Database Security using Intrusion Detection System

Yashashree Dawle,Manasi Naik,Sumedha Vande,Nikita Zarkar

Abstract— We propose a project named "Database Security Using IDS".SQL injection attack is the most common attack in websites now-a-days. SQL Injection refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server. In this project we propose database intrusion detection mechanism to enhance the security of database through a Website. We will make a system which will log all the activities of an Intruder using SQL Injection in a website. Some malicious codes gets injected to the database by unauthorized users and because of this attack, the actual database can be stolen or destroyed or modified by the hacker. Administrator will be shown the details of the user and can block him if needed. User details are secured using AES encryption algorithm which makes this system more secure.

Index Terms— SQL injection,AES symmetric key.

1 INTRODUCTION

Intrusion detection system detects the malicious activity in the database and notifies the administrator of the system accordingly. To secure data and detect malicious activities in database, intrusion detection system is integrated with the shopping site and detects malicious activities in site's database. Intrusion Detection System is a system in which malicious activity performed by any user or program is logged and can be viewed later by the admin. Anyone who gets access to the database login/password used by the application has the ability to frequently read or modify the database, bypassing all the security features built into the application. Therefore, security measures have been taken to ensure security at the application logic level, we need to have the ability to detect any malicious actions into the database.
As a part of the project, a website will be developed where users can buy shopping products. Different users- Administrator and customer will have different access rights to the system. This project is a base project for developing a Intrusion Detection System related subsystem which can be used in any other application.

1.1 Objectives

Building a system which can log of all the activities in a database. Currently this is integrated with a shopping website and will be able to track activities like selection of categories of product, adding product to the cart, buying products etc.
The main objective of this project is to secure the database using Intrusion Detection System.
Sql injection attack is very common attack to hack details of the user's transaction details present in the shopping site. Sql injection is a technique to insert sql queries in the database to steal the data from database. So as to secure database from this attack the parameterized queries developed by the user from this an attacker is not able to change the intent of a query, even if SQL commands are inserted by an attacker. In the safe example below, if an attacker were to enter the userID of tom' or '1'='1, the parameterized query would not be vulnerable and would instead look for a username which literally matched the entire string tom' or '1'='1. The objective of this project is to develop a secure path for transaction done by the user. Using AES (Advanced Encryption Standard) encryption technique, the transaction and user account details can be made secured.

2 Literature Review

Intrusion is any set of actions that attempt to compromise the confidentiality, availability or integrity of a resource. Intrusion Detection Systems determines the illegal actions performed against computer systems and accordingly alerts the system administrator.
Log based approach: Database Malicious Transaction Detector (DBMTD) mechanism is a log based mechanism for the detection of malicious transactions in DBMS. In practice malicious database transactions are related to security attacks carried out either externally or internally to the organization. External security attacks are intentional unauthorized attempts to access or destroy the organization's private data. On the other hand, internal security attacks are intentional malicious actions executed by authorized users. In a typical database environment the profile of the transactions that each user is allowed to execute is usually known by the DBA, as the database transactions are programmed in the database application code.
The audit log is used by DBMTD to obtain the sequence of commands executed by each user, which is then compared with the profile of the authorized transactions to identify potential malicious transactions. The data entered into the database is encrypted with the help of AES algorithm, which avoids any intruder attacks such as snooping. The sequence of commands, done by the user can be viewed in our database only by our administrator. *Fig. 5.1* DB system using the DBMTD mechanism But there are some drawbacks of this approach, which are listed .Logs, are difficult to manage especially when systems of different operating systems are accessing. Also, if at all the logs are stored in the secondary storage the time is taken more to fetch and compare with the transaction profile.

