DEFENDING AGAINST SQL INJECTION ATTACK IN CLOUD COMPUTING

P.Ponmuthukalaiselvi
(Department of CSE,National Engineering College,Kovilpatti)

S.K.Santha
(Department of CSE,National Engineering College,Kovilpatti)

Mrs.B.Shunmugapriya
(AP,Department of CSE,National Engineering College,Kovilpatti)

Abstract:

Cloud Computing is an emerging paradigm that allows customers to obtain cloud resources and services according to an on-demand, self-service, and pay-by-use business model. There are number of web application threats in cloud computing one among them is SQL injection attack (SQLiA). In this the attackers produce a query of their interest to have illegal access to the database. In order to prevent this we use Twofish encryption algorithm is used to secure the clients sensitive information. In this algorithm the file uploaded by the data owner is encrypted using Twofish algorithm and then stored it in a database.

Keywords **—Cloud computing, SQL injection attack (SQLiA), twofish encryption and decryption.**

I. INTRODUCTION

Cloud computing pretends a major change in storing information and run applications. Instead of running programs and data on an individual desktop computer, everything is hosted in the "cloud"—a nebulous assemblage of computers and servers accessed via the Internet. Through cloud computing wecan access all our applications and documents from anywhere at anytime in different locations for collaboration.

Cloud is hosted by Google that consists of both small PCs and larger servers. Cloud of Google is a private one that can be publicly accessed by the Google users. Cloud can be extended beyond a single company or enterprises. Cloud serves the applications and data which are available to users, cross enterprises and cross-platforms. Cloud can be accessed via the Internet. Any authorized user can access these docs and apps can be accessed by any authorized users from any computer. But the technology and infrastructure behind the cloud is invisible to the user.

There are six key properties of cloud computing:

- User-centric
- Task-centric
- Powerful
- Accessible
- Intelligent
- Programmable

A. Cloud Computing: The Next Step in Collaboration

For cloud based projects, the users can collaborate from multiple locations within the corporation and from multiple organizations. Google has a collection of servers which is used to power its massive search engine. On the infrastructure side, IBM, Sun Systems, and other big iron providers are offering the hardware necessary to build cloud networks. On the software side, dozens of companies are developing cloud-based applications and storage services.

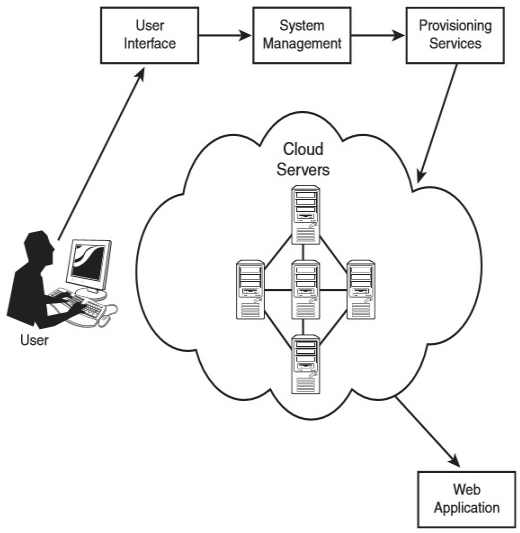

Fig.1. Architecture of cloud computing

B. Cloud Storage

Cloud computing is mainly used for data storage. Rather than using dedicated servers for storing data, multiple third-party servers in cloud storage is used. When storing data, the user sees a virtual server—that is, it appears as if the data is stored in a particular place with a specific name. But that place doesn't exist in reality. In reality, the user's data could be stored on any one or more of the computers used to create the cloud. The cloud dynamically manages available storage space. The advantages of Cloud storage is based on both financial and security associates. For financial, virtual resources in the cloud are typically cheaper than dedicated physical resources connected to the pc or network. The data stored in the cloud is secure from accidental erasure which is related to security. If one machine in the cloud crashes, the data is duplicated on other machines in the cloud.

C. SQL Injection Attack (SQLIA) Process

The websites like, data driven are vulnerable to SQL Injection attack, where database is a black box in three tier architectures. In this architecture the SQL statements are generated in response to HTTP requests.These HTTP request may contain parameters that are used by attackers to produce a query of their interest to have illegal access to the database as shown in Fig. 2.

