IJCSE International Journal of Computer Sciences and Engineering Open Access
ReviewPaper Volume-4, Special Issue-2, April 2016 E-ISSN: 2347-2693

Protecting Websites from Dissociative Identity SQL Injection Attack- a Patch for Human Folly

Aman Prasad[1], Gangasagar Pitla[2], Satish Patil[3], Prof. Rachana Patil[4]

[1]*Computer Engineering (BE), Mumbai University, India*
[2]*Computer Engineering (BE), Mumbai University, India*
[3]*Computer Engineering (BE), Mumbai University, India*
[4]*Computer Engineering (ME), Mumbai University, India*

Available online at: www.ijcseonline.org

***Abstract*—** 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 project, we proposed a detection and prevention technique for preventing SQL Injection Attack (SQLIA) using Aho-Corasick pattern matching algorithm. The basic goal of our project is to minimize the web-based attacks like SQL Injection Attack (SQLIA) and reduce the load of server.

***Keywords*—**SQL Injection; SQL Injection Attack; Aho-Corasick Algorithm; Anomaly Scoring;

I. INTRODUCTION

Companies and organizations use web applications to provide better service to the end users. The Databases used in web applications often contain confidential and personal information .These databases and user personal information is target to the attacks. 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 within 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 possess 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 users input is treated as SQL code. SQL injection is a technique offer used to attack a website. This is done by including portions of SQL statements in a web application entry field in an attempt to get the website to pass a newly formed rogue SQL command to the database. SQL Injection is a code injection technique that exploits security vulnerability in website software. The vulnerability happens when user input of either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. One of the most efficient mechanisms to defend against web attacks uses Intrusion Detection System (IDS) and Network Intrusion Detection System (NIDS). IDS use misuse or anomaly detection to defend against attack. IDS that use anomaly detection technique establish a base line of normal usage patterns. Misuse detection technique uses specifically known patterns of unauthorized behaviour to predict and detect subsequent similar kind of attacks. These kinds of patterns are called as signatures. NIDS are not support for the service oriented applications (web attack).

II. RELATED WORK

The current system is very complex, lengthy and time consuming. It is easy for any user to get access to the database of the system as the existing system has some limitations that said as follows:

1. Improper filtering of input given by users.
2. Existing System does not use PREPARE statement.
   E.g. SELECT * FROM user WHERE username=? AND password=?
3. Lacks of security.
4. Anomaly calculations takes more time.
5. No proper String matching algorithm used.
6. Use of tokenization for detecting SQLi.
7. Increased load on server.

The main focus of implementation of this is to prevent and detect SQL injection attack and try create a patch for human folly.

III. PROPOSED SYSTEM

Architecture of proposed system i.e. the patch file flow where the information is being analyzed. It contains two phases viz.

1. Static Phase.
2. Dynamic Phase.

Static Phase

Step 1: User generated SQL Query is send to the proposed Static Pattern Matching Algorithm.

Step 2: Aho-Corasick algorithm is used as the Static Pattern Matching Algorithm which creates the automated DFA for the query generated string.

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.

Dynamic Phase:

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 an 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. [1]

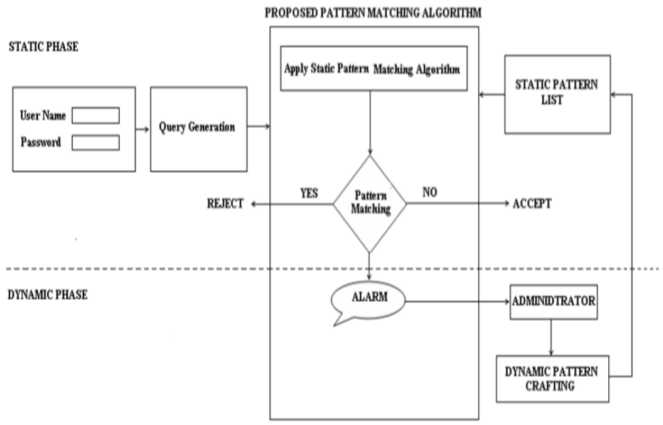

Fig: Proposed System (Referred from [1])

