A Novel Encryption approach in Database Security

Aarthi.G
Research Scholar
Mother Therasa Women University
Kodaikanal.

Dr. E. Ramaraj
Director , Computer Centre
Alagappa University,
Kaaraikudi.

Abstract:

Database servers are the most important thing in company environment. They store client details, financial information, human resource details and all the data are should be keptas very secret. Database security concerns the use of a broad range of information security controls to protect databases potentially including the data, the database applications or stored functions, the database systems, the database servers and the associated network links against compromises of their confidentiality, integrity and availability. It involves various types or categories of controls, such as technical, procedural/administrative and physical. This paper giving a novel security approch in data base encryption.

Keywords:

Database security, Encryption, Decryption.

I . Introduction:

Databases introduce a number of unique security requirements for their users and administrators. Databases are designed to promote open and flexible access to data at the same time it is this same open access that makes databases vulnerable to many kinds of malicious activity. This article is the first in a series that will look at a number of database-specific security concerns and provide one solution to overcome this issue.

One of the main issues faced by database security professionals is avoiding inference capabilities. Basically, inference occurs when users are able to piece together information at one security level to determine a fact that should be protected at a higher security level. These are the some security issues commonly faced by database administrators.

- Unauthorized or unintended activity or misuse by authorized database users, database administrators or systems managers, or by unauthorized users or hackers.
- Malware infections causing incidents such as unauthorized access, leakage or disclosure of personal or proprietary data, deletion of or damage to the data or programs, interruption or denial of authorized access to the database, attacks on other systems and the unanticipated failure of database services.
- Overloads, performance constraints and capacity issues resulting in the inability of authorized users to use databases as intended;
- Physical damage to database servers caused by computer room fires or floods, overheating, lightning, accidental liquid spills, static discharge, electronic equipment failures and obsolescence.
- Design flaws and programming bugs in databases and the associated programs and systems, creating various securities.
- Data corruption or loss caused by the entry of invalid data or commands, mistakes in database or system administration processes, criminal damage etc.

The vast majority of databases in use today have some form of web interface,

allowing internal or external users easy access through familiar browser software. If you are security conscious, you have undoubtedly spent a significant amount of time setting appropriate security permissions on your databases and web servers.

a) SQL Injection Attack:

One common type of database attack, the SQL Injection, allows a malicious individual to execute arbitrary SQL code on your server.

Let's take a look at how it works by analysing a very simple web application that processes customer orders. Suppose Acme Widgets has a simple page for existing customers where they simply enter their customer number to retrieve all of their current order information. The page itself might be a basic HTML form that contains a textbox called CustomerNumber and a submit button. When the form is submitted, the following SQL query is executed:

SELECT *FROM OrdersWHERE CustomerNumber = CustomerNumber

The results of this query are then displayed on the results page. During a normal customer inquiry, this form works quite well. Suppose John visits the page and enters his customer ID 14. The following query would retrieve his results:

SELECT *FROM OrdersWHERE CustomerNumber = 14

However, the same code can be a dangerous weapon in the hands of a malicious user. Imagine that Mal comes along and enters the following data in the CustomerNumber field: “14; DROP TABLE Orders”. This would cause the following query to execute:

SELECT * FROM OrdersWHERE CustomerNumber = 14; DROP TABLE Orders

There are several steps that you can take to protect your server against SQL Injection attacks:

Implement parameter checking on all applications. For example, if you’re asking someone to enter a customer number, make sure the input is numeric before executing the query. You may wish to go a step further and perform additional checks to ensure the customer number is the proper length, valid, etc.

Limit the permissions of the account that executes SQL queries. The rule of least privilege applies. If the account used to execute the query doesn’t have permission to drop tables, the table dropping will not succeed. Use stored procedures to prevent users from directly interacting with SQL code.

b) Access Controls in SQL:

Security is paramount to database administrators seeking to protect their gigabytes of vital business data from the prying eyes of unauthorized outsiders and insiders attempting to exceed their authority. All relational database management systems provide some sort of intrinsic security mechanisms designed to minimize these threats. This article focuses on the security mechanisms common to all databases that implement the Structured Query Language. Together, we will walk through the process of strengthening data access controls and ensuring the safety of your data. Server based databases all support a user concept similar to that used in computer operating systems.

It is highly recommended that you create individual database user accounts for each person who will be accessing your database. It's technically possible to share accounts between users or simply use one user account for each type of user that needs to access your database. If a specific user leaves your organization and you wish to remove his or her access from the database, you'll be forced to

change the password that all users rely upon.

The methods for creating user accounts vary from platform to platform and you'll have to consult your DBMS-specific documentation for the exact procedure. Microsoft SQL Server users should investigate the use of the sp_adduser stored procedure. Oracle database administrators will find the CREATE USER command useful. You also might want to investigate alternative authentication schemes. For example, Microsoft SQL Server supports the use of Windows NT Integrated Security. Under this scheme, users are identified to the database by their Windows NT user accounts and are not required to enter an additional user ID and password to access the database. This approach is extremely popular among database administrators because it shifts the burden of account management to the network administration staff and it provides the ease of a single sign-on to the end user.