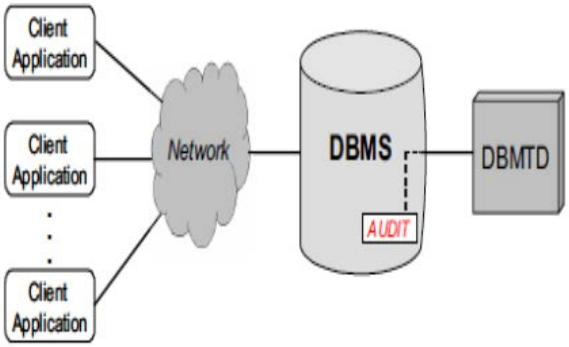


Fig. DB System using DBMTD mechanism

3 Problem Statement

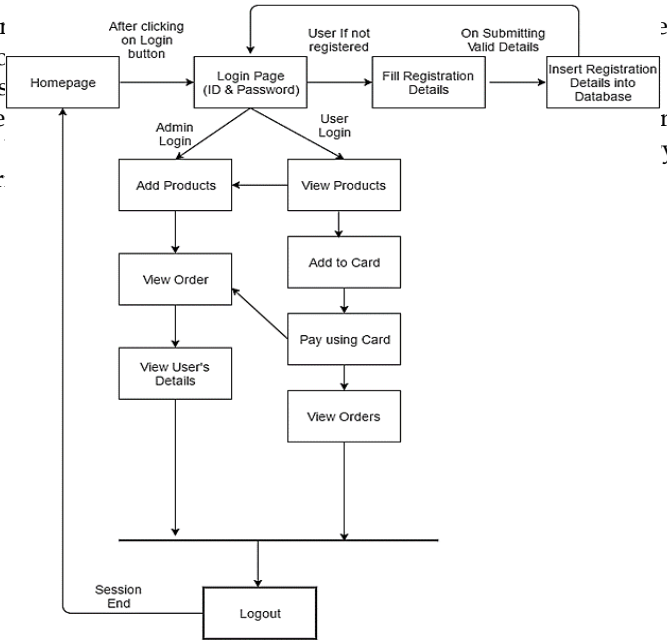
A website needs to be developed that will allow users to buy products. A registered user can buy the product by providing his/her login details after which payment details are submitted. The system should be secure and should not allow a user to perform tasks that he is not authorized to do. All the activities on the website should be logged into an Intrusion Detection System to identify if any malicious activity is being performed on the system. A user with admin rights can view these logs.
In this system, logs consisting transaction sequence helps Intrusion detection , to detect the malicious transactions. Transaction profiling corresponds to the identification of the sequence of commands that constitute each valid transaction.
The administrator can view the database at any time. He can send the data to analyze the database for automatic analysis that will give results in the form of all unauthorized requests sent by a user. The intrusion detection system detects the malicious tasks happens in the system and report of that malicious activities to the administrator and administrator takes the appropriate action.
Basically, this system perform following steps, All the transactions (activities) performed on the website should be logged into database system to identify if any malicious activity is performed on the system. This system will be able to track the transactions performed by the users – authorized and unauthorized. The administrator and higher authorities only can view these logs and perform profiling wherein they identify different transactions that are possible in the system.

4 Existing System

In existing system, the software based site is used to buy products, the customer checks the products and then add to the cart and proceed to the payment/transaction.
The customer selects the product according to their requirements and add to the cart, the quantity of the product is choose by the customer and total amount is displayed in transaction page.
The customer can make payment using a credit card. The customer fills all transaction details required to perform the transaction but it will not be secure. Any attacker can inject a sql command in database and can see the details of the customer. The existing system do not support the intrusion detection system that did not gave the information about any malicious activity done in database to the administrator.

4.1 Limitations

In existing system, customer does r e card details are stored as it is into the database without any encryptic
There is a possibility of card details
The attacker can anytime check the n.
Because of this attack the software y details and coding, programs are seen by the attacker and can perfor



5 Proposed System

Considering the major anomalies into the existing system, we conclude to build this proposed system.
The highlighted part here is encryption of card data using AES (Advanced Encryption Standard) technique.
The Online Shop secures the card payment and won't let the card data to get hacked.
While user doing a card payment, all the card data is encrypted and then stored into database.
System also keeps user details in an encryption form using AES encryption.
The system is built of handling SQL Injection capabilities which doubles the security of database and prevents from injection hacking codes into the database.
In the proposed website there are different parts or modules which are summarized as follows:

CUSTOMER REGISTRATION:

Customers are required to register on the website before they can do the shopping. The website also provides several features for the non-registered user. Here they can choose their id and all the details regarding them are collected and a mail is sent to the email address for confirmation.

