International Journal of Emerging Trend in Engineering and Basic Sciences (IJEEBS)
ISSN (Online) 2349-6967
Volume 2 , Issue 1(Jan-Feb 2015), PP264-269

PREVENTION AND DETECTION OF SQL INJECTION ATTACK AND INTRUSION (USING PATTERN MATCHING ALGORITHM)

Abhilasha Kumari[1] ,Pooja Jambhale[2] , Pragati kakade[3] , Avinash kharade[4]

[1 2 3 4](Dept.of I.T. ISB&M School Of Technology NANDE, Pune University, Maharashtra, India

ABSTRACT:- Pattern matching is a technique that can be used to identify or detect any anomaly packet from a sequential action. Injection attack is a method that can inject any kind of malicious string or anomaly string on the original string. Most of the pattern based techniques are used static analysis and patterns are generated from the attacked statements. In this paper, we proposed a detection and prevention technique for preventing SQL Injection Attack (SQLIA) using Aho–Corasick pattern matching algorithm. In this paper, we proposed an overview of the architecture. In the initial stage evaluation, we consider some sample of standard attack patterns and it shows that the proposed algorithm is works well against the SQL Injection Attack.Web applications are typically interact with backend database to retrieve persistent data and then present the data to the user as dynamically generated output, such as HTML web pages. This communication is commonly done through a low– level API by dynamically constructing query strings with in a general purpose programming language. This low–level interaction (or) communication is dynamic (or) session based because it does not take into account the structure of the output language. The user input statements are treated as isolated lexical entries (or) string. Any attacker can embed a command in this string, which poses a serious threat to web application security. SQL Injection Attack (SQLIA) is one of the very serious threats for web applications. The web applications that are vulnerable to SQL Injection may allow an attacker to gain complete access to the database. In some cases, attacker can use SQL injection attack to take control and corrupt the system that hosts the web application. SQL injection refer to a class of code–injection attacks in which data provided by the user is included in an SQL query of such a way that part of the user's input is treated as SQL code.

Keywords - Pattern Matching, SQL Injection, SQL Injection Attacks, SQL Queries.

I. INTRODUCTION

The pattern matching is a technique that can be used to determine or recognized any anomaly packet from a sequence action. Injection attack is a methodology that could be inject any kind of malicious string or anomaly string on the original string. Most of the pattern based techniques used the static analysis and the patterns are generated from the attacked statement. To detect or preventing the SQL Injection, introduced SQL Injection Attack using the Aho-Corasick pattern matching algorithm and also introduced the overview of the architecture. Some sample of standard attack patterns and it shows that the proposed algorithm is well work against the SQL Injection.

To provide the better services to the user's, companies and organization using the web applications. In the web application, database mostly contains the confidential and personal information. Attack to the target these databases and personal information will help. To retrieve the persistence data, web application are typically interact with the databases and then present the data to the user, the result which is given to the user is dynamically generated by the HTML web pages. This communication is commonly completed through a low level API. Low level is a dynamic and session based. Whatever user inputting the data which is treated as isolated lexical entries and like a string. Any attacker can embed a command in this string, which poses a serious threat to web application security. SQL Injection Attack (SQLIA) is one of the very serious threats for the web applications.

A. SQL Injection

SQL Injection is one of the many web attack mechanisms used by hackers to steal data from organizations. It is perhaps one of the most common application layer attack techniques used today. It is the type of attack that takes advantage of improper coding of your web applications that allows hacker to inject SQL commands into say a login form to allow them to gain access to the data held within your database. In essence, SQL Injection arises because the fields available for user input allow SQL statements to pass through and query the database directly.

B. Aho-Corasick Algorithm

The Aho–Corasick [1] string matching algorithm is a string searching algorithm. It is a dictionary-matching algorithm that locates elements of a finite set of strings (the "dictionary") within an input text. It matches all patterns simultaneously. Informally, the algorithm constructs a finite state with additional links between the various internal nodes. These extra internal links allow fast transitions between failed pattern matches to other branches that share a common prefix. This allows the automaton to transition between pattern matches without the need for backtracking.

II. RELATED WORK

Karma et al [7], [8] proposed an enhanced model that can identify intruders in databases where there are no roles associated with each user. Bertino [4] proposed a framework based on anomaly detection technique and association rule mining to identify the query that deviates from the normal database application behavior. Bandhakavi[3] proposed a misuse detection technique to detect SQLIA by discovering the intent of a query dynamically and then comparing the structure of the identified query with normal queries based on the user input with the discovered intent. Halfond developed a technique that uses a model–based approach to detect illegal queries before they are executed on the database. William proposed a system WASP to prevent SQL Injection Attacks by a method called positive tainting. Srivastava offered a weighted sequence mining approach for detecting data base attacks.

