ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 3, pp: (349-356), Month: July - September 2014, Available at: www.researchpublish.com

OBFUSCATED PROTECTION WITH CHAFFING AND WINNOWING AGAINST SQL INJECTION

[1] Ms. A. Sivasankari, [2] Sangeetha.V

[1, 2]Department of Computer Science, D.K.M College for Women, Vellore, Tamil Nadu, India

Abstract: **User application has been developed with very rapid progress. Security vulnerabilities due to amendment of escape characters and comment characters with the actual keywords becomes predominant in the current IT environments. There are some malicious attacks which can deceive this SQL. To stop string literal escape characters injections, many techniques have been proposed but they require large code modification and/or large extra time overhead.**

The work of this paper proposes a technique SQL Injection Protector for Authentication (SQLIPA) with the help of removal of SQL Query attribute Algorithm. In addition to the proposed technique we are trying to store the user credentials in the images and retrieve the same for validation. These images will be stored in BLOB format with Encrypted layer in the backend to hide the tuples used for storage. Validating the XML content with typed dataset will scrutinize the input data further.

Keywords: **Chaffing and Winnowing, SQL Injection, Obfuscation, Hash defined algorithm.**

1. INTRODUCTION

Intrusion detection is one of the most challenging area in the IT industry. Most of the time, the system is hacked through some of the techniques. Once if we have fix up the hole, there will be another hole for the hacker to enter into the system. Most of the attacks made on the web target the vulnerability of web applications.

SQL-Injection Attack is not as damaging to systems using and operating web applications as other attacks, but because of its ability to obtain and charge the sensitive information, such as military systems, banks, and e-business, etc are exposed to a great security risk.

Many divisions are researching a variety of methods to detect and prevent SQLIAs, and the most preferred techniques are Web Framework, Static Analysis, Dynamic Analysis, Combined Static and Dynamic Analysis, and Machine Learning Techniques. Providing filtering methods using the user's input data. However, it is only able to filter special characters therefore, other attacks cannot be prevented.

This paper proposes a simple and effective method to accurately detect SQLIAs by using a combination of SQL query parameter removal and Combined Static and Dynamic Analysis methods. Furthermore, the effectiveness of this method in web applications has been tested and validated.

In order to understand the SQLIAs, there will be a brief explanation on the architecture and processes of applications. Also, there will be a discussion on possible SQL-Injection vulnerabilities and attacks made on these applications. SQL-Injection vulnerabilities and attacks occur between the Presentation tier and the CGI tier. Most vulnerability are

accidentally made in the development stage. The data flow of each tier using normal and malicious Input data. It depicts the user's authentication step.

When an authenticated user enters its ID and Password, the Presentation tier uses the GET and POST method to send the data to the CGI tier. The SQL query within the CGI tier connects to the database and processes the data. This paper proposes a new SQL injection attack detection method that utilizes both Static and Dynamic Analysis.

This method compares and analyzes by removing the attribute value of the SQL queries. The detection method is elaborated based on the contents and last of all the explanation of the proposed algorithm is discussed. This paper proposes a SQL Injection detection method by comparing the static SQL queries with the dynamically generated queries after removing the attribute values.

Furthermore, we evaluated the performance of the proposed method by experimenting on a vulnerable application. We also compared our method with other detection methods and confirmed the efficiency of our proposed method. The proposed method simply removes the attribute value in the SQL queries for analysis which makes it independent from the DBMS.

The techniques implemented in this system are too simple and strong. We are trying to address some of the huge issues like, Tautological way of attack can be fixed up with White Box Static analysis approach Spam bot attack which can be fixed up with Anti spoofing approach. We are designing a system which can overcome hacking at all the levels of the system. The system can be hacked at the presentation layer level which is fixed. The system can be hacked at the business logic level which is fixed. The system can be hacked at the database level which is also fixed.