A. *Aho-Corasick Algorithm*

Aho-Corasick algorithm is one of the earliest multi-pattern exact matching algorithms. Aho-Corasick algorithm is a direct extension of the KMP algorithm by combining with the automata. [2] The UNIX operating provides standard texts (or files) facilities. Among them is the series of grep command that locate patterns in files. We describe in this section the algorithm underlying the fgrep command of UNIX. It searches files for a finite set of strings and can for instance outputs lines containing at least one of the strings. If we are interested in searching for all the occurrences of all patterns taken from a finite set of patterns, a first solution consists in repeating some string-matching algorithm for each pattern. If the set contains k patterns, this search runs in time O(kn). Aho and Corasick designed an algorithm to solve this problem in O(nlog∂). The running time is independent of the number of patterns. The state machine starts with an empty root node which is the default nonmatching state. Each pattern to be matched adds states to the machine, starting at the root and going to the end of the pattern. The state machine is then traversed and failure pointers are added from each node to the longest prefix of that node which also leads to a valid node in the trie. We show a single node of the state machine in (Fig. Example Aho–Corasick). Aho-Corasick Node Selection Given a set of patterns = {search, ear, arch, chart}, Fig.1 shows the state machine and goto function. Aho-Corasick algorithm scans the character in text one by one without any jump.

B. *System Requirement*

Hardware Requirements

1. Pentium IV processor.
2. 1 GB RAM.
3. 80 GB Hard Disk.

Software Requirements

1. PHP(5.5)-MYSQL
2. Apache
3. PHP-Java Bridge
4. Tomcat
5. BIRT Environment

C. *N-Tier Architecture*

To minimize the load on the server we have introduced n-tier architecture that indents to provide an assistant to the main server for resolving the malicious string input and also works effectively with the Anomaly Scoring of the input string.

D. *Anomaly Scoring*

In the static phase, each anomaly pattern from the Static Pattern List is checked with the user generated query. The proposed scheme will calculate the Anomaly Score value of the query for each pattern in the Static Pattern List. If the Query is match 100% with any of the pattern from the Static Pattern List, then Query is affected with SQL Injection Attack. Otherwise, the high matching score is called as an

Anomaly Score value of a query. If the Anomaly Score value is more than the Threshold value (assume that 50%), then the query will be transfer to the Administrator. Anomaly scoring provides following benefits in detecting Intrusion attack: An increased confidence in blocking - since more detection rules contribute to the anomaly score, the higher the score, the more confidence you can have in blocking malicious transactions. Allows users to set a threshold that is appropriate for them - different sites may have different thresholds for blocking. Allows several low severity events to trigger alerts while individual ones are suppressed. One correlated event helps alert management. Exceptions may be handled by either increasing the overall anomaly score threshold, or by adding rules to a local custom exceptions file where TX data of previous rule matches may be inspected and anomaly scores re-adjusted based on the false positive criteria.

Following graph shows the result of introducing anomaly scoring.

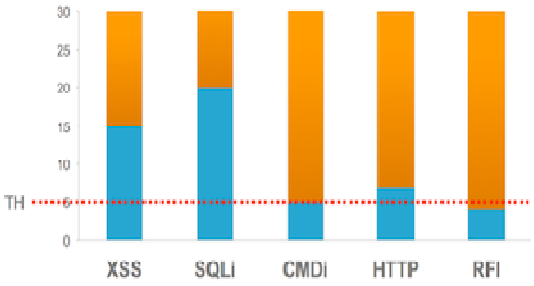

Fig: - Analysis of Different Attack after introducing Scoring

IV. RUN-TIME ANALYSIS

Approximate String matching is a problem in computer science which is applied in text searching, pattern recognition and signal processing applications. For a text T[1..n] and pattern P[1...m], we are supposed to find all the occurrences of pattern in the text whose edit distance to the pattern is at most K. The edit distance between two strings is defined as minimum number of character insertion, deletion and replacements needed to make them equal.

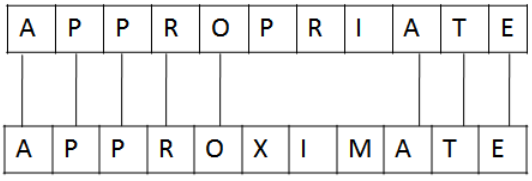

Fig: Approximate String Matching Example

Here K(T,P) = 3.

Approximate string matching problem is solved with the help of dynamic programming.