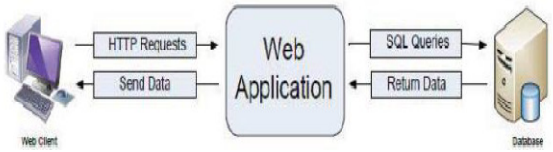

Fig. 2. SQL Injection attack process.

Fig. 4 shows the log in page which is most vulnerable for the SQL injection attack and the following PHP code snippet produces dynamic query in response to user input as shown in Fig. 5and 6.

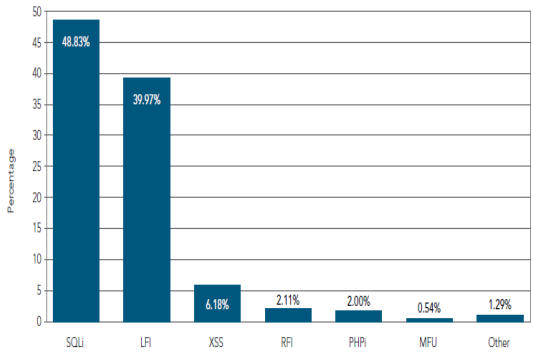

Fig.3.Web Application Attack Frequency

Fig. 4.Log In form.

```
//connect to a database
mysql_connect(servername,username,password);
//store user input in the variables collected from the user input login form
$username=$_POST[username];
$password=$_POST[password];
//dynamically build the query from the user input
$query="SELECT"FROM the_users WHERE username="$username" AND password="$password"
//execute a query
$result-mysql_query($query);
If($result)
```

```
        return true;
else
        return false;
```

Fig. 5. PHP Code snippet to generate dynamic query in response to user input.

```
SELECT *FROM tbl users WHERE
username='user_Name AND PASSWORD='pwd';
```

Fig. 6. SQL query as a result of code.

In next Fig.8 at the same form user try to attempt a simple SQLIA to bypass the authentication.

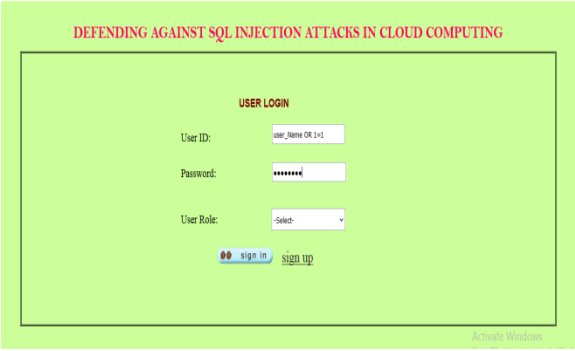

Fig. 7.Simple attempt for SQLIA.

```
SELECT *FROM tbl users WHERE
usernamae='user_Name OR 1=1 AND
password='Whatever'
```

Fig. 8.Dynamic generated query in response to above input.

In Fig. 8 attacker try to ignore the password by using the comment operator as everything would be ignored after the comments operator even the password. In this scenario user name is tried to be true using the OR operator. This, the simple scenario and with different techniques intruders want to add query of their interest to have access to the information of their interest.

D. Techniques of SQL Injection Attack (SQLIA)

1) Tautology Based Attacks
In tautology attack, malicious contents are added using the conditional statement that always evaluate to true. Previous scenario is the perfect example of this attack.
Select * from tbl users Where username='raja' or '1'='1' and password ='anything'

2) Union Attack
In this technique, malicious query is added with the safe query using the UNION keyword .
['UNION SELECT pwd FROM user-info WHERE id='abc' and pwd='']

3) Logically Incorrect query Attack
In this type of technique logically incorrect type of query is performed to have information about some structures of the data base to proceeds further.

4) Piggybacked Query
Certain delimiters like ",",","," used to join the legitimate query with the illegitimate one.
Select * from users where id='raja' and pwd=''; Drop table users…'

5) Alternate Encoding
By changing the coding schema, the illegal query can be bypassed through the filter that tests the legitimacy of the query.

6) Inference Attack
Blind and timing techniques are used in inference attacks. In blind attack, a series of the simple queries are performed to have guess about the structure of the data base. In timing attack the query processing time is observed to infer some information presence in the data base.

E. Consequences of SQL Injection Attacks

To gain information about data base finger prints like type of data base, SQL language used, etc. This information helps the attacker to proceeds or use more sophisticated attacks.
1) To gain information about user credentials.
2) To get the database schema.
3) To extract and modify the data base.
4) To perform Denial of Services like shutting down the data base, dropping tables, etc.
5) Replacements of files with false or tempered information.
6) Execution of remote commands.
7) Shop lifting, account balance change.
8) Interacting with underlying operating system.