This is the topmost level of the application. The presentation tier displays information related to such services as browsing merchandise, purchasing and shopping cart contents. It communicates with other tiers by which it puts out the results to the browser/client tier and all other tiers in the network.

In simple terms it is a layer which users can access directly such as a web page, or an operating systems GUI. Application tier business logic, logic tier, or middle tier. The logical tier is pulled out from the presentation tier and, as its own layer, it controls an application's functionality by performing detailed processing.

II. CHAFFING AND WINNOWING

Chaffing and winnowing is a cryptographic technique to achieve confidentiality without using encryption when sending data over an insecure channel. The name is derived from agriculture after grain has been harvested and threshed it remains mixed together with inedible fibrous chaff. The chaff and grain are then separated by winnowing, and the chaff is discarded. The technique was conceived by Ron Rivets. Although it bears similarities to both traditional encryption and steganography, it cannot be classified under either category.

This technique allows the sender to deny responsibility for encrypting their message. When using chaffing and winnowing, the sender transmits the message unencrypted, in clear text. Although the sender and the receiver share a secret key, they use it only for authentication. However, a third party can make their communication confidential by simultaneously sending specially crafted messages through the same channel.

The sender wants to send a message to the receiver. In the simplest setup, Sender enumerates the symbols usually bits in her message and sends out each in a separate packet. In general the method requires each symbol to arrive in-order and to be authenticated by the receiver. When implemented over networks that may change the order of packets, the sender places the symbol's serial number in the packet, the symbol itself both unencrypted, and a message authentication code (MAC).

Many MACs use a secret key Sender shares with Receiver, but it is sufficient that the receiver has a method to authenticate the packets. The third party, who transmits Sender's packets to Receiver, interleaves the packets with corresponding bogus packets (called "chaff") with corresponding serial numbers, arbitrary symbols, and a random number in place of the MAC.

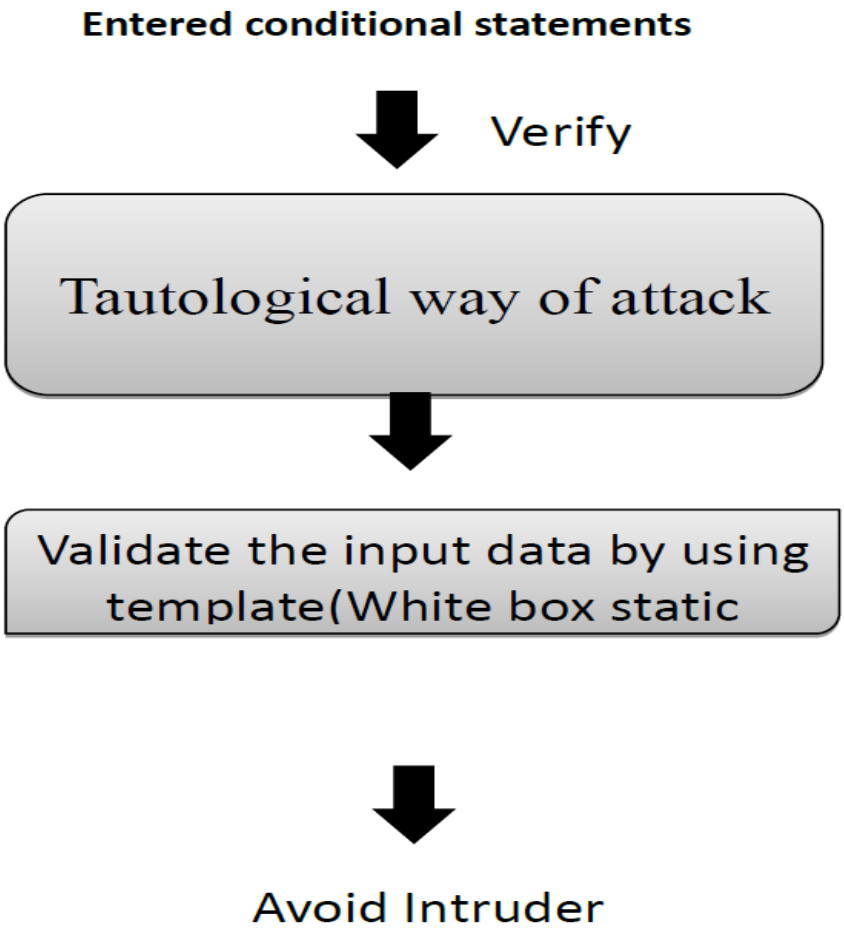
Third party does not need to know the key to do that (real MAC are large enough that it is extremely unlikely to generate a valid one by chance, unlike in the example). Receiver uses the MAC to find the authentic messages and drops the "chaff" messages. This process is called "winnowing".