Pandey et al. [2] proposed for securing web applications. Different application level attacks are prevented through these applications. The Secure Web Applications Project (SWAP) is an interdisciplinary research initiative between the Laboratory for Communications Engineering and the Computer Laboratory of the University of Cambridge that seeks to address the problem of application-level Web security at a higher level. The project reduces application development time and protects against a large class of application-level attacks more effectively than existing Web development methodologies.

Axelsson et al. [9] proposed a system to construct the taxonomy of intrusion detection principles proves to be a fruitful exercise that provides us with many insights into the field, and a number of lines for further research. When completing the survey one is struck by the lack of research earlier in the detection chain, into taxonomies and models of intrusive and normal behavior, and into what information to observe in order to detect intrusions, etc. self-learning class is conspicuous, particularly since detectors in this class would probably prove very useful, combining the advantages of self-learning systems not having to perform the arduous and difficult task of specifying intrusion signatures. With the detection efficiency of signature based systems. Two tiered detectors also show promise. Some of the oldest systems are designed according to these principles, but there is little research into the specific abilities of such systems. In more general terms, signature based systems with a more explicit normal behavior model and anomaly based systems with a better formed intrusion/attack model are of interest, if only for the fact that current systems are, almost without exception, firmly entrenched in one camp or the other.

Ezeife et al. [5] proposed a system called, Sensor Web IDS, has three main components: the Network Sensor for extracting parameters from real-time network traffic, the Log Digger for extracting parameters from web log files and the Audit Engine for analyzing all web request parameters for intrusion detection. To combat web intrusions like buffer-over-flow attack, Sensor Web IDS utilizes an algorithm based on standard deviation theory's empirical rule of 99.7% of data lying within 3σ of the mean, to calculate the possible maximum value length of input parameters. Association rule mining technique is employed for mining frequent parameter list and their sequential order to identify intrusions.

Yusufovna [10] proposed the application of data mining approaches for an intrusion detection system. Intrusion detection is the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of the resource of a computer system. In this paper, an IDS model is presented as well as its limitation in determining security violations. Furthermore, this paper focuses on several data mining techniques that can aid in the process of intrusion detection.

III. EXISTING SYSTEM

The existing approach [6] works by combining static analysis and runtime monitoring. The key intuition behind the approach is that;

(1) The source code contains enough information to infer models of the expected, legitimate SQL queries generated by the application, and
(2) An SQLIA, by injecting additional SQL statements into a query, would violate such a model.

In its static part, the technique uses program analysis to automatically build a model of the legitimate queries that could be generated by the application. In its dynamic part, the technique monitors the dynamically generated queries at runtime and checks them for compliance with the statically-generated model. Queries that violate the model represent potential SQLIAs and are thus prevented from executing on the database and reported.

The technique consists of four main steps. We summarize the steps and then describe them in more detail in subsequent sections.

- **Identify hotspots:**

  Scan the application code to identify *hotspots*—points in the application code that issue SQL queries to the underlying database.

- **Build SQL-query models:**

  For each hotspot, build a model that represents all the possible SQL queries that may be generated at that hotspot. A *SQL-query model* is a non-deterministic finite-state automaton in which the transition labels consist of SQL tokens (SQL keywords and operators), delimiters, and place holders for string values.

- **Instrument Application:**

  At each hotspot in the application, add calls to the runtime monitor.

- **Runtime monitoring:**

  At runtime, check the dynamically-generated queries against the SQL-query model and reject and report queries that violate the model.

IV. PROPOSED SYSTEM

A. Existing System Limitations

In existing system, the model of legitimate queries is built automatically and then checks the dynamically entered queries with these statically-generated models. The problem occurs if the some new legitimate queries are developed, then someone have to enter those queries in the static model. Until that point, the system will take those legitimate queries as adversary queries.

B. Our Solution

For the above mentioned problem, the best solution is to keep updating the statistically generated model automatically. Thus, a technique that uses the user queries and add new queries from those entered queries in to the generated model. This will make the system more flexible and efficient.

