INTERNATIONAL CONFERENCE ICGTES-19, ICITBDIT-19
Conference World
ISBN : 978-81-941721-9-2
OUCIP, Osmania University Campus, Hyderabad (India)
05th October 2019 www.conferenceworld.in

User Centric Security Models for Improving the Data Security from SQL Injections and Cross Site Scripting Attacks

[1]Vamsi Mohan V, [2]Dr. Sandeep Malik

[12]Department of Computer Science,
[12]School of Engineering and Technology, Raffles University, Neemrana, India

ABSTRACT -SQL Injection Attacks (SQLIA) and Cross-site Scripting Attacks (XSSA) are frequently used by hackers to hack the user data. SQL Injection (SQLI) executes malicious code in the form of SQL statement at database by directly executing the query or changing the requested parameters, changing the triggered URL. In case of Cross-Site Scripting, the attacker targets to execute the malicious code at the client side.

Index Terms—SQL Injections (SQLI), Cross Site Scripting (XSS), Static Analysis, Identity Based Encryption (IEB), Advanced Encryption Standard (AES), Regression Neural Network (RNN), Dynamic Analysis.

I. INTRODUCTION

Web usage and its application has reached to a wide extend in past decade because of the ease of accessibility and information availability. With this wide spread usage of web information's, various security lapses have also emerged which remains to be a major drawback. For any web user, the information's that is being employed should be secured to avoid any unauthorized usage. Majorly these web security issues make the hackers extract complete user information. This can cause diverse effect on user credentials. SQL injection and cross site scripting (XSS) has emerged as the recent vulnerability in web world. Since the emergence of cloud computing and e banking applications, the above vulnerability has become an important issue in various web application as they can easily penetrate over them. Hence the requirement of developing novel approaches to overcome these vulnerabilities has increased. SQL injection and XSS are basically input validation-based vulnerabilities and they majorly involve information from untrusted and unknown sources, and presenting those values to in the form of programs or program fragments. The major issue in validating these vulnerabilities is that no techniques developed has the tendency to characterize and mitigate these attacks. In this paper, we explained how to transfer data through secure methods avoiding SQL Injection attacks and Cross Site Scripting attacks. We used Hybrid Optimization Techniques for enhancing the detection accuracy through regression Neural networks for SQLIA and XSS attacks.

II. SQL INJECTION ATTACKS (SQLIA) AND CROSS SITE SCRIPTING ATTACKS (XSS)

SQL Injection (SQLI) is an injection attack that makes it possible to execute malicious SQL statements by adding or modifying the existing query through URL or from the backend database. These statements control the database server behind web applications. Attacker uses SQL Injection vulnerabilities to bypass application security measures. They cheat the authentication and authorization of a web page or web application and retrieve the content of the database. Normally, attackers perform add, edit or delete operations on the database either to grab the data or destroy the existing data.

Hackers use SQL injections to gain unauthorized access to the sensitive data such as customer information, personal data, trade secrets, intellectual property, and more. According to OWASP, SQL Injection attacks are one of the oldest, prevalent, and dangerous web application vulnerabilities.

In our experiment, we have used the technique to "Retrieve the hidden data". We tried to add an additional query to the original query triggered by the application.

Original Query:http://<hostname:port>/Sql_Injection/Shared-Datas-AES.jsp?id=8
Query Action: The above queried to retrieve the user data, where id = 8.

Morphed Query: http://<hostname:port>/Sql_Injection/Shared-Datas-AES.jsp?id=8 OR id >=1

Query Action: The above queried to retrieve the user data, where id = 8 and concatenated an additional query with OR id >=1, which is always true. Hence, the system tries to retrieve the query with all the id >= 1 (entire table).

Cross-site Scripting (XSS) is a client-side or browser side code injection attack. The attacker tries to execute malicious code by changing the actual URL in a web browser of the victim in a legitimate web page or web application. The actual attack occurs or results when the user visits the web page that executes the malicious code or in another case, by changing the application package. The web page or web application delivers the malicious script to the user's browser.

In our experiment, we tried to block the access of the application through in-cognitive mode. With this, hackers fail to access the application without proper authentication. Hence, hackers or user has to depend on the cognitive or regular mode of accessing the application. If someone is accessing through cognitive mode by changing the URL, it will be recorded and the administrator can get the details about the user, including the ip address of the user along with the date and time of application access.

By disabling the application access from the in-cognitive or private mode, the users or hackers cannot access hiding their authenticity.