Chaffing and winnowing lends itself especially well to use in packet-switched network environments such as the Internet, where each message whose payload is typically small is sent in a separate network packet. In another variant of the technique, Third party carefully interleaves packets coming from multiple senders. That eliminates the need for Third party to generate and inject bogus packets in the communication. However, the text of Alice's message cannot be well protected from other parties who are communicating via Third party at the same time. This variant also helps protect against information leakage and traffic analysis.

1. SQL injection

SQL injection is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution to dump the database contents to the attacker. SQL injection must exploit a security vulnerability in an application's software, when user input is either incorrectly filtered for string literalescape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

This form of SQL injection occurs when user input is not filtered for escape characters and is then passed into a SQL statement. This results in the potential manipulation of the statements performed on the database by the end-user of the application.



We can prevent the system from all type of SQL injections based upon the classifications. We can block the hackers to enter into the system by comparing with the pre-defined template. If the hacker enters into the first level of authentication means we can block them by using XSD validation and Hash key generation methods. Here we use the anti-spoofing method and also the winnowing and chaffing method to give the prevention from the hacking.

Code injection is the exploitation of a computer bug that is caused by processing invalid data. Code injection can be used by an attacker to introduce or "inject" code into a computer program to change the course of execution. The results of a code injection attack can be disastrous. For instance, code injection is used by some computer worms to propagate.

Code injection techniques are popular in system hacking or cracking to gain information, privilege escalation or unauthorized access to a system. Code injection can be used malevolently for many purposes, including:

- Arbitrarily modify values in a database through a type of code injection called SQL injection. The impact of this can range from website defacement to serious compromise of sensitive data.
- Install malware or execute malevolent code on a server, by injecting server scripting code such as PHP or ASP.
- Privilege escalation to root permissions by exploiting Shell Injection vulnerabilities in a setuid root binary on UNIX, or Local System by exploiting a service on Windows.

Attacking web users with HTML/Script Injection

2. CODE injection

Code injection is the exploitation of a computer bug that is caused by processing invalid data. Code injection can be used by an attacker to introduce or "inject" code into a computer program to change the course of execution. The results of a code injection attack can be disastrous. For instance, code injection is used by some computer worms to propagate.

Injection flaws occur when an application sends untrusted data to an interpreter. Injection flaws are very prevalent, particularly in legacy code. They are often found in SQL, LDAP, Xpath, or No SQL queries; OS commands; XML parsers, SMTP Headers, program arguments, etc. Injection flaws are easy to discover when examining code, but frequently hard to discover via testing. Scanners and fuzzers can help attackers find injection flaws.

