.

In second scenario, the minimum length of the password is seven characters to make the password field more secure as compared with the existing OpenStack Swift Model (Cigoj and Klobucar, 2012; Albaroodi *et al*., 2015) and the maximum length is based on the user. This means that the longer the password is, the longer it will take to retrieve the password. In the CloudSecure model, used a three-level security (CAPTCHA, Blowfish, MD5 hash function) in the encryption using the Blowfish symmetric cryptography which will make it quite difficult for the intruder to compromise them. Now, the security layer and longer password length will make the CloudSecure model resistant to Brute force attacks whereas the single-layer security process with a short password length of the existing OpenStack Swift Model will be more vulnerable to security loopholes as indicated in Fig. 6.

Brute attack: A Birthday attack is type of cryptanalysis technique used to break the MD5 hash function by searching for collision in the cryptographic hash. This type of attack is usually used for SHA1 or MD5 hash function. The main reason of using this type of attack is exploiting both parties from doing communication. Birthday attacks are considered class of brute force attacks with advantage of having random input with return of one of the K values. Where K is the maximum number of attacks attempts to break the specific key value. The operation consists of repeatedly evaluating different inputs with same output expected. This leads us to examine popular design principles such as the MD (Merkle-Damg°ard) transform, from the point of view of balance preservation and to mount experiments to determine the balance of popular hash functions. In this research, a birthday attack is used to test the vulnerability of the MD5 hash function used in the user authentication process between the client and the Keynote as illustrated in Fig. 7. In the first scenario, the token-based security authentication has been applied between client and Keystone module. Moreover, for data integrity, no hash

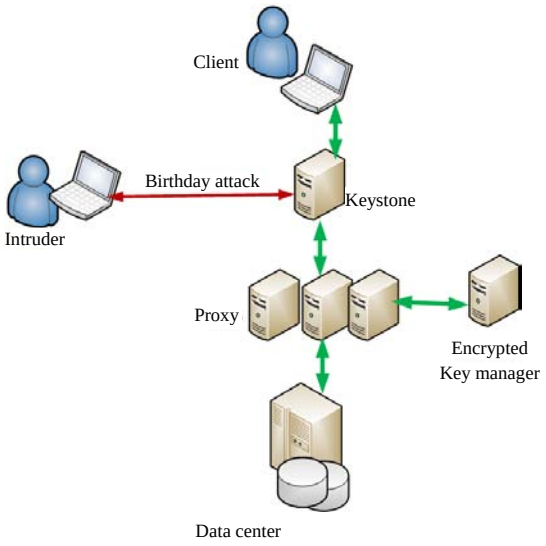

Fig. 7: Birthday attack between client and keystone

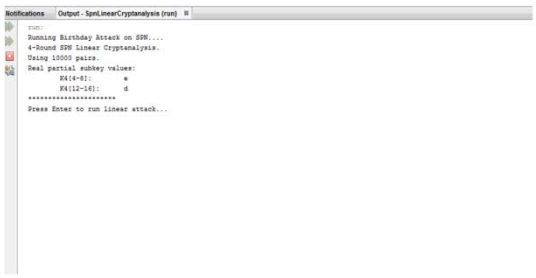

Fig. 8: Before running the birthday attack on the existing OpenStack Swift

function is applied on the client side and the keystone server side. This makes the existing OpenStack Swift Model weaker as it contains single level token security but with no data integrity feature. So, when the client send the user credential information to the keystone server, it is not been encrypted making it vulnerable to variety of attacks. Now if the attack has been done with modification of the user credentials information, then upon receiving in the keystone server side, there is no way to verify the data integrity of the received user credential information. Hence, the Birthday attack is successful on the existing OpenStack Swift model as there is no hash code applied on both sides; the client side and the keystone server side as indicated in Fig. 8 and 9.

The second scenario, encryption along with MD5 hash function is used for user authentication process to provide security and robustness, rather than the first scenario, no hash functionality is used in the user authentication process.

