Third International Conference on NexGen Technologies
(IEI, Chandigarh) Institution of Engineers, India, Chandigarh
10th-11th August 2019 www.conferenceworld.in
Conference World
ISBN : 978-81-941721-2-3

Review on prevention& detection of compounded SQL injection attacks (SQL + XSS)

Kiran Prakash[1], Sunil Ghildiyal[2]

[1,2]*Department of CSE, Uttaranchal University, Dehradun, India*

ARTICLE INFO

Article History:
Received 00 Aug 00
Received in revised form
00 Aug00
Accepted 00 Aug 00

Keywords:
SQLI, XSS, Compound
SQL injection Attacks,
Taint Analysis

ABSTRACT

As per OWASP's (Open Web Application Security Project) list for web-basedapplication exploitation, SQL injection is one of the most widespread and potent vulnerabilities. SQL injection is an attack vector that is directed at the database layer of a web application and service to exploit a security vulnerability. SQL injection attacks take place in web applications during their interaction with the databases for various transactions. Most of modern-day web applications have critical, confidential and large amount of information stored in their databases. A single intrusion into these web applications can compromise the entire web applications. While basic SQL injection attacks have been analysed in detail, but there are multiple SQL injection attacks which need to be analysed for identifying better prevention, detection techniques. In this research work, review of the advanced SQL injection attack such as fast flux SQL injection, Compounded SQL injection and deep blind SQL injection has been done and then a detailed analysis regarding compound SQL injection attack consisting of SQL injection + XSS along with detection and prevention methods has been done.

Conference World
ISBN : 978-81-941721-2-3

1. Introduction

Today advanced threat actors such as nation-states, organized cybercriminals represent the greatest information security threat to computer-based applications. Many organizations are not able to detect these threats due to their secret nature and their intentional slow approach. One of the most common vulnerabilities found in web application is SQL injection. SQL injection is a code input methodology that uses a security flaw present in the database section of the application.Major types of advanced SQL injection attacks are detailed below: -

a) Fast flux SQL injection Attack

Fast flux is a Domain Name Server (DNS) technique to hide vulnerable websites behind aconstantly changing network of infected / compromised host. Principle characteristic of the fast flux network is that the network regularly keeps on changing its domains, IP addresses, and nameservers. These regular changes hide the real nature of the network and make it significantly difficult for security professionals and researchers to identify, analyse and mitigate the propagation of the harmful code.

b) Compounded SQL injection Attack

Compounded SQL injectionattack is a deliberate mixture of two or more attacks which target applications, websites and causes more significant damage than the basic SQL injection attacks. Compounded SQL injection attacks have emerged due to rapid emergence of prevention and detection techniques for the different types of basic SQL injection attacks. To bypass the improving defensive techniques, the malicious attackers devised a technique called compounded SQL injection. Compounded SQL injection is obtained from the deliberate mixture of SQL injection and other common web application attacks like:

- SQL injection + DDoS attacks
- SQL injection + XSS

c) Deep blind SQL injection Attack

Generally, database administrators respond to SQL injection attacks by suppressing the display of error messages related to database server. However, this methodology does not address the

root problem, which is related to poor code development process. Hiding error messages is like making the web application secure by obscurity. This methodology does pose a hurdle for the attacker, as now the attacked must develop an understanding of the web application, the database schema, etc., through the injection process primarily. This type of attack methodology is called a blind SQL injection attack, because the attacker cannot take guidance from the detailed error messages from the database server or different sources of information about the web application.

The classical SQL injection attacks were easy to prevent, detect and several procedures, methodologies are available to mitigate SQL injections attacks. However, it is observed that tools & methods for tackling advanced SQL injections attacks are lacking. Timely detection of threat, mitigating the threats and resuming steady state operations are some of the critical phases involved. If appropriate care is not taken during the coding, it can lead to the vulnerability of advanced SQL injection Attacks. It is observed that the practices such as parameterized statements and stored procedures should be implemented from the start [1]. Critical consequences of SQL injection attacks are on the following characteristics: -authorization, authentication, confidentially, integrity, database fingerprinting [2].