III. IMPACT OF SQLIA AND XSS

SQL Injection Attacks (SALIA)

To make a SQL Injection attacks, an assailant should initially discover vulnerable user inputs within the web site or application. A site page or web application that has a SQL Injection weakness uses such client input straight forwardly in a SQL query. The attacker can create input content. Such content is called a pernicious or malicious payload and is the key to perform attack. After the attacker sends this content, SQL commands are executed on the table in the database.

SQL is a query language that was intended to oversee information stored in databases. You can utilize it to view, edit, and delete information. Many web applications and websites store every information in SQL databases. At times, users can user SQL commands to run OS commands. Hence, SQL Injection attacks can have serious consequences.

- Attackers can utilize SQL Injections to find user details in the database. They would then be able to mimic these users. The imitated user might be a database administrator, who is having admin privileges to the database.
- SQL allows to get information from the database. A SQL Injection vulnerability could enable the attacker to gain access to all information in a database server.
- SQL likewise gives user to alter the data or add new records to the database table. For instance, in a budgetary application, an attacker could utilize SQL Injection to change, void transactions, or transfer cash to their own accounts.
- You can utilize SQL to erase records from a database, even drop tables.
- In some database servers, you can get access to operating system through the database server. This might be purposeful or unintentional.
- In such case, an attacker could utilize a SQL Injection as the underlying vector and then attack the internal network behind a firewall.

Crosssite Scripting (XSS)

XSS vulnerabilities are seen as less risky than SQL Injection vulnerabilities. Outcomes ofthe ability to execute JavaScript on a site page may not seem dire at first. Most internet browsers run JavaScript in a firmly controlled condition. JavaScript has constrained access to the client's working framework and the user files. In any case, JavaScript can even now be hazardous whenever abused as a major aspect of malevolent substance:

- Pernicious JavaScript has access to all the objects that the rest of the web page has access. This incorporates access to sensitive user information through cookies. Cookies are frequently used to store

session tokens. If the attacker obtains the cookies of the user's browser, they can impersonate the user without proper authentication.

- JavaScript can peruse the browser DOM and make arbitrary changes to it. Fortunately, this is just conceivable inside the page where JavaScript is running.
- JavaScript can utilize the `XMLHttpRequest` article to send `HTTP` requests with arbitrary content to arbitrary destinations.
- JavaScript in modern browsers can utilize HTML5 APIs. For instance, it can access the client's geolocation, webcam, receiver, and even explicit records from the user's file system. The majority of these APIs require user optin;however, the assailant can utilize social engineering to circumvent that impediment.

IV. PROPOSED MODEL FOR SECURE DATE TRANSFER

When data is being sent from the sender to the receiver, there is every possibility hacker or eavesdroppers will hack the data or grab the data and use it for their own purpose. In normal mode of communication, the sender sends the data in plain text file, where it can be read by anyone. In our proposal model, the send can send the data by splitting into parts and encrypt using IBE and AES encryption techniques. This helps the sending to transfer the data securely. By splitting the data into parts, it is difficult to the hackers to hack the complete information as it is split into different sub-parts.

Steps followed in the proposed model for secure data transfer:

Step 1:

Data packets from different users are fed to the server. We employed SQL injection and cross site scripting vulnerabilities detection to conclude the key motive following the mortal impact of these attacks on web applications

Step2:

The transmission will be subjected to SQL injection and cross site scripting vulnerabilities its effects are analysed.

Step 3:

The data will be subjected to advanced encryption algorithm by combination of IBE and AES techniques. Once the data are secured through encryption, this will be fed to the server based on the web application

Step 4:

For detection process, we employ advanced multi-objective classifiers to detect the attacks, thereby neglecting its effect on web applications

Step 5:

In our proposed model Regression neural network are being incorporated with hybrid optimization techniques for enhancing the detection accuracy.

Step 6:

Finally, the evaluation parameters like detection accuracy for the technique are estimated and compared with existing techniques.

V. IBE AND AES TECHNIQUES

Identity-Based Encryption (IBE)

IBE, stands for Identity Based Encryption a technique where public key is an arbitrary string (ID). Example: user's email-ids, current date, etc., It is standardized by IEEE 1363.3.

