International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 04 | Apr -2017 www.irjet.net p-ISSN: 2395-0072

Using AMNESIA to secure web applications and database against SQL injection attack

Disha Sharma[1], Komal Kale[2], Chandrakala Date[3], Prof. Diksha Bhave[4]

[1,2,3] B.E. Students, Department of Computer Engineering, Shivajirao S. Jondhale College of Engineering, Dombivli(E)
[4]Assistant Professor, Department of Computer Engineering, Shivajirao S. Jondhale College of Engineering, Dombivli(E)
Mumbai University, Maharashtra, India

Abstract - *In today's world use of the internet and web applications has become more and more common in our routine activities, like reading the news, paying bills, and shopping on-line. As the availability of these services grows, we are witnessing a rise within the variety and class of attacks that target them. The sensitive data of the user may get leaked from the database resulting in serious losses of liveliness and intellectual property. The current approach uses AMNESIA, which detects and prevents SQL injection attacks by combining static analysis and runtime monitoring.*

Key Words: **Internet, web applications, AMNESIA, static analysis, run time monitoring.**

1. INTRODUCTION

This SQLIA (Structured query language Injection Attacks) is type of code injection technique that targets the databases to steal information from the organizations. The attacker enters the malicious SQL commands into a SQL statement via the unconstrained user input parameters to manipulate the SQL queries logic. This leads to the threat to all those web applications that access their databases, through SQL commands establish with external input data. Through SQL injection the attacker neglects authentication phase and provides confidential information to the attacker. Authorized access to confidential information by a crafted user has unprotected their authority, confidentiality and integrity. The results could be like the system couldn't deliver proper services to its customers. During this paper, we present this method, which will act against all those malicious content and can actively work on those hotspots wherever injection may occur [1].

SQL injection vulnerabilities are due to poor input validation. As the input from the user to a web application leads to the creation of a database query but it does with poor validation, thus SQL injection occurs. An attacker uses this vulnerability of the application as an opportunity by enclosed malicious SQL commands within the input that are then executed by the databases. SQLIAs could also be prevented by plenty of application of defensive coding techniques. However, these techniques have been less than effective in addressing the matter as a result of they are in danger of human errors and expensive to use on large inheritance code-bases.

2. EXISTING SYSTEMS

When we mention the defense techniques that are being used, then we have a pair of defensive techniques specifically defensive coding and Runtime monitoring. Defensive coding has subclasses like Parameterized query Insertion, Manual Defensive coding practices, SQL DOM. This defensive coding technique ensured secure code but is labor intensive and time-consuming. Manual defensive coding practices are performed manually and might be finished with the assistance of OWASP. SQL DOM is helpful in terms of larger flexibility when developer needs to use the dynamic queries rather than parameterized one. Runtime checking could also be a technique used for against the illegitimate SQL statements for every variety of SQLIA's by checking them at the runtime. However, its disadvantage is that it needs a robust dynamic observation system.

3. PROPOSED SYSTEM

Our proposed solution is amnesia, which is technique that works by combining static and dynamic part for detecting web application vulnerabilities at the runtime. The concept behind this solution is that the source code contains enough knowledge to interpret models of the authorized SQL queries generated by the application. The static part of our technique uses program analysis to automatically build a model of the authorized queries which will be generated by the application. In its dynamic part, it supervises the dynamically generated queries at runtime and checks them for conformation with the model that was generated at the static part. SQL Queries that violate the model represent potential SQLIAs and are therefore prevented from execution on the database and noted. The technique consists of 4 main steps. We have describes the steps in summarized detail.

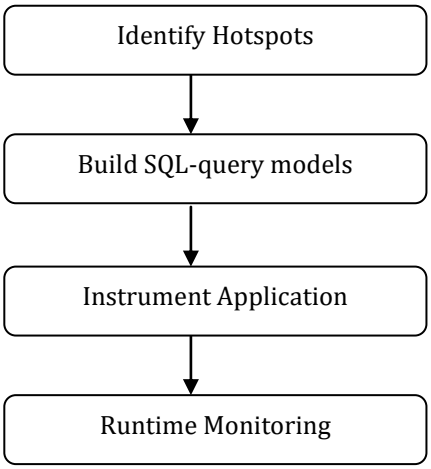

Fig -3.1: Basic flow of AMNESIA

3.1 Identify Hotspots

- This is the first step in AMNESIA technique.
- In this step, it scans the application code to identify hotspots within them that issue SQL queries to the underlying database.