Code injection techniques are popular in system hacking or cracking to gain information, privilege escalation or unauthorized access to a system. Code injection can be used malevolently for many purposes, including:

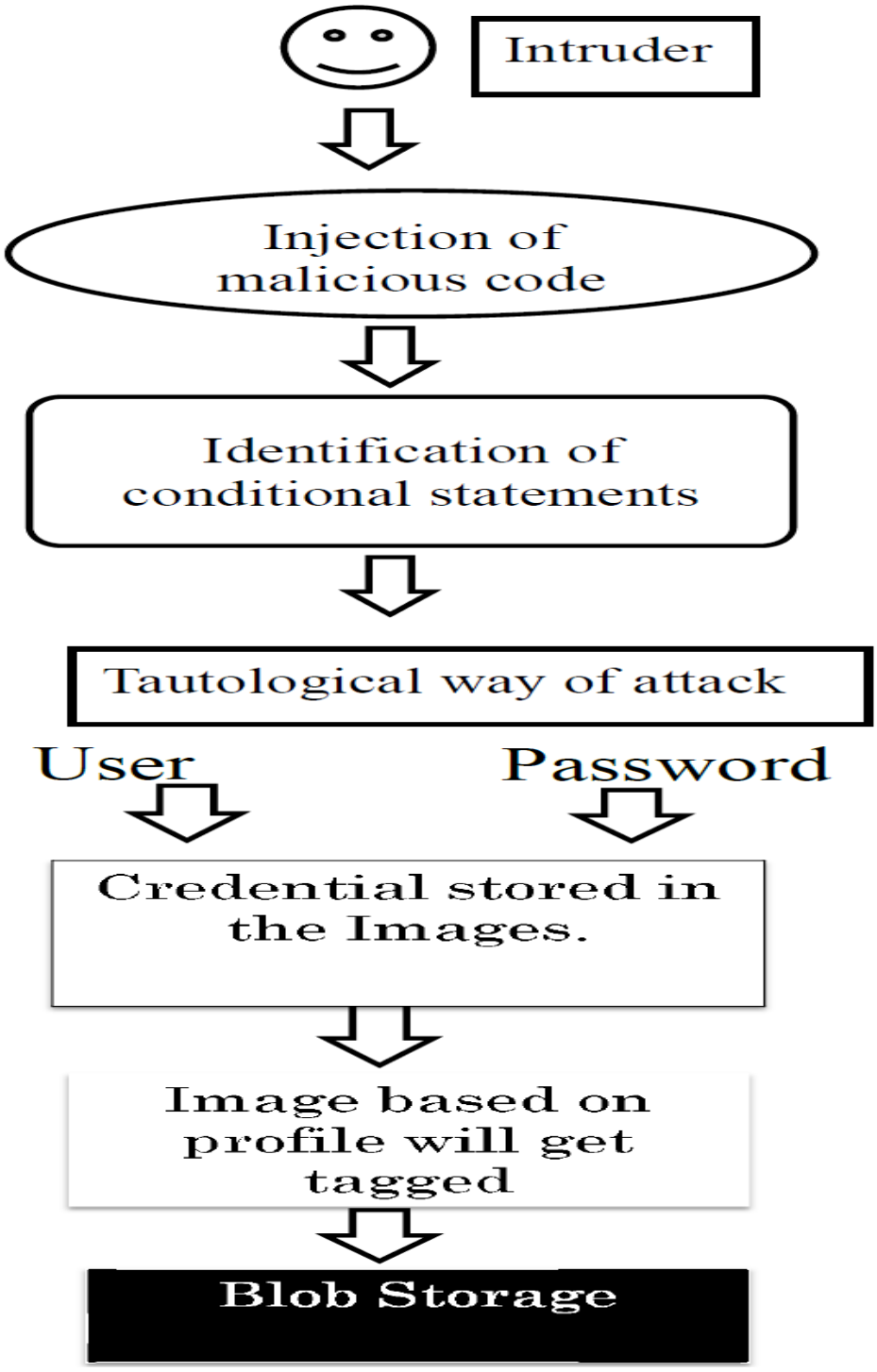
- Arbitrarily modify values in a database through a type of code injection called SQL injection. The impact of this can range from website defacement to serious compromise of sensitive data.
- Install malware or execute malevolent code on a server, by injecting server scripting code such as PHP or ASP.
- Privilege escalation to root permissions by exploiting Shell Injection vulnerabilities in a setuid root binary on UNIX, or Local System by exploiting a service on Windows.
- Attacking web users with HTML/Script Injection

In Tautological way of attack, the intruders will be injecting the data in the conditional statements. So that, the condition is always be true.