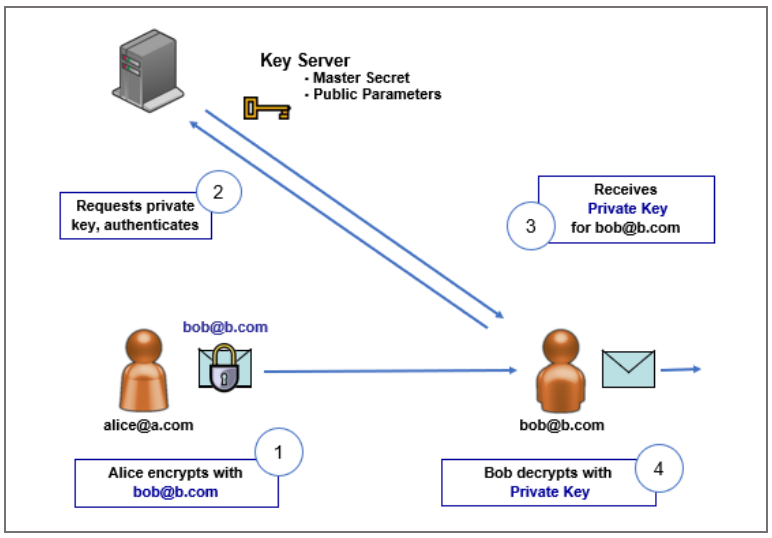

Fig. 1.0

In the above example, Alice (sender) tries to send the mail (from alice@a.com) through encrypting public address of receiver (bob@b.com) and sends to bob@b.com address. When Bob receives the mail, he requests private key from the Keygen server. Keygen Server sends the private key to Bob and he decrypts the received mail using the private key.

IBE is a policy driven encryption. It is controlled by the administrators. It automatically enforces based on the message flow or the content. Users can opt-in or opt-out based on the keywords (automatically through policy guidelines) and no client software required.

Examples:

```
Encrypt all traffic to xyz.com
Encrypt from john@co.com
Encrypt all ePHI (lexicon)
Encrypt if subject contains "confidential"
```

Advanced Encryption Standard (AES)

We have used Advanced Encryption Standard (AES), which is at least six time faster than triple DES. The number of rounds in AES is depends on the key length. AES use 128-bit, 192 bit and 256 bits using 10, 12, and 14 rounds.Each of these rounds use different 128-bit round key calculated from the AES original key.

The features of AES are given below:

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys with 10/12/14 rounds
- Stronger and faster than Triple-DES algorithm
- Provide full specification and design details

AES is an iterative approach. It is based on 'substitution–permutation network' method. It includes of a sequence of linked operations, which includes changing input by using particular output or substitutions and others contain shuffling bits round as permutations.

AES performs all its computations mainly on bytes alternatively than bits. Hence, AES treats the 128 bit of a plain textual content block as sixteen bytes. These 16 bytes are arranged in 4 columns and four rows for processing as a matrix.

The schematic of AES structure is shown in the below illustration:

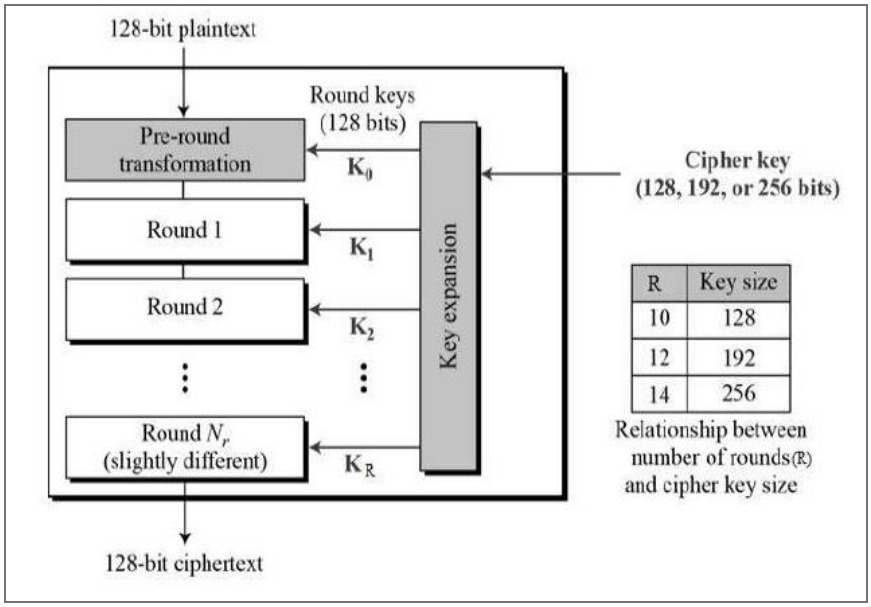

Fig. 2.0

Encryption Process

We restrict to description of a typical round of AES encryption. Each round comprises of four sub-processes.

