International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 3 | ISSN : 2456-3307

Enhancement of Public Cloud, Application Security, Using Bcrypt Algorithm

Shri Ranjani. N*[1], Ravikumar. B[2]
*[1] Department of Computer Science and Engineering, Velammal Engineering College, Chennai
[2] Assistant Professor, Department of Computer Science and Engineering, Velammal Engineering College, Chennai

ABSTRACT

The emergence of internet has lead to the booming of internet based applications and services. One such domain which has developed at an exponential rate is cloud computing. Nowadays, most of the applications run on cloud environments. Even a highly critical application such as banking applications use cloud backends for storing data. But, such applications, which has a very high confidential data do not run on private clouds. The use of private cloud computing may not be possible for all applications as it requires more cost for creating a cloud environment and maintaining it. We can have our application in public clouds if the data stored are not highly confidential. This is where application level security in cloud comes into play. In this paper, we deal about the enhancement of application level security in any application where data resides in a public cloud. Here, we use a highly complicated cryptographic algorithm, Bcrypt, which is used to store passwords.

Keywords: Bcrypt, Public cloud, Security, Cryptography, Hashing

I. INTRODUCTION

Many applications are being deployed to cloud environment, nowadays. The most common thing among these applications would be an authentication page, where a user would be prompted to enter login credentials. We will be seeing applications which forces the user to use a strong passwords with criterias such as Alphanumeric, Non-sequential etc ., in-order to increase the level of security of the application and the data present in the application.

As the speed of processors have grown in an enormous rate and it is predicted to have a large amount of growth in the future, password cracking or guessing has become very fast and easy. So, it is the duty of the application to keep the user passwords safer. We have enormous hardware support to achieve this. Therefore, it is high-time that we use a strong and complex password hashing scheme like Bcrypt, which is derived from Blowfish cryptographic algorithm.

In the upcoming sections, we discuss the existing mechanisms to store password in any application. And also, we get to understand how the data are stored in variety of cloud platforms, provided by different vendors.

II. METHODS AND MATERIAL

There are several password hashing mechanisms existing in the market such as SHA(Secure Hash Algorithm), MD(Message Digest) and so on.

Bcrypt uses a configurable iteration count. A single time bcrypt invocation is 10 times a MD5 invocation. This means that, using the brute-force approach to crack a password will be 10 times more expensive with bcrypt than with MD5.

The functionality, configurable slowness of Bcrypt, is that what it differs from other hashing algorithms. The password hashing function can be made how much ever speedness you want. To be precise, as slow as you can tolerate: indeed, a slow function is slow for everybody, attacker and defender alike.

Due to the kind of operations that are used within the Bcrypt algorithm, we can see that it is slower even in GPU based systems. Whereas, MD5 is, by comparison, very easy to implement and can efficiently run on a GPU. This means that usage of GPU in MD5 might enhance the speed of password cracking; while with the bcrypt the need to use the same kind of CPU as the defender should be met by the attacker.

Now coming to the SHA-512 algorithm, it is also a fast hashing algorithm like MD-5. This means, once again the attacker can crack the password faster and easier.

The main purpose of this algorithm is to prevent the rainbow table attacks. That is, even if the attacker gets to know the password of one user, he may not be able to crack the other user's passwords stored. This is made possible by using the mechanism called as Salting. Salting is nothing but a mechanism by which a random string called salt is appended to the original string/password. In Bcrypt, a separate complex algorithm is used, to generate these salts.

One of the major advantages of the Bcrypt password hashing scheme is that, it is future adaptable. This means that the complexity of the algorithm can be increased by increasing the number of rounds in the salt generation algorithm.

Blowfish has a block size of 64-bit and a key length which varies from 32 bits up to 448 bits. It uses large S-boxes and uses a Feistel structure of 16 rounds. First we have to setup the key schedule of the blowfish algorithm. For that, we first initialize the P-array and S-boxes which does not have a pattern. This randomness is brought by initializing them with hexadecimal digits of the pi value. The secret key is XORed with all the P-entries in order. Then a 64-bit all-zero block is encrypted with the algorithm. The P1 and P2 are replaced by the resultant ciphertext. The same ciphertext is then encrypted again with the new subkeys. Now, the P_3 and P_4 are replaced by the resultant new cipher text. This continues until the entire P-array and all the S-box entries are replaced.

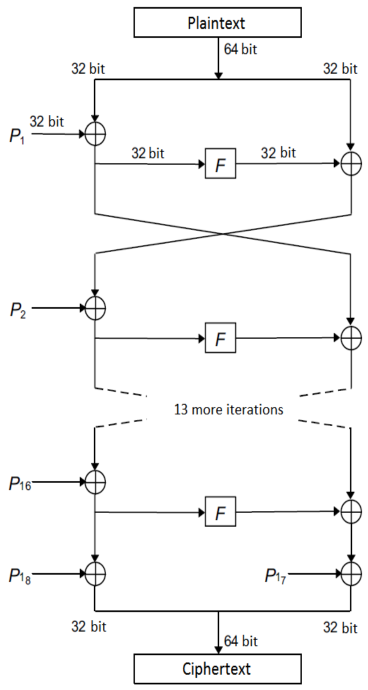

Figure 1. Blowfish feistel structure

The Bcrypt algorithm uses a setup with expensive keys known as eksblowfish which is very similar to the normal blowfish algorithm.This setup requires three inputs a cost, salt and key(which is used for encryption).