C. Our Contributions

We proposed a detection and prevention technique for preventing SQL Injection Attack (SQLIA) using Aho–Corasick pattern matching algorithm. In this paper, we proposed an overview of the architecture. In the initial stage evaluation, we consider some sample of standard attack patterns and it shows that the proposed algorithm is works well against the SQL Injection Attack.

The proposed scheme has the following two modules, Static Phase and Dynamic Phase.

- *Static Phase:*

In the Static Pattern List, we maintain a list of known Anomaly Pattern. In Static Phase, the user generated SQL Queries are checked by applying Static Pattern Matching Algorithm.

- *Dynamic Phase:*

In Dynamic Phase, if any form of new anomaly is occur then Alarm will indicate and new Anomaly Pattern will be generated. The new anomaly pattern will be updated to the Static Pattern List.

D. Mathematical Model

In this paper, the proposed model is formulated as follows.

In this, for the Static Pattern List with stored anomaly patterns, the anomaly score is calculated. Depending upon these anomaly scores, the new queries are added in the List.

1. SPMA(Query, SPL[])

Query - It is user Generated Query

SPL [] - Static Pattern List with anomaly Patterns

SPMA() Function returns if the query is found in Static Pattern list then Returns its specified result from the list Otherwise It is given to the next Phase OF Anomaly Score Generation.

2. $ANOMALY_{SCORE} =$

$$\frac{MATCHING_{VALUE}\,(Query, SPL[j])}{StringLength(SPL[j])} * 100$$

Here,

Anomaly $_{score}$ - Score of the Current SQL Query

String length - Query String length

3. AC(y,n,q0)

y - Array of m bytes representing the text Input (SQL Query Statement)

n - Integer representing the text Length

q0 - Initial State

Proposed System Algorithm

- *Static Phase*

Step 1. User generated SQL Query is sending to the proposed Static Pattern Matching Algorithm.
Step 2. The Static Pattern Matching is performed.
Step 3. The Anomaly patterns are maintained in Static Pattern List, during the pattern matching process each pattern is compared with the stored Anomaly Pattern in the list.
Step 4. If the pattern is exactly match with one of the stored pattern in the Anomaly Pattern List then the SQL Query is affected with SQL Injection Attack.