Module describes the interface implemented by authentication technology providers; here the user is allowed to create his credentials to login into the system. An admin will approve the users created and login approval the user will be allowed to log into the system.

Users can able to sign on because this associates content they create with their account and allows various permissions to be set for their roles.Users can use their own name or handle and can fine tune some personal configuration settings through their individual my account page. It will be allowed to log into the system.Users can able to sign on because this associates content they create with their account and allows various permissions to be set for their roles.Users can use their own name or handle and can fine tune some personal configuration settings through their individual my account page.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 3, pp: (349-356), Month: July - September 2014, Available at: www.researchpublish.com



Hash functions are primarily used to generate fixed-length output data that acts as a shortened reference to the original data. This is useful when the original data is too cumbersome to use in its entirety.

One practical use is a data structure called a hash table where the data is stored associatively. Searching linearly for a person's name in a list becomes cumbersome as the length of the list increases, but the hashed value can be used to store a reference to the original data and retrieve constant time barring collisions.

Another use is in cryptography, the science of encoding and safeguarding data. It is easy to generate hash values from input data and easy to verify that the data matches the hash, but for certain hash functions hard to 'fake' a hash value to hide malicious data. This is the principle behind the PGP algorithm for data validation.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 3, pp: (349-356), Month: July - September 2014, Available at: www.researchpublish.com

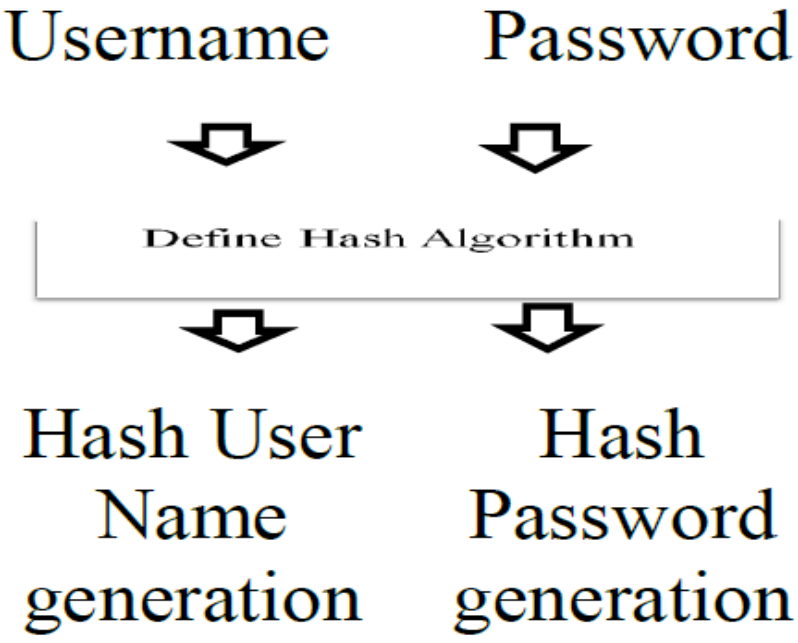
Hash functions are also frequently used to accelerate table lookup or data comparison tasks such as finding items in a database, detecting duplicated or similar records in a large file and finding similar stretches in DNA sequences.

Blind SQL Injection is used when a web application is vulnerable to an SQL injection but the results of the injection are not visible to the attacker. The page with the vulnerability may not be one that displays data but will display differently depending on the results of a logical statement injected into the legitimate SQL statement called for that page. This type of attack can become time-intensive because a new statement must be crafted for each bit recovered. There are several tools that can automate these attacks once the location of the vulnerability and the target information has been established.

