Journal of Information Security and Applications 41 (2018) 75–91

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Token based Detection and Neural Network based Reconstruction framework against code injection vulnerabilities

Teresa K. George[a,*], K. Poulose Jacob[b], Rekha K. James[c]

[a] Department of Computer Science, Cochin University of Science and Technology, India
[b] Cochin University of Science and Technology, India
[c] Department of Electronics, Cochin University of Science and Technology, India

ARTICLE INFO

Article history:

Keywords:
Code injection attack
Neural network
Query validation
Reconstruction of queries
Security vulnerability
Web application

ABSTRACT

Security vulnerabilities are frequently detected and exploited in modern web applications. Intruders obtain unrestricted access to the information stored at the back-end database server of a web application by exploiting security vulnerabilities. Code injection attacks top the list due to lack of effective strategies for detecting and blocking injection attacks. The proposed Token based Detection and Neural Network based Reconstruction (TbD-NNbR) framework is a unique approach to detect and block code injections with negligible processing overheads. This framework makes use of an efficient token mapping and validation technique to match the statically generated legal query tokens against the parsed dynamic query tokens at run time. The proposed approach also has the provision to reconstruct queries from authenticated users. The prototype implementation of TbD-NNbR shows that it does not demand any source code modifications and incurs only a negligible computational overhead without any incidents of false positives or false negatives.

1. Introduction

Security vulnerabilities are becoming a severe issue in web applications as successful attacks lead to loss of integrity, confidentiality and make it a very sensitive subject in software security. Code Injection through a dynamic web page is one of the most dangerous threats that exploit the application layer vulnerabilities [1]. The existing techniques or strategies may not be enough to handle many of the vulnerabilities due to the unknown and often obscure nature of vulnerability issues. Existing input validation techniques still require more sophistication. The attack on a given database violates the Confidentiality, Integrity, Availability (CIA) triangle of security. Most of the SQL injection attack prevention approaches result in false positives, which will decrease the system availability for the authenticated users [2].

SQL injection attack is rated as one of the top 10 security vulnerabilities targeting back-end databases [3]. In these attacks, an attacker attempts to change the syntax and semantics of legitimate SQL statements by inserting unintended keywords, symbols or malicious codes on the SQL statements accepted through dynamic web pages. By exploiting this vulnerability, an attacker can directly interact with the database server and gain access to the critical data and thus compromise security. These types of attacks can evade traditional intrusion detection systems and firewalls and can breach the security mechanism of authentication, availability, accountability and confidentiality of the database. Even though vulnerability scanners and automated tools are available for verifying SQL Injection Attack (SQLIA), a single mechanism or procedure cannot efficiently handle the potential sophisticated attacks.

This paper proposes a Token based Detection and Neural Network based Reconstruction (TbD–NNbR) framework against code injection vulnerabilities. The proposed framework blocks all malicious entries and only the benign query can access the data from the back-end database server. The TbD–NNbR framework also has the provision to reconstruct the queries from authenticated users at run time, using the neural network, which increases the system availability and mitigates the denial of service attack [2]. A prototype has been designed and implemented using a Java-based application program to test the performance and the effectiveness of the system. Various online applications provide a comprehensive list of legal and injected queries. The proposed model creation and implementation use these queries [1,5].

The rest of the paper is organized as follows: Section 2 deals with SQL injection attack categories. Section 3 handles the related works. The proposed Token based Detection and Neural Net-

* Corresponding author.
E-mail addresses: susanteresa12@gmail.com (T.K. George), kpj0101@gmail.com (K.P. Jacob), rekhajames@cusat.ac.in (R.K. James).

https://doi.org/10.1016/j.jisa.2018.05.005
2214-2126/

work based Reconstruction (TbD–NNbR) framework are explained in Section 4. The prototype implementation of TbD–NNbR is described in Section 5. Section 6 discusses the evaluation of the proposed model and a conclusion is given in Section 7.

2. SQL injection attack categories

SQL injection attack is one of the most dangerous types of vulnerability attacks adopted by web hackers to compromise the security features of a critical application. In most of the vulnerability analysis, tautology, union queries, piggybacked queries, logically-incorrect queries, stored procedures, inferences and alternate encoding are the classifications of SQL injection attacks. A detailed description of SQL injection attacks along with examples are as follows [1,2,6].