SHOPPING CART:

Shopping cart module tries to simulate the working of a store where user can view each Product type, size and price of the product available. The items they like can be added to the logical cart and can be removed if not required later. Billing and other payment related matters are handled here.

ADMINISTRATION:

This is the part of the website where the administrators can add delete or update the product information. Administrators are also responsible for adding and deleting the customers from the website. In addition, newsletter and promotions are also handled by the site administrator via e-mail.

SEARCH:

This facility is provided to both registered and unregistered user. User can search for the availability and type of products available on the website.

ADD TO CART:

Users can add Product to cart.

CREDIT CARD PAYMENT:

After total bill is calculated user can pay via credit card online with homomorphic encryption technique.

6 Implementation Details

The whole implementation is done using windows operating system. The design and coding is done using PHP, WAMP Server & Notepad++. Database is created and maintained using MySQL.

6.1 AES Algorithm

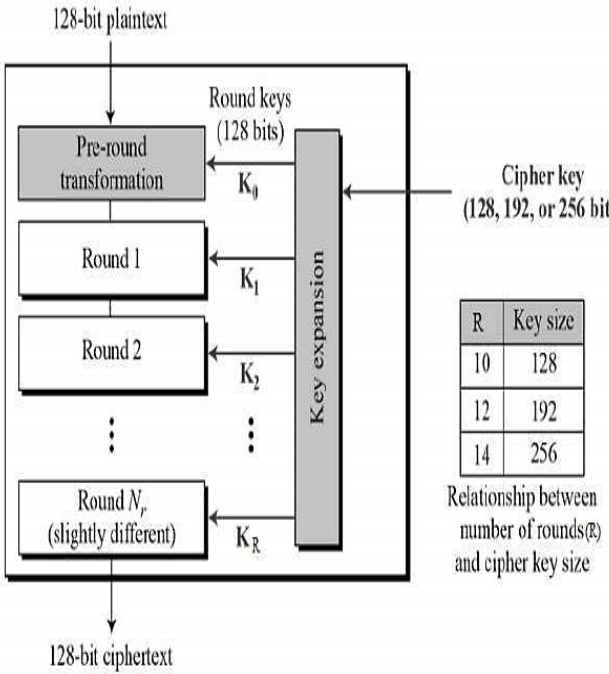
AES is based on a design principle known as a substitution-permutation network, a combination of both substitution and permutation, and is fast in both software and hardware. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits.It is found at least six time faster than triple DES.

The features of AES are as follows −

Symmetric key symmetric block cipher
128-bit data, 128/192/256-bit keys
Stronger and faster than Triple-DES
Provide full specification and desig
Software implementable in C and Ja

Operation of AES

AES is an iterative rather than Feist ome of which involve replacing inputs by specific outputs (substitutio ns).
Interestingly, AES performs all its c s the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arrange ix −
Unlike DES, the number of rounds i AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rc ent 128-bit round key, which is calculated from the original AES key.
The schematic of AES structure is giv



R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

Encryption
Here, we restrict to description of a typical round of AES encryption. Each round comprise of four sub-processes. The first round process is depicted below −

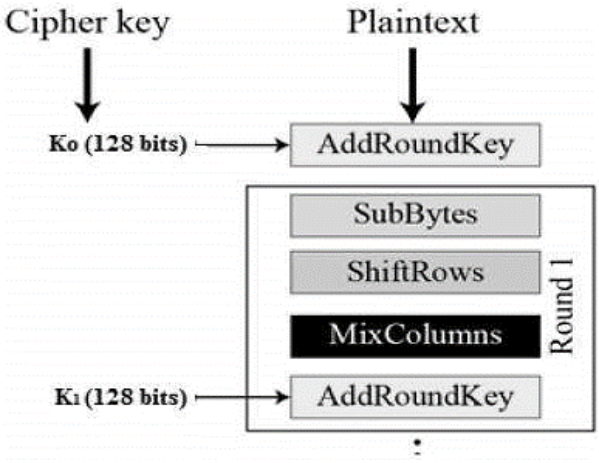


Fig6.2 Encryption first round process

Byte Substitution (Sub Bytes)
The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.
Shift rows
Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of row. Shift is carried out as follows −
First row is not shifted.
Second row is shifted one (byte) position to the left.
Third row is shifted two positions to the left.
Fourth row is shifted three positions to the left.
The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.
Mix Columns
Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.
Add round key
The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round, then the output is the cipher text. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.
Decryption
The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order −
Add round key
Mix columns
Shift rows
Byte substitution
Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms needs to be separately implemented, although they are very closely related.