If you're in an environment with a small number of users, you'll probably find that creating user accounts and assigning permissions directly to them is sufficient for your needs. However, if you have a large number of users, you'll most likely be overwhelmed by the burden of maintaining accounts and proper permissions. To ease this burden, relational databases support the notion of roles. Database roles function similarly to Windows NT groups. User accounts are assigned to role and permissions are then assigned to the role as a whole rather than the individual user accounts. For example, we could create a DBA role and then add the user accounts of our administrative staff to this role. Once we've done this, we can assign a specific permission to all present administrators by simply assigning the permission to the role. Once again, the procedure for creating roles varies from platform to platform. MS SQL Server administrators should investigate the sp_addrole stored procedure while Oracle DBAs should use the CREATE ROLE syntax.

C) Cryptography:

There are many aspects to security and many applications, ranging from secure commerce and payments to private communications and protecting passwords. One essential aspect for secure communications is that of cryptography. It is important to note that while cryptography is *necessary* for secure communications, it is not by itself sufficient. Within the context of any application-to-application communication, there are some specific security requirements, including:

- Authentication: The process of proving one's identity. (The primary forms of host-to-host authentication on the Internet today are name-based or address-based, both of which are notoriously weak.)
- Privacy/confidentiality: Ensuring that no one can read the message except the intended receiver.
- Integrity: Assuring the receiver that the received message has not been altered in any way from the original.
- Non-repudiation: A mechanism to prove that the sender really sent this message.

Cryptanalysis [6] is the study of methods for obtaining the meaning of encrypted information, without access to the secret information which is normally required to do so.

II. Proposed Method:

In this paper we are providing an innovative approach for database security using simple and basic encryption & decryption method. This cryptography approach enables the additional database security mechanism for improving the complexity of data. Important data should be secured from the data theft and other intrusion

methods. This encryption scheme completely based on table number (T), column value(C), row value(R). In this proposed scheme every data field get a unique key based on their position and place instead of common key value for all field.

Encryption Algorithm:

- Obtain the source text,
- Fetch their position values
- Use that as Encryption key value (E=T+R+C)
- Obtain the cipher text.
- Place the encrypted data in the corresponding field.
- Execute the query obtain cipher text values
- Apply the key values get the original values (plain Text).

This is a simple algorithm which provides an effective solution for data security process because encryption scheme should be very easy and simple for decryption process otherwise it increases the complexity of query retrieval. Running time of query is matter while coming to the encryption process. Compare the running time of normal execution with encrypted database execution will slightly huge and complexity will more while fetching the record.

A good encryption scheme should be computationally secure and it has to take more time to decrypt while third party accessing your information but while increasing the complexity of algorithm we have to take care of the running time and performance. After incorporating the encryption scheme that should be increase your system performance not to degrade. Encrypted data always slightly huge volume compare with original data due to that while designing the table field value size should be more compare with actual data field size what we need. While us choosing the encryption scheme that should not affect the structure of the database. Decryption key should not depend on another record or another field. Suppose data tried to modify some other person, alert information should show while decrypting the data by legitimate person. The following illustration explains how the key value allocation happening for every data filed. This is a sample example retrieved from the database discussed in appendix 1.

III. Conclusion:

Information security is very important in the trendy world. Every big concerns and institution contain their own database having varieties of information in different context. We proposed a simple method to enhance the database security. Even though this is simple approach it will yields strong security to protect the data's. We hope our research work will yields good impact on the database fields. In future we can enhance this method using some other encryption mechanism and also comparison analysis will make.

IV. References:

[1] H. Hacig¨um¨us, S. Mehrotra, and B. Iyer, "Providing database as a service," in International Conference on Data Engineering - ICDE 2002. IEEE Computer Society, 2002, pp. 29–39.

[2] L. Bouganim and Y. Guo, "Database encryption," in Encyclopedia of Cryptography and Security.Springer, 2010, 2nd Edition.

[3] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in Advances in Cryptology - CRYPTO 2007, ser. Lecture Notes in

Computer Science, vol. 4622. Springer, 2007, pp. 535–552.

[4] T. Ge and S. Zdonik, "Fast, secure encryption for indexing in a columnoriented DBMS," in International Conference on Data Engineering - ICDE 2007. IEEE, 2007, pp. 676–685.

[5] S. Goldwasser and S. Micali, "Probabilistic encryption," J. Comput.Syst. Sci., vol. 28, no. 2, pp. 270–299, 1984.

[6] Boneh D, Crescenzo GD, Ostrovsky R, Persiano G (2004) Public Key Encryption with Keyword Search. Encrypt 2004, LNCS 3027. pp. 506-522.

[7] Agrawal R, Kiernan J, Srikant R, Xu Y (2004) Order Preserving Encryption for Numeric Data. The ACM SIGMOD'2004, Paris, France.

[8] He J, Wang M (2001) Cryptography and Relational Database Management Systems, Proceedings of IEEE Symposium on the International Database Engineering & Applications, Washington, DC, USA.

[9] Chen G, Chen K, Dong J (2006) A Database Encryption Scheme for Enhanced Security and Easy Sharing. CSCWD'06, IEEE Proceedings, IEEE Computer Society, Los Alamitos. CA, pp. 1- 6.

[10] The Forrester Wave: Database Encryption Solutions, Q3 2005.

Appendix1:

Database Table No=13

Key(column1)	Key(column2)	Row (no)	Column1(data)	Column2(data)
E(13,22,1)	E(13,22,2)	22	55	234
E(13,23,1)	E(13,23,2)	23	45	245
E(13,24,1)	E(13,24,2)	24	123	34
E(13,25,1)	E(13,25,2)	25	456	3
E(13,26,1)	E(13,26,2)	26	567	245
E(13,27,1)	E(13,27,2)	27	456	789