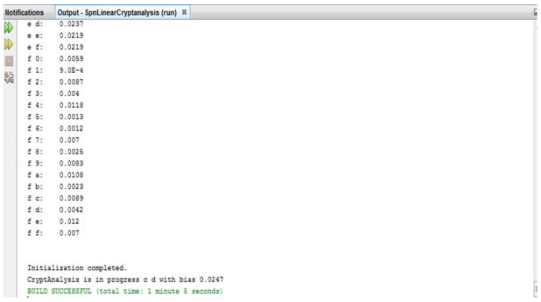

Fig. 9: After running the birthday attack on the existing OpenStack Swift

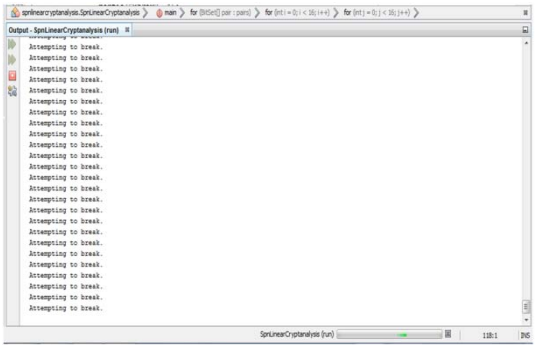

Fig. 10.: Birthday attack on the CloudSecure model A

Moreover, in the CloudSecure model the birthday attack attempted a series of attacks but all of the attacks failed. This outcome means that CloudSecure model three-level authentication security is more secured and robust compared with the existing OpenStack Swift Model (Wadhwa and Dabas, 2014). When a user enters their username, password and CAPTCHA, the CloudSecure model encrypts the data using the Blowfish symmetric cryptography algorithm and subsequently applies an MD5 hash function to maintain the security and integrity of the user data that are transmitted to the keynote module. The birthday attack will be used on the hash data generated by the user authentication information by the MD5 hash function as illustrated in Fig. 10.

Man-in-The-Middle (MITM) attack: MITM is the most important attack that performed on both models the existing OpenStack Swift Model and the CloudSecure model. The MITM attack will occur after user authentication proses and authorisation proses. Although, MITM can be applied in the authentication process but can be applied on the data transfer process as well. But the scope of this research is to use MITM during the data transfer process as the continuous stream of data communication will be processed for the uploading and the downloading of any file between the client and the

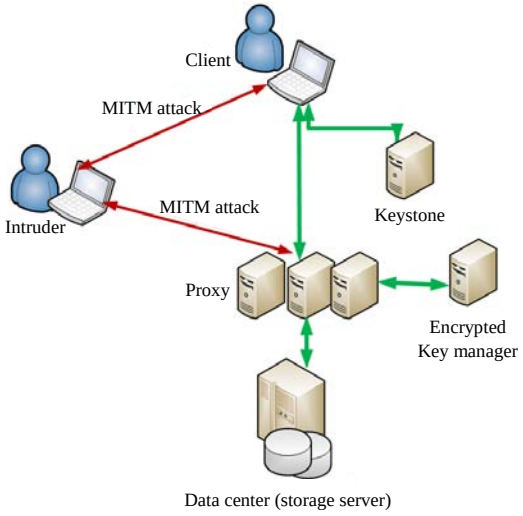

Fig. 11: MITM attack between client and proxy

proxy module in the cloud. The attacker will split the connection between the client and proxy server. Once the connection is intercepted, the attacker acted as a proxy server with the client and the attacker acted as the client with proxy server, reading, inserting and modifying the data communication as illustrated in Fig. 11.

Let's assume the value of p and q are very close to each other and then it will be easy to factor n using Fermat factorization method. According to the RSA, n = pq and supposed that the value of p is greater than q such that p>q. Then will get:

$$n = \left((p+q)/2\right)^2 - \left((p+q)/2\right)^2 \quad (1)$$

Since, the value of p and q are close to each other, so, assume s is a variable to hold the least value for p and q. Similarly, t is a variable to hold the most value for p and q. Now by calculating s and t, find is s smaller than t:

$$s = (p+q)/2 \quad (2)$$

$$t = (p+q)/2 \quad (3)$$

Then can make perfect square equation like $\sqrt{n}$ and t2-n = s2 such that the value of t will be calculated. Until t2-n is a perfect square s2. Here, $e\lceil z \rceil +$ represents the least integer $n \geq z$, p = t+s and q = t-s. MITM can use this mathematical model to estimate the value of p and q in order to break the key. Once the value of p and q has been estimated, then it will be easier to calculate n.