Element	Search type	Time Complexity	Multiple String
Brute Force	Prefix	O((n-m+1)m)	*No*
Rabin Karp	Prefix	Θ(m),Θ(n+m)	*No*
Boyer-Moore	Suffix	O(M+|Σ|),O(n)	*No*
Kruth-Morris-Pratt	Prefix	O(m),O(n+m)	*No*
Aho-Coarsick	Prefix	O(n),O(m+z): Z=no. Of matches	*Yes*
Commertz Walter	Suffix	-	*Yes*

Table: Comparative analysis of Different Algorithm.

This work categorizes the algorithms into various categories to emphasize the data structure that drives the matching. These categories are automaton-based, heuristics-based and hashing-based. An *automaton-based* algorithm builds a finite state automaton from the patterns in the pre-processing stage and tracks the partial match of the pattern prefixes in the text by state transition in the automaton A *heuristics-based* algorithm allows skipping some characters to accelerate the search according to certain heuristics. Some algorithms require a verification algorithm following a possible match to verify if a true match occurs. A *hashing based* algorithm compares the hash values of characters in the text segment by segment with those of the characters in the patterns. If both hash values are equal, a possible match may occur. The characters in the text and those in the patterns are then compared to verify if a true match occurs. The main advantage of the Aho-Corasick algorithm is that it runs in linear time to the input patters regardless of the number of patterns and gives the beast result for the multiple pattern matching strings. However, the main disadvantage lies in devising a practical implementation due to the large memory needed to store the FSM. So one of the primary area of focus in the IDS area is in devising a performance and area efficient strategy for the Aho-Corasick algorithm.

CONCLUSION

SQL injection attack has characteristics of simplicity of operator, high performance of concealment, serious hazard and hard to being detected. It has become most serious hazard To web application system. The deployment of our

project which advances the technology and will try to minimize the risk of SQL Injection attacks on web based application. This will ensure more security to user and web server's details. Later the file will be made compatible with other web scripting languages. Aho-Corasick builds an automata of size proportional to the sums of the lengths of all of the substring patterns, and then running it over an input string in a single pass and gives substring matches.

ACKNOWLEDGEMENT

I thank my guide Prof. (Mrs) Rachana Y Patil whose guidance, support and dedication is priceless. I especially thank to IEEE (Institute of Electrical and Electronics Engineers, Inc.) for providing recent information about my document.
It has been a challenging, yet rewarding journey which I couldn't have completed alone and I am grateful for my team member's support.

REFERENCES

[1] Aho, Alfred V.; Margaret J. Corasick (June 1975). "Efficient string matching: An aid to bibliographic search". Communications of the ACM 18 (6): 333–340
[2] Commentz-Walter B. A string matching algorithm fast on the average, Proc. 6th International Colloquium on Automata, Languages, and Programming (1979), pp. 118-132.
[3] Alsmadi I., Nuser M., String Matching Evaluation Methods for DNA Comparisons, International Journal of Advanced Science and Technology, Vol.47, 2012.
[4] Amir A., Lewenstein M., and Porat E., Faster Algorithms for String Matching with K-Mismatches, Journal of Algorithms 50(2004) 257-275.
[5] Gomaa N.H., Fahmy A.A., Short Answer Grading using String Similarity and Corpus-Based Similarity, International Journal of Advanced Computer Science and Applications, Vol 3,No.11, 2012.
[6] https://www.blogs.akamai.com/
[7] C.J. Ezeife, J. Dong, A.K. Aggarwal, "Sensor Web-IDS: A Web Mining Intrusion Detection System", International Journal of Web Information Systems, volume 4, pp. 97-120, 2007

AUTHORS PROFILE

Aman Prasad: He is persuing the BE degree in Computer Engineering from A. C. Patil College of Engineering, Kharghar, Navi Mumbai, Mumbai University. His research interests includes Internet and Web Technology, Cyber Securities, and Intelligent Networks.

Gangasagar Pitla: He is persuing the BE degree in Computer Engineering from A. C. Patil College of Engineering, Kharghar, Navi Mumbai, Mumbai University. His research interests includes Internet and Web Technology.

Satish Patil: He is persuing the BE degree in Computer Engineering from A. C. Patil College of Engineering, Kharghar, Navi Mumbai, Mumbai University. His research interests includes Internet and Web Technology, Business intelligence system.

Rachana Patil: She has received the ME degree in computer engineering from Mumbai University, Mumbai, India, in 2012.She is currently working as an Assistant Professor with Computer Engineering Department, A. C. Patil College of Engineering, Kharghar, Navi Mumbai. Her research interests include Computer Network .Wireless network Security, Business intelligence systems. Mobile communication and systems