6.2 SQL Injection

SQL Injection Attacks have emerged as one of the most serious threats to the security of database-driven applications. They are very common due to two factors:
the significant prevalence of SQL Injection vulnerabilities, and
the attractiveness of the target (i.e., the database typically contains all the interesting/critical data for your application).
SQL injection attacks are initiated by manipulating the data input on a Web form such that fragments of SQL instructions are passed to the Web application. The Web application then combines these rogue SQL fragments with the proper SQL dynamically generated by the application, thus creating valid SQL requests. These new, unanticipated requests cause the database to per-

form the task the attacker intends. To clarify, consider the following simple example. Assume we have an application whose Web page contains a simple form with input fields for username and password. With these credentials the user can get a list of all credit card accounts they hold with a bank. Further assume that the bank's application was built with no consideration of SQL injection attacks. As such, it is reasonable to assume the application merely takes the input the user types and places it directly into an SQL query constructed to retrieve that user's information. In PHP that query string would look something like this:

$query = "select accountName, accountNumber from creditCardAccounts where username='".$_POST["username"]."' and password='".$_POST["password"]." ' "

Normally this would work properly as a user entered their credentials, say johnSmith and myPassword, and formed the query:

$query = "select accountName, accountNumber from creditCardAccounts where username='johnSmith' and password='myPassword'

This query would return one or more accounts linked to Mr. Smith. Now consider someone with a devious intent. This person decides they want to see if they can access the account information of one or more of the bank's customers. To accomplish this they enter the following credential into the form:

' or 1=1 -- and anyThingsAtAll

When this SQL fragment is inserted into the SQL query by the application it becomes:

$query = "select accountName, accountNumber from creditCardAccounts where username='' or 1=1 -- and password = anyThingsAtAll

The injection of the term, ' or 1=1 --, accomplishes two things. First, it causes the first term in the SQL statement to be true for all rows of the query; second, the -- causes the rest of the statement to be treated as a comment and, therefore, ignored during run time. The result is that all the credit cards in the database, up to the limit the Web page will list, are returned and the attacker has stolen the valuable information they were seeking.

7 Conclusion

In the project "Database Security using IDS" we have proposed AES encryption approach to prevent the intrusion in database of the online shopping website. Also detects and prevents the intrusion attacks like SQL injection. It provides an additional layer of security in database management system (DBMS).It can be considered as generic approach for any database and overcomes the limitations of the existing database security mechanisms.

Acknowledgment

We are pleased to present **"Database Security using IDS"** project and take this opportunity to express our profound gratitude to all those people who helped us in completion of this project. We express our deepest gratitude towards our project guide for her valuable and timely advice during the various phases in our project.

References

[1] Udai Pratap Rao, Dhiren R. Patel, "Database Security Architecture for Detection of Malicious Transactions in Database".
[2] Marco Vieira, Henrique Madeira, "Detection of Malicious Transactions in DBMS".
[3] Christina Yip Chung, Michael Gertz, Karl Levitt, "DEMIDS: A Misuse Detection System for Database Systems".
[4] Flavio Bonomi1, Michael Mitzenmacher2, Rina Panigrahy3, Sushil Singh1, and George Varghese1, "An Improved Construction for Counting Bloom Filters".
[5] http://www.google.co.in
[6] Sonam Panda, Ramani S, "Protection of web application against Sql Injection attacks" Vol.3, Issue.1, Jan-Feb. 2013 pp-166-168
[7] Dr. Amit Chaturvedi, Aijaz Ahemad Rather, "Analysis of SQL injection attacks and prevention methods in web applications" Volume 5, Issue 12, December 2015