The cost parameter determines the effectiveness/ complexity/expensiveness of the eksblowfish algorithm. In this algorithm, a salt is a 128 bit value, which is different for different key. This means that no two secret keys will have the same salt generated. Even in the salt generation, the user is given an option to choose his own number of rounds. The higher the number of rounds the more complex the resultant salt will be.

This highly complex structure requires more powerful hardware. As the cloud servers have a really high power, it can be utilised to the fullest, thus increasing the security of the application.

A dictionary attack is one in which the attacker determines/precomputes the list of possible keys and then use them all to determine the password. In bcrypt algorithm, this attack is not feasible. This is because, such type of attacks are possible only in one way functions which can be computed easily.

Now let us compare the security features in the public cloud platforms provided by different cloud vendors/providers. Some of the providers are Google, Amazon, IBM and so on.

CLOUD PROVIDER	ENCRYPTION ALGORITHM
Google	AES
Amazon	AES 256-bit encryption
IBM	Advanced Encryption Standard (AES) 256-bit encryption and Secure Hash Algorithm (SHA)-256 hash

Figure 2. Comparison of encryption algorithm used by different cloud providers/vendors

If we use the highly expensive, Bcrypt algorithm in a partially secured environment, the password cracking becomes highly time consuming for the attacker who is trying to attack the system. Thus, we can enhance the security of the application with the help of highly fast computing feature of a cloud platform.

III. CONCLUSION

In this paper, we have discussed about the different password hashing algorithms existing in the market and the different cloud environments provided by different vendors. As the future trend relies mostly on the cloud platforms and environments, it is very important for us to move towards that trend, without compromising the security feature of the application. As the processing speed increases, it is our duty to utilise the product to the fullest. If we fail to use the computational power wisely, the application will not be able to adapt to the future. From this, we can conclude that the Bcrypt algorithm, which has a highly expensive setup can be used to enhance the security feature of any cloud deployed application.

IV. REFERENCES

[1]. Provos, Niels, Mazières, David, Talan Jason Sutton 2012 (1999). "A Future-Adaptable Password Scheme". Proceedings of 1999 USENIX Annual Technical Conference: 81–92.

[2]. Antony G. Robertiello and Kiran A. Bandla. 2005."Attacks on MD5 Hashed Passwords," Technical Report, George Mason University, USA.

[3]. C.H. Chen, G. Horng and C.H. Hsu. 2009. "A novel private information retrieval scheme with fair privacy in the user side and the server side", Int. J. Innovat. Comput. Inf. Control, Vol. 5, No. 3, pp. 801–810.

[4]. Shai Halevi and Hugo Krawczyk. "Public Key Cryptography and password protocols", Proceedings of the 5th ACM Conference on Computer and Communications Security,1998.

[5]. M. Dürmuth, T. Güneysu, M. Kasper, C. Paar, T.Yalçin, and R. Zimmermann. 2012. "Evaluation of Standardized Password-Based Key Derivation against Parallel Processing Platforms," in Computer Security– ESORICS 2012, pp. 716–733

[6]. Sarvar Patel. "Number theoretic attacks on secure password schemes". In Proceedings of the 1997 IEEE Symposium on Security and Privacy, Oakland,CA,May 1997

[7]. Chad R, Dougherty (31 Dec 2008). "Vulnerability Note VU#836068 MD5 vulnerable to collision attacks". Vulnerability notes database. CERT Carnegie Mellon University Software Engineering Institute. Retrieved 3 February 2017.

[8]. Philip Hawkes and Michael Paddon and Gregory G. Rose: "Musings on the Wang et al. MD5 Collision", 13 October 2004. Retrieved 27 July 2008.

[9]. Stevens, Marc; Bursztein, Elie; Karpman, Pierre; Albertini, Ange; Markov, Yarik. "The first collision for full SHA-1"

[10]. Sotirov, Alexander, Stevens, Marc, Appelbaum, Jacob, Lenstra and Arjen et al. 2008. MD5 considered harmful today (online), Technische Universiteit Eindhoven.

[11]. Thomas Wu.The secure remote password protocol. In Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium March 1998

[12]. Google Cloud Platform Documentation, https://cloud.google.com/datastore/docs

[13]. IBM Bluemix Documentation, https://console.bluemix.net/docs

[14]. Amazon Web Service Documentation, https://docs.aws.amazon.com