In the first scenario, the data transfer between the client and proxy server is without RSA asymmetric

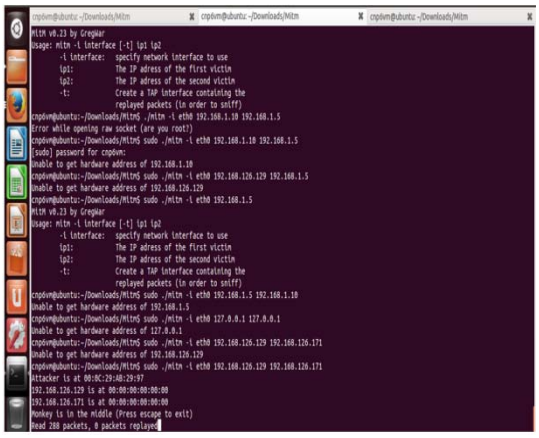

Fig. 12: MITM attack on the existing OpenStack Swift

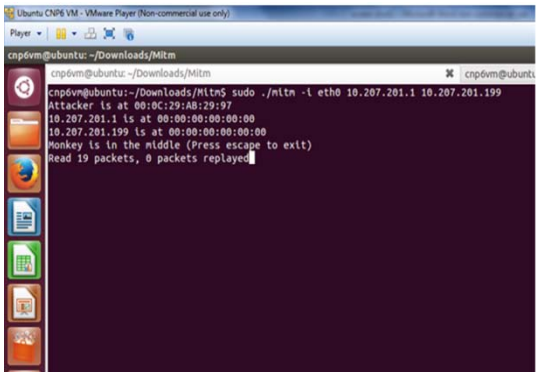

Fig. 13: MITM attack on the CloudSecure model

cryptography. As the communication between the client and the proxy server is in the unsecure communication channel, there is good chance that MITM attack will take place and the un-encrypted data will be tempered by intruder (Cigoj and Klobucar, 2012). MITM attack can capture packets from any entity in the existing OpenStack Swift Model as indicated in Fig. 12.

In the second scenario, the communication between clients will directly occur with the proxy as the keystone module using the already selected best suitable proxy based on the available hardware and software resources. In addition, RSA asymmetric cryptography has been applied which has a maximum key size of 4096 bits, notes this key size has yet to be compromised. Moreover, in the CloudSecure model, the public keys are randomly generated at regular interval of time. Upon successful authentication process, the public key will be send to the client on the secure communication chancel to the proxy module. Once the communication has ended, the generated public key will be destroyed as well. The new public key will be

generated for every new communication session as mentioned earlier. Figure 13 indicated the MITM attack on the CloudSecure model.

CONCLUSION

The findings of this research proves distinctly that there are avenues to improve the security in CC environment since data and content are hosted in a remote location under the care of a third party, i.e., CSPs. This research, addressed these concerns with the security problems that exist in the OpenStack Swift Model of CC. OpenStack and its storage component, Swift as highlighted earlier in this research is the leading cloud platform and continue to grow. The CloudSecure model implemented robust symmetric and asymmetric cryptography for the user authentication process and to address the downloading and uploading of data during data transmission between the client and a remote server. The CloudSecure model also focuses on the data integrity of the server which cannot be tampered by an anonymous user. A research evaluation is performed to solidify the justification of the CloudSecure model. A detailed discussion is conducted to analyse the robustness and reliability of the CloudSecure model in terms of security. The attributes of the CloudSecure model are compared in conjunction with the research evaluation. The future work of this research needs additional studies on the security of cloud-based services that consider other OpenStack projects such as the OpenStack Compute and OpenStack Image service. Theses sercices should also be investigated for security related problems and potential improvement of the model.

REFERENCES

Albaroodi, H., S. Manickam and M. Anbar, 2015. A proposed framework for outsourcing and secure encrypted data on OpenStack object storage (swift). J. Comput. Sci., 11: 590-597.

Albaroodi, H., S. Manickam and P. Singh, 2013. Critical review of OpenStock security: Issues and weaknesses. J. Comput. Sci., 10: 23-33.