These types of attacks were not present a few years ago. However, with advancement in the field of UI/UX design and various other technological changes such as the introduction to JSON, jQuery have resulted in new vulnerabilities. SQLinjection attack countermeasures techniques are divided into three main approaches: static, dynamic and hybrid approaches. Static approaches are preferred while development and testing phase of software's. Static approaches mitigate the probability of SQLinjection attacks at compile time. On the other hand, dynamic approaches are useful for analysis of dynamic SQL query, generated by web application.In hybrid approaches a combination of static and dynamic approachesis used [4].There are multiple parameters such as performance, overhead and simple implementation of provided solutions for countering SQLIA [5].

Of the different types of advanced SQL injection attacks, compound SQL injections comprising of mixture of SQL injection and XSS are complex attacks and their prevention, detection

becomes difficult as most of the websites usually use JavaScript. Malicious injection of the code within susceptible webapplications to deceive users and redirect them to unreliable websites is called cross-site scripting (XSS). XSS enables malicious code tobe injected into the client-side script of webpages viewed byother users [6].There are innumerable websites which are vulnerable to the XSS + SQL injection Attack. The negligence at the initial stage can lead to monetary losses at later stage. Aforementioned attacks are caused due to baduser input. Attacks on web applications caused due to unvalidated user input areobserved as being the most common and critical [10, 15].

2. Literature Review

In 2017, Haibin Hu et al. [7] discuss that among numerous web security issues, SQL injection is the most notable and dangerous. The paper analyses the characteristics and procedures of SQL injection, and illustrates the methodology for identifying SQL injection attack. The defence resistance and remedy model of SQL injection attack is established from the perspective of nonintrusive SQL injection attack and defence. They further discuss comprehensive improvement in the ability of resisting SQL attack from the perspective of server security and policy settings of database. They also establish the role played by defence remedy model of SQL injection attack established in this study in resisting attacks for cases in which the server has already been attacked by SQL injection.

In 2016, Singh JP et al. [8] discuss about different types of SQL injection attacks, concepts related to it and negligence at initial stage can lead to monetary losses at later stage. They analyse how the recent the advancement in the field of UI/UX design and various other technological changes such as the introduction to JSON, jQuery have resulted in advanced SQL injection flaws. Next, they list the advanced SQL injection Attacks and present the techniques for the prevention and detection of such attacks. Finally, the various detection and prevention mechanisms are compared.

In 2016, Lawal et al. [9] discuss that SQL injection attack is a common threat to web applications that use inadequate input sanitization leading compromise of target databases. Their study gives comprehensive review on SQL injection attack, detection and prevention techniques. They conclude by evaluating the different techniques to check effectiveness of each technique based on how many methods of attack it can detect and prevent.

In 2017, Aliero et al. [14] discuss that SQL injection vulnerability is the one of the most widely occurring web-based application flaw that can be executed by SQL injection attack to get connected to

confidential data, circumvent authentication process, and execute unauthorized data manipulation language. They explain that defensive coding is a simple and affordable way to tackle this type of attack. But there are alternate issues concerning implementation of defensive coding which makes the application in totality, more vulnerable and less flexible to attack. In their paper they provide detailed background of SQL injection Attack, categorized defensive coding in various sections, analysed existing technique that are related, evaluate pros and cons of such methodology, recommend these techniques as per the statistics related to number of attacks they were able to prevent and apprise individual category of approach based on its deployment specifications concerning their inheritance. In their research work they aim to provide programmers with common issues that need to be considered before choosing a technique. Further, they also endeavour to raise awareness of issues related to these techniques as many of these are not meant for the purpose of protection of SQL injection attack. Finally, they provide suggestions on developing good SQL injection protection tools.

3. Implementation of Dynamic Taint Analysis Algorithm

Taint analysis hasplayed a central role in computer security and recent work has exhibited a new application of this technique,concerned with detection of exploits on contemporary software [16]. Taint analysis could be divided into static taint analysisand dynamic taint analysis on the criteria of running state. Static analysis monitors tainted databy performing semantic and grammatical analysis of thesource code. Dynamic analysis identifies and tracks certaindata during program runtime[17].