II. TWOFISH ENCRYPTION ALGORITHM

Twofish is our submission to the AES selection process. It meets all the required NIST criteria—128- bit block; 128-, 192-, and 256-bit key; efficient on various platforms; etc.—and some

strenuous design requirements, performance as well as cryptographic, of our own. Twofish can:
• After a 12700 clock-cycle key setup, the data can be encrypted at 285 clock cycles per block on a Pentium Pro.
• After a 1250 clock-cycle key setup, the data can be encrypted at 860 clock cycles per block on a Pentium Pro.
• After a 1750 clock-cycle key setup, the data can be encrypted data at 26500 clock cycles per block on a 6805 smart card.

A.TWOFISH DESIGN GOALS

Twofish was designed to meet NIST's design criteria for AES. Specifically, they are:
•A 128-bit symmetric block cipher.
• Key lengths of 128 bits, 192 bits, and 256 bits.
• No weak keys.
• Efficiency, both on the Intel Pentium Pro and other software and hardware platforms.
• Flexible design: e.g., accept additional key lengths; be implementable on a wide variety of platforms and applications; and be suitable for a stream cipher, hash function, and MAC.
• Simple design, both to facilitate ease of analysis and ease of implementation.
Additionally, we imposed the following performance criteria on our design:
• Accept any key length up to 256 bits.
• Encrypt data in less than 500 clock cycles per block on an Intel Pentium, Pentium Pro, and Pentium II, for a fully optimized version of the algorithm.
• Be capable of setting up a 128-bit key (for optimal encryption speed) in less than the time required to encrypt 32 blocks on a Pentium, Pentium Pro, and Pentium II.
• Encrypt data in less than 5000 clock cycles per block on a Pentium, Pentium Pro, and Pentium II with no key setup time.
• Not contain any operations that make it inefficient on other 32-bit microprocessors.
• Not contain any operations that make it inefficient on 8-bit and 16-bit microprocessors.
• Not contain any operations that reduce its efficiency on proposed 64-bit microprocessors; e.g., Merced.
• Not include any elements that make it inefficient in hardware.
• Have a variety of performance tradeoffs with respect to the key schedule.
• Encrypt data in less than less than 10 milliseconds on a commodity 8-bit microprocessor.
• Be implementable on a 8-bit microprocessor with only 64 bytes of RAM.
• Be implementable in hardware using less than 20,000 gates.
Our cryptographic goals were as follows:
• 16-round Twofish (without whitening) should have no chosen-plaintext attack requiring fewer than 280 chosen plaintexts and less than 2 N time, where N is the key length.
• 12-round Twofish (without whitening) should have no related-key attack requiring fewer than 264 chosen plaintexts, and less than 2N/2 time, where N is the key length.

B.TWOFISH

Twofish is a symmetric key block cipher with a block size of 128 bits and key sizes up to 256 bits. Twofish is related to the earlier block cipher Blowfish.

Two fish's distinctive features are the use of pre-computed key-dependent S-boxes, and a relatively complex key schedule. One half of an n-bit key is used as the actual encryption key and the other half of the n-bit key is used to modify the encryption algorithm (key-dependent S-boxes). Twofish borrows some elements from other designs; for example, the pseudo-Hadamard transform (PHT) from the SAFER family of ciphers. Twofish has a Feistel structure like DES. Twofish also employs a Maximum Distance Separable matrix.
Twofish is a Feistel network. This means that in each round, half of the text block is sent through an F function, and then XORed with the other half of the text block. DES is a Feistel network. Blowfish (another Schneier algorithm) is a Feistel network. Five of the AES submissions are Feistel networks. Feistel networks have long been studied in cryptography, and we know how they work.

$$F: \{0,1\}^{n/2} X \{0,1\}^{N} \rightarrow \{0,1\}^{n/2}$$

Fig.11. Fiestel structure for twofish

In each round of Twofish, two 32-bit words serve as input into the F function. Each word is broken up into four bytes. Those four bytes are sent through four different key-dependent S-boxes. The four output bytes (the S-boxes have 8-bit input and output) are combined using a Maximum Distance Separable (MDS) matrix and combined into a 32-bit word. Then the two 32-bit words are combined using a Pseudo-Hadamard Transform (PHT), added to two round

subkeys, then XORed with the right half of the text. There are also two 1-bit rotations going on, one before and one after the XOR. Twofish also has something called "prewhitening" and "postwhitening;" additional subkeys are XORed into the text block both before the first round and after the last round.