A hash function should be deterministic: when it is invoked twice on identical data two strings containing exactly the same characters, the function should produce the same value. This is crucial to the correctness of virtually all algorithms based on hashing. In the case of a hash table, the lookup operation should look at the slot where the insertion algorithm actually stored the data that is being sought for, so it needs the same hash value.

Hash functions are designed to minimize the probability of collisions. For cryptographic uses, hash functions are engineered in such a way that it is considered impossible to reconstruct any input from the hash alone without spending great amounts of computing time.

In cryptography and steganography, plausibly deniable encryption is encryption that allows its users to convincingly deny that some specific encrypted data exists, that a given piece of data is encrypted, or that they are able to decrypt a given piece of encrypted data. Such denials may or may not be genuine. For example, although suspicions might exist that the data is encrypted, it may be impossible to prove it without the cooperation of the users. If the data is encrypted, the users genuinely may not be able to decrypt it. Deniable encryption serves to undermine an attacker's confidence either that data is encrypted, or that the person in possession of it can decrypt it and provide the associated plaintext.



3. XSD validation modul

In this module, the XML data transformed will be validated with the defined XSD model. XSD format needs to be defined so that the data needs to be validated for their types and its value range. Activity diagram are typically used for business process modeling for modeling the logic captured by a single use case or usage scenario, or for modeling the detailed logic of a business rule. Although UML activity diagrams could potentially model the internal logic of a complex operation it would be far better to simply rewrite the operation so that it is simple enough that you don't requires an activity diagram. In many ways UML activity diagrams are the objects-oriented equivalent of flow charts and data flow diagrams (DFDs) from structured development.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 3, pp: (349-356), Month: July - September 2014, Available at: www.researchpublish.com

III. MOBFUSCATION

Obfuscation means "to make difficult to perceive or understand". Code obfuscation in programming world means making code harder to understand or read, generally for privacy or security purposes. Security through obscurity has long been viewed with disregard in the security community. However, there are applications where obscurity can provide a higher level of protection to its source code.

Code Obfuscation

Obfuscation means "to make difficult to perceive or understand". Code obfuscation in programming world means making code harder to understand or read, generally for privacy or security purposes. Security through obscurity has long been viewed with disregard in the security community. However, there are applications where obscurity can provide a higher level of protection to its source code.

Code obfuscation can be achieved through one or more of the following methods:

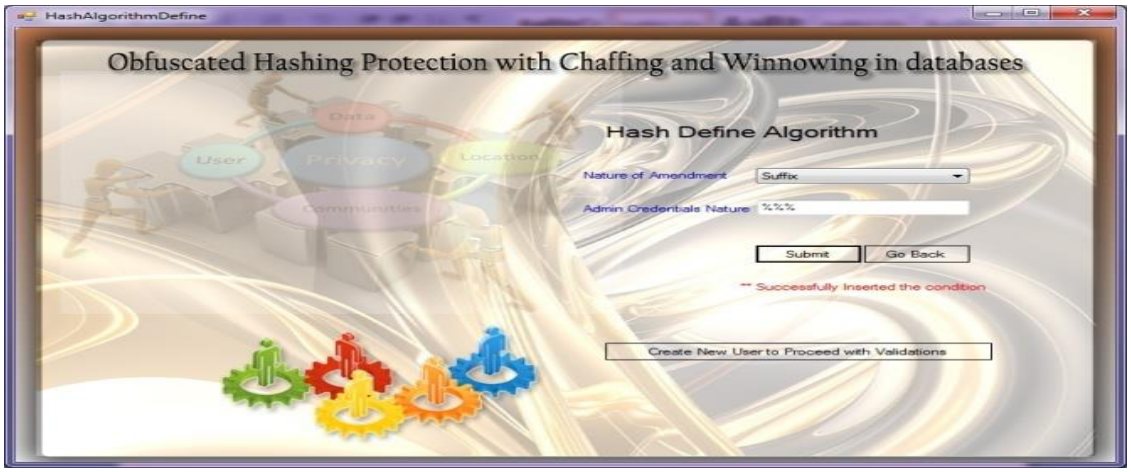
- **Source or binary structure obfuscation** - A source code obfuscator accepts a program source file, and generates another functionally equivalent source file, which is much harder to understand or reverse-engineer. This is useful for technical protection of intellectual property when source code must be delivered for public execution purposes.
- **Data Obfuscation** - This is aimed at obscuring data and data structures. Techniques used in this method range from splitting variables, promoting scalars to objects, converting static data to procedure, change the encoding, changing the variable lifetime etc.
- **Control Flow Obfuscation** - This aims at changing the control hierarchy with logic preservation. Here false conditional statements and other misleading constructs are introduced to confuse decompilers, but the logic of the code remains intact.
- **Preventive Obfuscation** - Here the focus is on protection against decompilers and reverse engineering methods. Renaming metadata to gibberish or less obvious identifiers is one such technique, like defining function Interest Calculation() as x().