Dynamic taint analysis marks the incoming data from the untrusted source as tainted. The flow of tainted data is tracked during the program execution. Whenever tainted data is used in a security-sensitive context, a proper action is taken. The execution may also be suspended depending upon the severity of the operation. Dynamic taint analysis tracks taint propagation when the program is in execution.Dynamic taint analysis can be executed in online or offline manner. Online analysis grasps taint propagation during the program run time, whereas offline analysis initially captures the trace of program execution and post that performs taint analysis by re-executing the program.[17]

As an implementation of Dynamic taint analysis algorithm, Ardilla, an automated tool for creating SQLI and XSS attacks in PHP/MySQL applications was developed by Kiezun et al. [13]. Ardilla is a white-box testing tool, thereby implying that it requires the application source code. Ardilla is developed for testing PHP based applications before deployment. Security vulnerabilities identified by Ardilla can be resolved before the software is exposed to the users.Ardilla identifies data coming in the application and ascertains whether tainted data is able to reach vulnerable sinks. An illustration of a vulnerable sink is the PHP MySQL query function, which executes a string argument as a MySQL statement. If a string

obtained from the tainted data is given to this function, then an attacker can potentially execute an SQL injection if the tainted string impacts the structure of the SQL query while it was supposed to only being used as data within the query. Similarly, passing tainted data into functions that output HTML can be reasons for causing XSS attacks.

Ardilla uses below three components to find various types of attack vectors that can compromise an application: -

a) Input generation
b) Taint propagation
c) Input mutation

These components are discussed below.

Ardilla has the ability to use any input generator. During each cycle of execution, this input generator tracks the program to record path constraints that capture the output of control-flow predicates. The input generator by itself and recursively generates new inputs by removing one of the observed constraints and providing solution for the modified constraint system. Each newly created input has the ability to pursue at least one more control-flow path.

Ardilla's taint propagation monitors the flow of tainted data through the database transactions. When tainted data is stored in the database, the information related to the taint is also stored with it. When the data is later obtained from the database, it is identified with the stored taint. Thus, it can be seen that data that was tainted while storing is tainted on retrieval. This accuracy makes Ardilla able to accurately identify second order (persistent) XSS attacks.

Ardilla creates accurate attack vectors by methodologically mutating inputs that propagate taints to vulnerable sinks, using a library of strings that can execute SQLI and XSS attacks. This step is mandatory as not every flow of tainted data to a sensitive sink may be a vulnerability since the data may pass through routines that verify or sanitize it. Ardilla then analyses the difference between the parse trees of application outputs (SQL and HTML) to check whether the attack may subvert the behaviour of a database or a web browser, respectively. This step enables Ardilla to reduce the number of false warnings and to precisely identify real vulnerabilities.

The process of attack-creation generates a set of accurate inputs, runs the program under test with each input, and dynamically monitors if data flows from an input to a vulnerable sink (e.g., a function such as MySQL query or echo), including any data-flows that leads to a transaction with a database. If an input

reaches a sensitive sink, the technique modifies the input by deploying a library of attack patterns, with the intention to pass harmful data through the program. It is to be noted that anyvisitor that browses an SQL-injected web server is at the risk of being infected [18]and hence it is very critical to identify such vulnerabilities and resolve them.

Tools can mitigate all types of SQLinjection attacks except stored procedure and normally, the most common attack will threaten the database system is the login system [19, 20].

4. Conclusion

An idea of different types of SQL injection attacks and the more critical among them i.e.compounded SQL injection attack is discussed explaining the need to mitigate such type of attacks. Dynamic taint analysis is the commonly used technique for detecting compounded SQL injection attack. This paper serves as the gateway to all the aspirants who desire to learn about compounded SQL injection attacks for wider applications in cyber security domain. It serves as a reference paper to those who intend to work on the research areas of compounded SQL injection attack, taint analysis, etc. Algorithms similar to Ardilla implementation will serve as method to understand the backend of the tools needed to mitigate compound SQL injection attacks. Anyone wishing to design a tool for mitigating compound SQL injection attacks may refer to this paper to understand the algorithm that operates behind most of the tools. With the ever-increasing popularity of web applications, and the prevalent security vulnerabilities that pose a threat to them, future researchers may like to discuss the input generation mechanism in Ardilla. Ardilla can only generate attacks for a sensitive sink if the input generator creates an input that reaches the sink. However, effective input generation for PHP is challenging, complicated by its evolving language features and run time model (executing a PHP program frequently creates an HTML page with forms and links that solicit user inputs to run code in more files). This leads to a serious hindrance in achieving high coverage in web applications with low line coverage.Ardilla uses the input generator as a black box and any improvement in input generation is likely to improve Ardilla's effectiveness. In this regard, future researchers can optimize Ardilla algorithm to include provision for user input so that the tool can achieve high coverage of web applications.