Each step of the round function is bijective. That is, every output is possible. We've seen too many attacks against ciphers that don't have this property not to include it. The round function mixes up operations from different algebraic groups: S-box substitution, an MDS matrix in GF(2^8), addition in GF(2^{32}), addition in GF(2) (also called XOR), and 1-bit rotations. This makes the algorithm difficult to attack mathematically.

The key-dependent S-boxes are designed to be resistant against the two big attacks of the early 1990s -- differential cryptanalysis and linear cryptanalysis -- and resistant against whatever unknown attacks come next. Too many algorithm designers optimize their designs against specific attacks, without thinking about resistance against the unknown. Our design philosophy was a bit different: good enough against known attacks, and enough nastiness to (hopefully) resist unknown attacks. Key-dependent S-boxes were one way we did that.

Key-dependent S-boxes were not selected randomly, as they were in Blowfish. Instead, we carefully designed S-box construction rules, and tested them with all possible 128-bit keys (and a subset of possible longer keys) to make sure that all the S-boxes were indeed strong. This approach allowed us to combine the strength of fixed, strong S-boxes with the strength of secret S-boxes. And Twofish has no weak keys, as Blowfish does in reduced-round variants.

The MDS matrix was carefully chosen to provide good diffusion, to retain its MDS property even after the 1-bit rotation, and to be fast in both hardware and software. This means that we had to search through all possible matrices and find the one that best met our criteria.
The PHT and key addition provide diffusion between the subblocks and the key. And using the LEA instruction on the Pentium (and above), we can do all four additions in just two operations.
The round subkeys are carefully calculated, using a mechanism similar to the S-box construction rules, to prevent related-key attacks and to provide good key mixing. One of the things we learned during this process is that a good key schedule is not grafted onto a cipher, but designed in tandem with the cipher. We spent a lot of time on the Twofish key schedule, and are proud of the results.

The 1-bit rotation is designed to break up the byte structure; without it, everything operates on bytes. This operation exists to frustrate cryptanalysts; it certainly frustrated our attempts at cryptanalyzing Twofish.

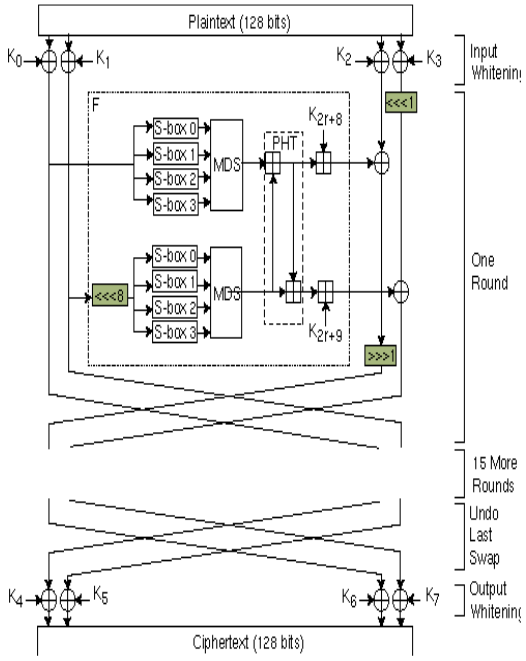

Fig.9.TWOFISH FUNCTIONALBLOCK DIAGRAM

The prewhitening and postwhitening seems to add at least a round to the difficulty of any attack. Since eight XORs are cheaper than a round, it makes sense to leave them in.

C. TWOFISH'S PERFORMANCE

Twofish has a variety of options. You can take longer for key setup and the encryption runs faster; this makes sense for encrypting large amounts of plaintext with the same key. You can setup the key quickly and encryption is slower; this makes sense for encrypting a series of short blocks with rapidly changing keys.

On smart cards, Twofish also has a variety of trade-offs. The RAM estimates assume that the key must be stored in RAM. If the key can be stored in EEPROM, then the algorithm only needs 36 bytes of RAM to run. The code size includes both encryption and decryption code. If only encryption has to be implemented, the code size and speed numbers improve somewhat.

Key setup on this processor is about 1750 clocks per key, which can be cut considerably at the cost of two additional 512-byte ROM tables. And the 6805's lack of a second index register has a significant impact onthe code size and performance of Twofish; a CPU with multiple index registers (the 6502, for instance) will be a better fit for the algorithm.