2.1. Tautologies

In this attack, the hacker injects code into a conditional statement to evaluate it as true there by allowing the malicious user to bypass the user authentication or extract data from a database. For example, suppose that a malicious user inputs the SQL statement as SELECT * FROM books WHERE ID = '1' or '1'='1'–AND password = 'pass'; the comparison expression uses one or more relational operators to compare the operands and always generate true conditions. The targeted query may return all the rows in the books table. The possible signature for this type of SQL injection attack are the string terminator ', OR, =, LIKE and SELECT. It is a kind of attack in which hackers try to bypass authentication and extract data from the database.

2.2. Logically incorrect query/illegal queries

This type of attack is used to gather information about the back-end database of a web application through error messages of type mismatch or logical error, while the query gets rejected. For example, the injected query on the given URL can be in the format: http://www.elearning/mct/? id_user = '123'. The debugging information shown in the rejected query will reveal the database table information which can later be utilized to conduct further attacks.

2.3. Union query

These types of queries trick the database into returning the results from database tables which are different from what was intended. For example, an injected query can be of the format: SELECT * FROM users WHERE userid = 22 UNION SELECT item, results FROM reports. Here the attacker joins injected queries to a safe, legal query with a word UNION and gets details from different tables than the intended ones. Attackers mainly use this technique to bypass the authentication and extract data.

2.4. Piggybacked queries

The attacker tries to inject additional queries along with the original queries, which are said to 'piggyback' onto the original query. Hence the database gets multiple queries for execution. For example, SELECT Login ID FROM users ID WHERE login ID = 'john' and password = ''; DROP TABLE users- AND ID = 2345. After performing the first query, the database encounters the query delimiters (;) and executes the second query.

2.5. Alternate encodings

These types of attacks use the char () function and ASCII /Hexadecimal encoding. For example: SELECT accounts FROM users WHERE login = "" AND pin = 0; exec (char (0 × 73687574646j776e)). Hackers use this technique to avoid being identified by secure defensive coding and automated prevention mechanisms by modifying the query using the alternate encoding such as ASCII or hexadecimal coding practices.

2.6. Stored procedure

In these attacks, hackers aim to perform privilege escalation, denial of service and remote command execution using stored procedures through the user interface to back-end servers. For example, UPDATE users SET password = 'Nicky' WHERE id = '2' UNION SHUTDOWN;–. The hackers use a shutdown command with which the back-end database will be shutdown.

2.7. Inference attack

Here, the hackers aim to identify the injectable parameters and extract data from databases. Blind SQL injection attack and Timing SQL injection attack are the two categories of Inference attacks. For example the injected queries in the Timing SQL injection category can be in the following format: SELECT name, password FROM user WHERE id = 12; IF (LEN (SELECT TOP 1 column_name from testDB.information_schema.columns where table_name = 'user'),4) WAITFOR DELAY '00:00:10'—in this example the hacker tries to work by understanding the behavior of the back-end database by injecting an always true statement and along with a "WAIT FOR" keyword.

3. Related works

SQL injection has been an issue for many years, and several tools and strategies are developed to tackle this situation. Still, the risk rate of SQL injection is increasing exponentially in most of the online applications as there are fully automated injection tools available to talented hackers [7,8]. In a susceptible application, an SQL injection attack uses crooked input that changes the SQL query and establishes an illegal connection to the database. There are different classes of threats occurring through the security holes. Researchers Calvi and Vigan, in 2016 proposed some automated approaches of testing the security of the web application against the constant attack [9]. Zhu, Jun, et al. recommend secure programming and interactive static analysis as one of the optimal choices to handle some of the major security threats occurring in a web application [10]. An appropriate survey should be performed to identify the vulnerabilities, exploitations and countermeasures to the injection attacks through these vulnerabilities. Researchers Johari and Pankaj, in 2012 conducted a detailed study on Injection vulnerabilities, and strategies for countermeasures [1]. Researcher Aleroud and Zhou in 2017 handles emerging attack types, targeted environments, countermeasures and zero-day attacks as per the priority and requirement of the applications [11]. Also, the taxonomy of attacks is surveyed and analyzed [11]. Business organizations protect their sensitive data and block all possible vulnerable points from exploitation. As analyzed and documented by Skrupsky and Bisht et al. in 2013, the website security is improved by implementing appropriate web vulnerability scanners, and penetration testers [12]. In 2010, Doupe and Vigna proposed an implementation strategy to minimize the consequences on par with the severity of the vulnerability [13]. During security testing phase, documented by Lebeau and Franck et al. in 2013, the vulnerability