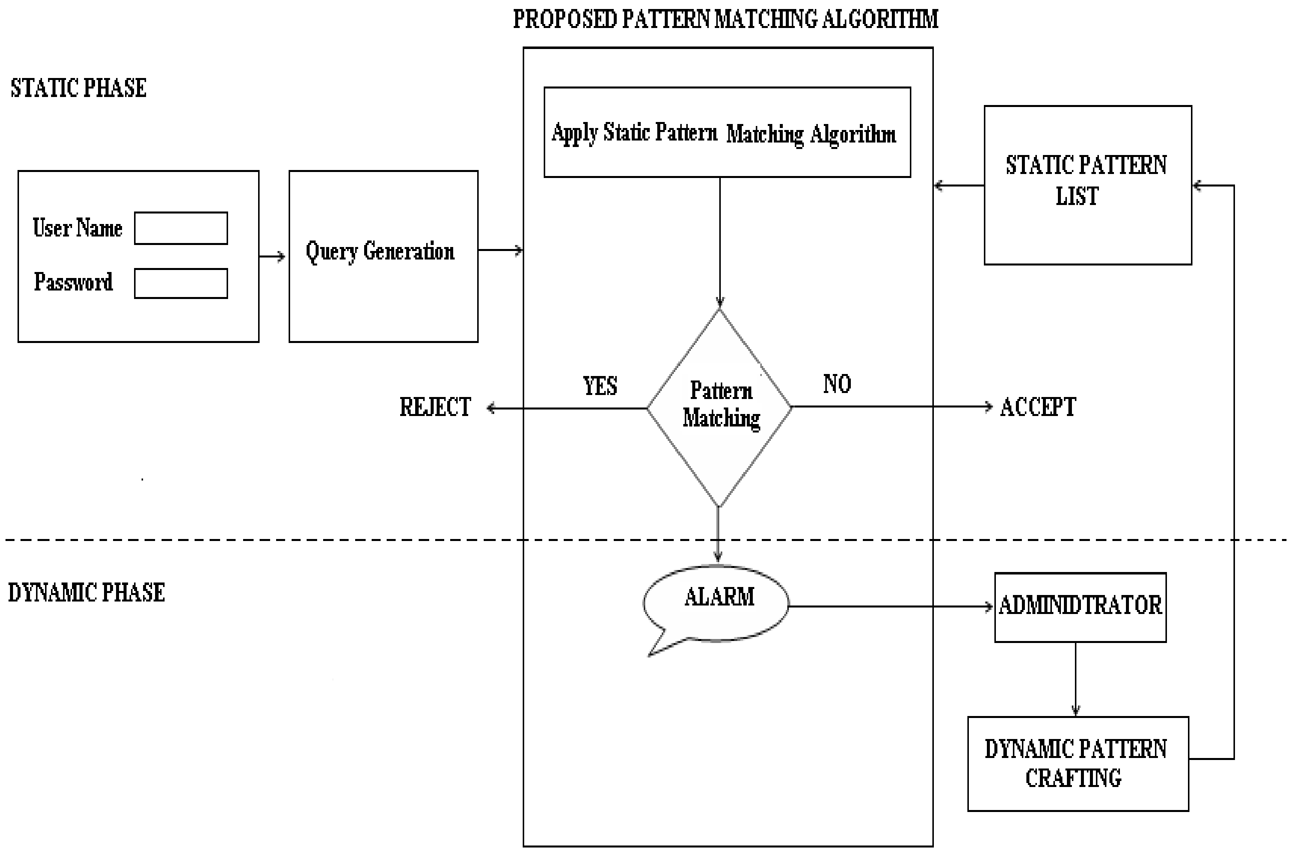

Fig. 1 System Architecture

Step 1. Otherwise, Anomaly Score value is calculated for the user generated SQL Query, If the Anomaly Score value is more than the Threshold value, then a Alarm is given and Query will be pass to the Administrator.
Step 2. If the Administrator receives any Alarm then the Query will be analyze by manually. If the query is affected by any type of injection attack then a pattern will be generated and the pattern will be added to the Static Pattern list.

E. System Architecture

- o Set Network
- o Set User
- o Set Database
- o Set Anomaly

- Dynamic Phase

SQLIA is a new kind of attack that penetrates the user queries that has been forwarded towards the servers and databases. This attack is basically based on the belief of the people that the queries cannot be compromised. Thus, in this paper, we have proposed a novel scheme for detecting and preventing the SQLIA. In this technique, we have used the Aho-Corasick string matching algorithm. The important aspect of this technique is that, it does not generate the false positive results.

Though, this techniques tackles down the threat of SQLIA, we do not assure the complete solution against SQLIA, as the attacks are always improvised with some new techniques. Thus, the future work is always welcome.

REFERENCES

[1]. Alfred V. Aho, M. J. Corasick, “Efficient String Matching: An Aid to Bibliographic Search”, Communications of the ACM 18(6): 333-340.
[2]. Amit Kumar Pandey, “SECURING WEB APPLICATIONS FROM APPLICATION-LEVEL ATTACK”, master thesis, 2007.
[3]. Bandhakavi, S., Bisht, P., Madhusudan, P., and Venkatakrishnan V., “CANDID: Preventing sql injection attacks using dynamic candidate evaluations”, in the Proceedings of the 14th ACM Conference on Computer and Communications Security, 2007
[4]. Bertino, E., Kamra, A, Terzi, E., and Vakali, A, “Intrusion detection in RBAC-administered databases”, in the Proceedings of the 21st Annual Computer Security Applications Conference, 2005.
[5]. C.J. Ezeife, J. Dong, A.K. Aggarwal, “SensorWebIDS: A Web Mining Intrusion Detection System”, International Journal of Web Information Systems, volume 4, pp. 97-120, 2007.
[6]. Halfond, W. G. and Orso, A , “AMNESIA: Analysis and Monitoring for Neutralizing SQL-Injection Attacks”, in Proceedings of the 20th IEEE/ACM international Conference on Automated Software Engineering, 2005.
[7]. Kamra A, Bertino, E., and Lebanon, G.,”Mechanisms for Database Intrusion Detection and Response”, in the Proceedings of the 2nd SIGMOD PhD Workshop on Innovative Database Research, 2008.
[8]. Kamra A, Terzi E., and Bertino, E.,“Detecting anomalous access patterns in relational databases”, the VLDB Journal VoU7, No. 5, pp. 1063-1077, 2009.
[9]. S.Axelsson, “Intrusion detection systems: A survey and taxonomy”, Technical Report, Chalmers Univ., 2000.
[10]. S. F. Yusufovna., “Integrating Intrusion Detection System and Data
[11]. Mining”, International Symposium on
Ubiquitous Multimedia Computing, 2008.