These estimates are for a 128-bit key. For larger keys, the extra code size is negligible: less than 100 bytes for a 192-bit key, and less than 200 bytes for a 256-bit key. The encryption time increases by less than 2600 clocks for a 192-bit key, and about 5200 clocks for a 256-bit key. Similarly, the key schedule precomputation increases to 2550 clocks for a 192-bit key, and to 3400 clocks for a 256-bit key.

The plaintext is split into four 32-bit words. In the input whitening step, these are XORed with four key words. This is followed by sixteen rounds. In each round, the two words on the left are used as input to the *g* functions. (One of them is rotated by 8 bits first.) The *g* function consists of four byte-wide key-dependent S-boxes, followed by a linear mixing step based on an MDS matrix. The results of the two *g* functions are combined using a Pseudo-Hadamard Transform (PHT), and two keywords are added. These two results are then XORed into the words on the right (one of which is rotated left by 1 bit first, the other is rotated right afterwards). The left and right halves are then swapped for the next round. After all the rounds, the swap of the last round is reversed, and the four words are XORed with four more key words to produce the ciphertext. More formally, the 16 bytes of plaintext p_0…… p_{15} are first split into 4 words P_0,..., P_3 of 32 bits each using the little-endian convention.

$$P_j = \sum_{j=0}^{3} P(4i+j)2^{8j} \quad i=0,\ldots\ldots,3$$

In the input whitening step, these words are XORed with 4 words of the expanded key.

$$R_{0,i} = P_i \oplus K_i \qquad i = 0,\ldots,3$$

In each of the 16 rounds, the first two words are used as input to the function *F*, which also takes the round number as input. The third word is XORed with the first output of *F* and then rotated right by one bit. The fourth word is rotated left by one bit and then XORed with the second output word of *F*. Finally, the two halves are exchanged.

$$(F_{r,0}, F_{r,1}) = F(R_{r,0}, R_{r,1}, r)$$

$$R_r + 1,0 = ROR(R_{r,2} \oplus F_{r,,0}, 1)$$
$$R_r + 1,1 = ROL(R_{r,3}, 1) \oplus F_{r,1}$$
$$R_r + 1,2 = R_{r,0}$$
$$R_r + 1,3 = R_{r,1}$$

for r = 0…… 15 and where ROR and ROL are functions that rotate their first argument (a 32-bit word) left or right by the number of bits indicated by their second argument. The output whitening step undoes the `swap' of the last round, and XORs the data words with 4 words of the expanded key.

$$C_i = R_{16,(i+2) mod\ 4} \oplus K_{i+4} \quad i = 0,\ldots,3$$

D. CRYPTANALYSIS OF TWOFISH

Our best attack works against five rounds of Twofish, without the prewhitening and postwhitening. It requires $2^{22.5}$ chosen plaintext pairs and 2^{51} work. We expect further research and clever techniques will extend this attack a few more rounds, but don't believe that there are any attacks against more than nine or 10 rounds.

We also have a related-key attack. It's a partial chosen-key attack on 10 rounds of Twofish without the prewhitening and postwhitening. To mount the attack, we have a pair of related keys. We get to choose 20 of the 32 bytes of each key. We have complete control over those 20 bytes of both keys. We don't know the remaining 12 bytes of key, but we do know that they are the same for both keys. We end up trying about 2^{64} chosen plaintexts under each key, and doing about 2^{34} work, to recover the remaining unknown 12 bytes of key. No, it's not a terribly realistic attack, but it's the best we can do.And we have reduced-round attacks on simplified variants: Twofish with fixed S-boxes, Twofish without the 1-bit rotations, and so on.

III.SURVEY PAPER

Ensuring data storage security in cloud computing

The main objective of this paper is to solve the security issues that are to prevent unauthorized access, it can be done with the help of a distributed scheme by using homomorphism token to provide security of the data in cloud.

Drawbacks:

The servers are must required to operate on specifiedrows to check correctness and verification for the calculation of requested token.

Privacy-preserving public auditing for data storage security in cloud computing

This paper proposes a secure cloud storage system that supporting privacy-preserving public auditing. The TPA is to perform audits for multiple users simultaneously and efficiently.

Drawbacks:

It can provide weaker security models.

Hidden attribute-based signatures without anonymity revocation

This paper presents hidden attribute based signature frompairings.