3.2 Build SQL-query models

- For each hotspot, it builds a model that represents all the potential SQL queries that will be generated at that hotspot.
- A SQL-query model is a non-deterministic finite-state automaton during which the transition labels consist of SQL tokens, delimiters, and place holders for input values.
- To create our SQL-query model for a given hotspot, we perform a depth first traversal of the non-deterministic finite-state automaton for the hotspot and group characters as SQL keywords, operators, or literal values and build a transition within the SQL-query model that's annotated with their literal value.
- For instance, a sequence of transitions labeled 'S', 'E', 'L', 'E', 'C', and 'T' would be recognized because the SQL choose keyword and appropriately classified into one transition labeled "SELECT".
- As a result of there are many SQL dialects every with their own set of keywords and operators, this part of the technique may be customized to acknowledge different dialects.

3.3 Instrument Application

- In this step, we assist the application by adding calls to the monitor that check the queries at runtime.
- At each hotspot, this step injects a call to the monitor before the call to the database.
- The monitor is invoked with 2 parameters: - the string that contains the particular query on the point of be submitted and a unique identifier for the hotspot.
- Using the distinctive symbol, the runtime monitor is able to correlate the hotspot with the particular SQL-query model that was statically generated for that point and check the query against the proper model.

3.4 Runtime Monitoring

- At runtime, the application executes normally till it reaches a hotspot.
- At this time, the query string is sent to the runtime monitor. It parses the query string into a sequence of tokens with respect to the particular considered SQL dialect.
- After parsing the query, it checks whether or not the given query violates the hotspot's SQL-query model.
- To do this, the runtime monitor checks whether the model accepts the sequence of tokens within the query string.
- While matching the query string against the SQL-query model, a token that corresponds to a numeric or string constant including the empty string, can match either constant or literal value.
- If the model does not accept the sequence of tokens, the monitor identifies the query as an SQLIA.

4. APPLICATIONS

- It does not disclose elaborated error messages to the user.
- It constrains and purifies input by validating their type, length, format, range.
- It prevents the hacker to break into the system to retrieve information or do harm.
- It protects the integrity of websites and web applications.

5. CONCLUSIONS

Thus, we have developed a detection and prevention mechanism to protect the vulnerable website and its information from being exploited by the SQL Injection attacks. We have provided the usefulness of AMNESIA against real-world SQLIAs to protect the websites and its information. Thus this concludes that our proposed technique can be very useful and valuable in detecting and preventing SQLIAs.

6. FUTURESCOPE

In our future work we are going to study different techniques for building SQL models where the static analysis cannot be used. Our current approach, we use a very precise and expensive analysis to build the character-level model that we then compact and remodel into our SQL-query model. Our technique is generally interested in SQL keywords and operators, and the strings representing them are typically constant strings that are seldom manipulated within the application. Therefore, we could also be ready to use a simplified string analysis to extract the SQL-query models that our monitoring technique needs directly from an acceptable illustration of the code.

REFERENCES

[1] Shashwat Gupta,Saket S. Ektate,Deepak Yadav.Sachin Pitrubh,"Security against SQL injection attacks using AMNESIA," IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 4, April 2015.

[2] Diksha G. Kumar, Madhumita Chatterjee,"Detection Block Model for SQL Injection Attacks," I.J. computer Network and Information Security, 2014.

[3] Mihir Gandhi, Jwalant Baria, "SQL INJECTION Attacks in Web application,"International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-6, January 2013

[4] K.G.S. Venkatesan. Dr. V. Khanna, S.B. Amarnath Reddy, "Providing Security for social Networks from Inference Attack"',International Journal of Computer Science Engineering & Scientific Technology, March – 2015.

[5] Sayyed Mohammad Sadegh Sajjadi and Bahare Tajalli Pour,"Study of SQL Injection Attacks and Countermeasures," International Journal of Computer and Communication Engineering,Vol.2, No. 5, September 2013.

[6] Dr.Manju Kaushik,Gazal Ojha," SQL Injection Attack Detection and Prevention Methods: A Critical Review," International Journal of Innovative Research in Science, Engineering and Technology ISSN: 2319-8753, Vol. 3, Issue 4, April 2014.