The first-round process is depicted below:

ISBN : 978-81-941721-9-2

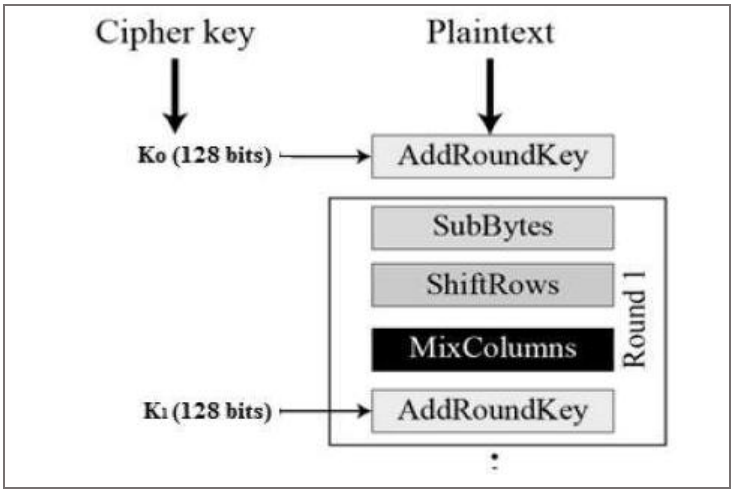

Fig 3.0

Byte Substitution (SubBytes)

The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The outcome is in a matrix of four rows and four columns.

Shiftrows

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.

MixColumns

Each column of four bytes is now transformed the use of a specific mathematical function. This characteristic takes as enter the 4 bytes of one column and outputs four totally new bytes, which exchange the original column. The end result is every other new matrix consisting of 16 new bytes. It has to be noted that this step is now not performed in the remaining round.

Addroundkey

The sixteen bytes of the matrix are now viewed as 128 bits and are XORed to the 128 bits of the spherical key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we start any other comparable round.

Decryption Process

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 need to be separately implemented, although they are very closely related.

VI. ADVANTAGES OF PROPOSED MODEL

1. Data transport is quite fast as it is split into different parts and secure from SQL injection and Cross Site scripting attacks.
2. Protects from the unauthorized users by using in cognitive mode, using IBE and AES algorithms.
3. History of the data access users with date and time logs.

VII. CONCLUSION AND FUTURE WORK

We tried to propose the model to transport data securely from the SQL injections and Cross site scripting attacks. However, the scope of the experimentation is limited to the data transfer than the web or mobile user transactions. The proposed algorithms can be further optimized based on the requirement. The proposed model is out of scope for Create, Retrieve, Update and Delete (CRUD) transactions.

REFERENCES

[1] Atefeh Tajpour ,Suhaimi Ibrahim, & Mohammad Sharifi (2012), Web Application Security by SQL Injection DetectionTools. IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 2, No 3, March 2012, ISSN (Online): 1694-0814.

[2] G. Wassermann and Z. Su, “Static detection of cross-site Scripting vulnerabilities,” In Proceeding of the 30th International Conference on Software Engineering, (2008)May.

[3] I. Hydara, A. B. M. Sultan, H. Zulzalil, N. Admodisastro, Current state of research on cross-site scripting (xss) c a systematic literature review, Information & Software Technology 58 (2014) 170–186.

[4] I. Parameshwaran, E. Budianto, S. Shinde, H. Dang, A. Sadhu, P. Saxena, Dexterjs: robust testing platform for dom-based xss vulnerabilities, in: Joint Meeting, 2015, pp. 946–949.

[5] I. Parameshwaran, E. Budianto, S. Shinde, H. Dang, A. Sadhu, P. Saxena, Auto-patching dom-based xss at scale, in: Joint Meeting, 2015, pp. 272–283.

[6] K. Z. Snow, R. Rogowski, J.Werner, H. Koo, F. Monrose, M. Polychronakis, Return to the zombie gadgets: Undermining destructive code reads via code inference attacks, in: Security and Privacy (SP), 2016 IEEE Symposium on, IEEE, 2016, pp. 954–968.

[7] Makera M Aziz and Dena Rafaa Ahmed (2015), Proposed Method to Prevent SQL Injection Attack. The Annual Conference on Network Security & Distributed Systems (NSDS'2015).

[8] Muhammad SaiduAliero, Abdulhamid Aliyu Ardo, Imran Ghani & Mustapha Atiku (2016), Classification of Sql Injection Detection And Prevention Measure. IOSR Journal of Engineering (IOSRJEN), ISSN (e): 2250-3021, ISSN (p): 2278-8719, Vol. 06, Issue 02 (February. 2016), ||V1|| PP 06-17.

[9] P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Krgel, G. Vigna, Cross site scripting prevention with dynamic data tainting and static analysis., in: Network and Distributed System Security Symposium, NDSS 2007, San Diego, California, Usa, February - March, 2007.

[10] P. Saxena, S. Hanna, P. Poosankam, D. Song, Flax: Systematic discovery of client-side validation vulnerabilities in rich web applications., in: Network and Distributed System Security Symposium, NDSS 2010, San Diego, California, Usa, February - March, 2010.