Drawbacks:

It can provide some reset attacks.

Dynamic audit services for integrity verification of outsourced storages in clouds

This paper proposes a dynamic audit service for verifying theintegrity of untrusted and outsourced storage.

Drawbacks:

It can provide a small, constant amount of overhead.

Provable data possession at untrusted stores

Thispaper introduce a model for provable data possession(PDP) that allows a client that has stored data at an untrustedserver to verify that the server possesses the original datawithout retrieving it. The model generates probabilistic proofsof possession by sampling random sets of blocks from theserver, which drastically reduces I/O costs.

Drawbacks:

It must require a small, constant amount of com-munication per challenges.

IV.PROPOSED SOLUTION

In this article, a solution is proposed that is based ontwofish encryption algorithm.

A. Proposed Solution Architecture

Proposed model based on thetwofish that works on the owner who upload the data will be encrypted from the vulnerabilities that occur from the hacker. While encrypting the key that is generated will be visible only to the user. When the data user wish to download the file uploaded by the user he request the generated key to the owner . By accepting the request the owner provide it to the user to decrypt the file that is downloaded.The file uploaded by the owner will be stored as encrypted file.So that the attacker cannot hack the data from the database.

The complete flow of proposed solution is shown in Fig 11.

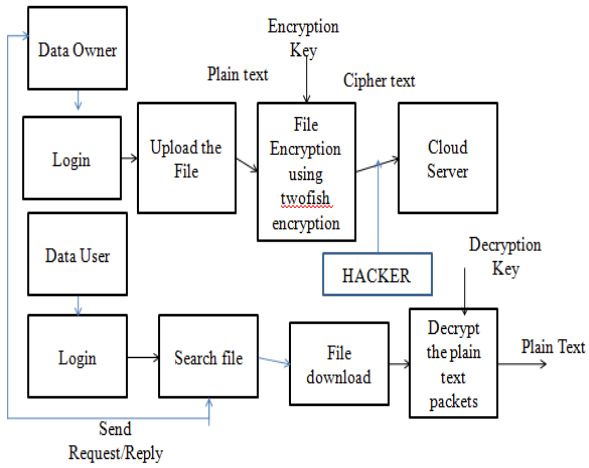

Fig.11.Data flow diagram for proposed solution.

B.ALGORITHM OF PROPOSED SOLUTION

```
Private string Encrypt(string clearText)
{
  string
EncryptionKey=”MAKV2SPBNI99212”;
  byte[] clearBytes=Encoding.Unicode.GetBytes

(clearText);
  using(Aesencrytor = Aes.create())
 {
  Rfc2898DeriveBytes pdb = new
Rfc2898DeriveBytes
  (EncryptionKey,new byte[] {0x49, 0x76 ,
0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65,
0x64,0x65,0x76});
encryptor . Key = pdb.GetBytes(16);
  using (MemoryStreamms = new
MemoryStream())
  {
   using (CryptoStreamcs = new CryptoStream
(ms,
encryptor.createEncryptor(),
CryptoStream.Write))
  {
cs .Write(clearBytes,0,clearBytes.Length);
cs .Close();
  }
clearText =
Convert.ToBase64String(ms.ToArray());
  }
  }
 return clearText;
  }
```

C.IMPLEMENTATION AND EVALUATION OF PROPOSED SOLUTION

To evaluate the proposed solution its performance is compared with the previous sections. These algorithm and proposed solutions are applied todetect the SQLIA and block the SQLIA in different types of web application specified in Table 1.

1) IMPLEMENTATION

Using ASP.Net different classes of web Application are used to evaluate the different tools against the different SQL Injection Attack.

Using ASP.Net different classes of web Application are used to evaluate the different tools against the different SQL Injection Attack.

2) EVALUATION SCENERIOS

Following criteria are used to judge the performance of twofish encryption algorithm.

1) Registration of data owner or data user
2) Uploading the file to encrypt
3) Encryption of data in the file
4) Downloading the file
5) Decryption of the file
6) Data stored in the database

Following dataset is used to evaluate the above-mentioned conditions.