IV. HASH DEFINE ALGORITHM

The user must login before used this technology. Therefore authorized users who have access rights to open, his data are already stored in the database. He only open this by clicking Login. And also to exit he chooses an adjacent button Exit.

The login form shows the below data as User name and password.

User name is a name where a data already register when this technology developed and the password is a key to encrypt the further contents in authenticate mode.



V. CONCLUSION

In this paper we have reviewed the most popular existing SQL Injections related issues. The proposed approach presents a new series of techniques to secure the authentication process of the database. Multi angle checks in securing the system with step by step process and data validation enable this system a more secure one. Concept of "Anti Spoofing" will avoid spam's through automated machines. Data filtration using XSD schema provides an additional filtration at the data level.

In today's information age, information sharing and transfer has increase exponentially. The treat of an intruder accessing secret information has been an ever existing concern for the data communication experts. Cryptography and steganography the most widely used technology to overcome this threat.

We can implement the system to protect our database from session oriented hacking. The cookies level injection can be detected and avoid those type of hackings.

REFERENCES

[1] E.BIHAM E.BARMAN AND N.KELLER. Instant cipher text only cryptanalysis of gsm."Advances in Cryptology – CRYPTO" 2003,2729 of LNCS: 600.616,August 2003.

[2] CHARLES BROOKSON. "Authentication: A mechanism for preventing man – in – the – middle attacks." Technical Report S3- 040036,3GPP Technical Specification Group Services and System Aspects, Edinburgh, Scotland,UK,2004.

[3] X. FU, X. LU, B. PELTSVERGER, S. CHEN, K. QIAN, AND L. TAO. "A Static Analysis Framework for Detecting SQL Injection Vulnerabilities", COMPSAC 2007, pp.87-96, 24-27 July 2007.

[4] ROICHMAN, A., GUDES, E.: "Fine-grained Access Control to Web Databases." In: Proc. of 12th SACMAT Symposium, France (2007).

[5] K. KEMALIS, AND T. TZOURAMANIS (2008). SQL-IDS: "A Specification-based Approach for SQL injection Detection."SAC'08. Fortaleza, Ceará, Brazil, ACM: pp. 2153 2158.

[6] S. THOMAS, L. WILLIAMS, AND T. XIE, On automated prepared statement generation to remove SQL injection vulnerabilities. "Information and Software Technology" 51, 589–598 (2009).

[7] M. JUNJIN, "An Approach for SQL Injection Vulnerability Detection," Proc. of the 6th Int. Conf. on Information Technology: New Generations, Las Vegas, Nevada, pp. 1411-1414, April 2009.

[8] Y. HAIXIA, N. ZHIHONG, "A database security testing scheme of web application," Proc. of ICCSE '09 , pp. 953-955, 25-28 July 2009.

[9] M. RUSE, T. SARKAR AND S. BASU. "Analysis & Detection of SQL Injection.