References

1. Gopali. 2018. 'Protecting Web Applications from SQL injection Attacks - Guidelines for Programmers'.
2. Alwan; Younis.2017. 'Detection and Prevention of SQL injection Attack: A Survey'. International Journal of Computer Science and Mobile Computing.
3. OWASP Top 10 Application Security Risks - 2017 (https://www.owasp.org/index.php/Top_10-2017_Top_10)
4. Sadotra; Sharma.2017. 'SQL injection Impact on Web Server and Their Risk Mitigation Policy Implementation Techniques: An Ultimate solution to Prevent Computer Network from Illegal Intrusion'. International Journal of Advanced Research in Computer Science.
5. VaseghiPanah, Khatoon Bayat, Asami, Shahmirzadi.2016 'SQL injection Attacks: A Systematic Review' International Journal of Computer Science and Information Security (IJCSIS), Vol. 14, No. 12
6. Ayeni, Sahalu, Adyanju. 2018. 'Detecting Cross-Site Scripting in Web Applications Using Fuzzy Inference System' Journal of Computer Networks and Communications Volume 2018, Article ID 8159548
7. Haibin Hu. 2017. 'Research on the Technology of Detecting the SQL injection Attack and Non-Intrusive Prevention in WEB System', AIP Conference Proceedings.
8. Singh, JP.2016. 'Analysis of SQL injection Detection Techniques'.
9. Lawal;Bakar;Shakiru. 2016. 'Systematic Literature review on SQL injection attack' International Journal of Soft Computing 11(1): 26-35,2016.
10. Li Qian; ZhenyuanZhu; Jun Hu;Shuying Liu.2015. 'Research of SQL injection attack and prevention technology' International Conference on Estimation, Detection and Information Fusion (ICEDIF), Harbin, 2015, pp. 303-306.
11. Devi, Ruby & Venkatesan, R &Koteeswaran, Raghuraman. 2016. 'A study on SQL injectiontechniques. International Journal of Pharmacy and Technology. 8. 22405-22415.
12. Dalai; Jena.2017. 'Neutralizing SQL injection Attack using Server-Side Code Modification in Web Applications'. Hindawi Security and Communication Networks Volume 2017.
13. Kiezun;Guo;Jayaraman; Ernst. 'Automatic creation of SQL injection and cross-site scripting attacks' 31st International Conference on Software Engineering Pages 199-209
14. Aliero;Ardo; Ghani; Atiku.2016. 'Classification of SQL injection Detection and Prevention Measure'. IOSR Journal of Engineering (p): 2278-8719 Vol. 06, Issue 02 (February. 2016), ||V1|| PP 06-17
15. TejasSaoji. 2017. 'Implementing Dynamic Coarse- & Fine-Grained Taint Analysis for Rhino JavaScript'

16. W. Xu, S. Bhatkar, and R. Sekar, 2006. ‘‘Taint-enhanced policy enforcement: A practical approach to defeat a wide range of attacks,’’ in Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15, ser. USENIX-SS’06.Berkeley, CA, USA: USENIX Association.
17. Wang, Ma, Dou, Jian, Chen. 2018. ‘OFFDTAN: A New Approach of Offline Dynamic Taint Analysis for Binaries’ Hindawi Security and Communication Networks Volume 2018, Article ID 7693861
18. Shin, Myers, Gupta. ‘A Case Study on Asprox Infection Dynamics’. U. Flegel and D. Bruschi (Eds.): DIMVA, LNCS 5587, pp. 1–20.
19. Nischay, Hareesha, Rajesh, 2018. ‘Prevention and Detection of SQL injection of Attacks’. International Journal of Innovative Research in Science, Engineering and Technology Volume 7, Special Issue 6, May 2018
20. Yunus, Brohan, Nawi, Surin, Najib, Liang, 2018. ‘Review of SQL injection: Problems and Prevention’.International Journal on Informatics Visualization VOL 2 (2018) NO 3 - 2