Applications	No Of Inputs
Portal	100
Online Shopping	100
University Database	100
Financial Database	100

D. Evaluation Results

Fig.12. Registration process for proposed solution

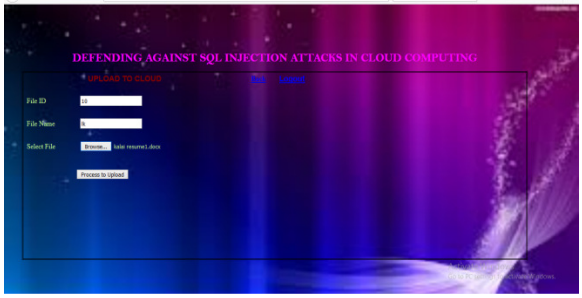

Fig.13. Uploading the file for encryption

Fig.14. Encryption of file using twofish algorithm

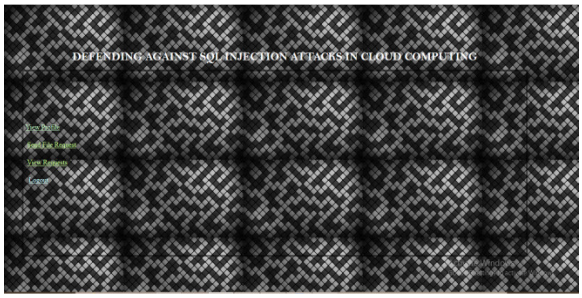

Fig.15. Data user for downloading the file

Fig.16. Server in cloud computing

V.CONCLUSION

Among many other web application threats SQL Injection Attack has emerged as major threats. Many solutions were proposed to detect the SQLIA vulnerabilities in web application. Proposed solution based on twofish algorithm has performed well to detect and block the SQLIA. One major advantage of

the proposed solution is that it can handle the advanced SQLIA techniques as knowledge base is updated to handle modern types of threats.

VI.REFERENCES

1. Ciampa, C. A. Visaggio and M. D. Penta, "A heuristic-based approach for detecting sql-injection vulnerabilities in web applications," in In Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems, SESS "10, New York, NY, USA,2010.

2. A. Tajpourand ..Shooshtari, "Evaluation of sql injection detection and prevention techniques," in Computational Intelligence, Communication Systems and Networks (CICSyN), 2010 Second International Conference,2010.

3. A. Ciampa, C. A. Visaggio and M. D. Penta, "A heuristic-based approach for detecting SQL-injection vulnerabilities in web applications," in Proceeding SESS '10 Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems, New York,2010.

4. B. Indrani and E. Ramaraj., "X–log authentication technique to prevent sql injection attacks," International Journal of Information Technology and Knowledge Management ., vol. 4, pp. 4:323–328,,2011.

5. A. Tajpour, S. Ibrahim and M. Masrom, "SQL injection Prevnetion and detection Techniques," International Journal of Advancements in Computing Technology, vol. 3, no. 7, pp. 85-91, August2011.

6. D. Das, U. Sharma and D. Bhattacharyya, "An approach to detection of sql injection attack based on dynamic query matching," International Journal ofComputer

7. A. Moosa, "Artificial Neural Network based Web Application Firewall for SQL Injection," World Academy of Science, Engineering and Technology, vol. 40, pp. 42-51, April 2010.

8. Jin Li, Kangio Kim, "Hidden attributes-based Signatures without anonymity revocation" IEEE transactions on cloud computing vol 4, pp.no 490-499, 2011.

9. C.Wang, Q.Wang, K. Ren and W. Lou, "Privacy- preserving public auditing for data storage security in cloud computing" in Proc. IEEE Conf. Comput. Commun., 2010, pp.525-533.

10. C.Wang, Q.Wang and K. Ren "Ensuring data storage security in cloud computing" International Journal of Research in Advent Technology, Vol.2, No.2, Feburary 2014 E- ISSN:2321-9637.

11. 12. Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau, "Dynamic audit services for integrity verification of outsourced storages in clouds," in Proc. ACM Symp. Appl. Comput., 2011, pp.1550–1557.

12. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in Proc. 14th ACM Conf. Comput. Commun. Security, 2012, pp. 598–609.

13. 1. Aparna. K, Jyothy Solomon, Harini . M, Indhumathi . V, "A Study of Twofish Algorithm", 2016 pp.2321-9939.

14. Solomon Ogbomon, UwagboleWilliam, J. Buchanan, Lu Fan, "An Applied Pattern-Driven Corpus to Predictive Analytics in Mitigating SQL Injection Attack",2017.

15. Xiang Fu and K. Qian, "SAFELI – SQL Injection Scanner Using Symbolic Execution," in Workshop on Testing, Analysis and Verification of Web Software, July 21,2008.