Albaroodi, H.A., S. Manickam and M.F. Aboalmaaly, 2013. The classification and arts of open source cloud computing: A review. Adv. Inf. Sci. Serv. Sci., 5: 16-25.

Carrez, T., 2013. OpenStack object storage (swift) 1.10.0 havana. Canonical Ltd, London, England, UK. https://launchpad.net/swift/+milestone/1.10.0

Carts, D.A., 2001. A review of the Diffie-Hellman Algorithm and its use in secure internet protocols. J. SANS., 1: 1-7.

Cigoj, P. and T. Klobucar, 2012. Cloud security and OpenStack. Proceedings of the 1st International Conference on Cloud Assisted Services, October 22-25, 2012, University of Ljubljana, Bled, Slovenia, pp: 1-25.

Diffie, W. and M.E. Hellman, 1976. New directions in cryptography. IEEE Trans. Inform. Theory, 22: 644-654.

Grobauer, B., T. Walloschek and E. Stocker, 2011. Understanding cloud computing vulnerabilities. IEEE Secur. Privacy, 9: 50-57.

Kalpana, P. and S. Singaraju, 2012. Data security in cloud computing using RSA algorithm. Int. J. Res. Comput. Commun. Technol., 1: 143-146.

Kaufman, L.M., 2010. Can public-cloud security meet its unique challenges?. IEEE. Secur. Privacy, 8: 55-57.

Khan, R.H., J. Ylitalo and A.S. Ahmed, 2011. OpenID authentication as a service in OpenStack. Proceedings of the 2011 7th International Conference on Information Assurance and Security (IAS), December 5-8, 2011, IEEE, Melaka, Malaysia, ISBN:978-1-4577-2154-0, pp: 372-377.

Mitchell, T.D and P.D. Jones, 2005. An improved method of constructing a database of monthly climate observations and associated high-resolution grids. Int. J. Climatol., 25: 693-712.

Mohamed, E.M., S. El-Etriby and H.S. Abdul-Kader, 2012. Randomness testing of modern encryption techniques in cloud environment. Proceedings of the 2012 8th International Conference on Informatics and Systems (INFOS), May 14-16, 2012, IEEE, Cairo, Egypt, ISBN:978-1-4673-0828-1, pp: CC1-CC6.

Pearson, S. and A. Benameur, 2010. Privacy, security and trust issues arising from cloud computing. Proceedings of the 2010 IEEE 2nd International Conference on Cloud Computing Technology and Science, November 30-December 3, 2010, IEEE, Indianapolis, Indiana, ISBN:978-1-4244-9405-7, pp: 693-702.

Ren, K., C. Wang and Q. Wang, 2012. Security challenges for the public cloud. IEEE Internet Comput., 16: 69-73.

Ristov, S., M. Gusev and A. Donevski, 2014. Security vulnerability assessment of OpenStack cloud. Proceedings of the 2014 6th International Conference on Computational Intelligence, Communication Systems and Networks, May 27-29, 2014, IEEE, Tetova, Macedonia, ISBN:978-1-4799-5076-8, pp: 95-100.

Ryan, P.Y.A., S. Schneider and V. Teague, 2015. End-to-end verifiability in voting systems, from theory to practice. IEEE. Secur. Privacy, 13: 59-62.

Saxena, A., N.S. Chauhan, S.K. Reddy, A.S. Vangal and D.P. Rodrguez, 2012. A new scheme for mobile based CAPTCHA service on cloud. Proceedings of the 2012 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), October 11-12, 2012, IEEE, Bangalore, India, ISBN:978-1-4673-4421-0, pp: 1-6.

Slipetskyy, R., 2011. Security issues in OpenStack. Master's Thesis, Department of Telematics, Norwegian University of Science and Technology, Trondheim, Norway.

Wadhwa, D. and P. Dabas, 2014. A coherent dynamic remote integrity check on cloud data utilizing homomorphic cryptosystem. Proceedings of the 2014 5th International Conference on Confluence the Next Generation Information Technology Summit (Confluence), September 25-26, 2014, IEEE, Noida, India, ISBN:978-1-4799-4236-7, pp: 91-96.