DESIGN AND ANALYSIS OF SEPTIC PROTOCOL FOR IDENTIFYING SQL INJECTION ATTACKS

CHEKURI SRIDIVYA NAGA DURGA [#1], V.SARALA [#2]

[#1] MCA Student, Master of Computer Applications,

D.N.R. College, P.G.Courses & Research Center, Bhimavaram, AP, India.

[#2] Assistant Professor, Master of Computer Applications,

D.N.R. College, P.G.Courses & Research Center, Bhimavaram, AP, India.

ABSTRACT

Now a days almost all the enterprises try to store their valuable data using back end storage. These databases often integrated with vulnerable applications, such as front end web pages for performing the tasks, which allow injection attacks to be performed. In order to identify such attacks it is very difficult because the system cannot find the difference between normal query and sql attack in easy way. In order to identify such attacks, we propose SEPTIC, a mechanism for DBMS attack prevention, which can also assist on the identification of the vulnerabilities in the applications. The mechanism was implemented in MySQL and evaluated experimentally with various applications and alternative protection approaches. Our results show no false negatives and no false positives with SEPTIC, on the contrary to other solutions.

1.INTRODUCTION

Research over the years has mostly pinpointed developers' lack of security awareness in web development to sanitized input as the cause of SQLIA, and at such have gravitated towards code based sanitation for their proposed solutions to address SQLIA. Also, SQLIA vulnerability is a sequel to a design fallout of the well-intentioned free text processing of the SQL engine itself, and as a consequence both legacy and cloud deployments lacking sanitation becomes vulnerable. A search of SQL hall of shame [1] which reports the recent trends in data pilfering by SQLIA shows the prevalence of this form of attack and so the ability to secure back-end database from SQLIA in an era of big data remains a topical issue. The SQL language syntax closely resembles plain English [2], and the SQLIA keywords are also in plain text. Therefore, the SQLIA problem in a big data context is a plausible candidate for predictive analytics of a supervised learning model trained via both known historical attack signatures and safe web requests patterns.

The attack signatures at injection points will contain patterns of SQL tokens and symbols as SQLIA positive while valid web requests would take the form of expected data from the application. In this paper, we build a predictive analytics web application with quantities of learning data to train a classifier.

What is SQL Injection?

SQL injection is a code injection technique that might destroy your database.SQL injection is one of the most common web hacking techniques.SQL injection is the placement of malicious code in SQL statements, via web page input.In current days web applications are applications that can be accessed over the Internet by using any Web browser that runs on any operating system and architecture. They have become ubiquitous due to the convenience, flexibility, availability, and interoperability that they provide. Unfortunately, Web applications are also vulnerable to a variety of new security threats. SQL Injection Attacks (SQLIAs) are one of the most significant of such threats. SQLIAs have become increasingly frequent and pose very serious security risks because they can give attackers unrestricted access to the databases that underlie Web applications. The following are the limitations that takes place due to these SQLIAs.

This paper presents a method of SQL injection detection based on SEPTIC protocol and compares it with traditional detection methods. This method can output a fixed two-dimensional matrix without truncating data and effectively detects the SQL injection of web applications. Based on the irregular matching characteristics, it can identify new attacks and is harder to bypass. Here we try to use SEPTIC protocol which require can able to identify the SQL injections present in the database and can able to differentiate normal queries and injection queries. In this proposed application we try to construct key/signature generator and signature comparator. By using these two modules we can able to download the data under secure manner and can also recover the original file, if there is any attack occurred for the input data

2. LITERATURE REVIEW

Literature survey is the most important step in software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, ten next steps are to determine which operating system and language used for developing the tool. Once the programmers start building the tool, the programmers need lot of external support. This support obtained from senior programmers, from book or from websites. Before building the system the above consideration r taken into for developing the proposed system.

MOTIVATION

1) SQL Injection Detection for Web Applications Based on Elastic-Pooling CNN

Authors: XIN XIE and CHUNHUI REN

An enterprise's data can be one of its most important assets and often critical to the firm's development and survival. SQL injection attack is ranked first in the top ten risks to network applications by the Open Web Application Security Project (OWASP). Its harmfulness, universality, and severe situation are self-evident. This paper presents a method of SQL injection detection based on Elastic-Pooling CNN (EP-CNN) and compares it with traditional detection methods. This method can output a fixed two-dimensional matrix without truncating data and effectively detects the SQL injection of web applications. Based on the irregular matching characteristics, it can identify new attacks and is harder to bypass.

2) GMSA: Gathering Multiple Signatures Approach to Defend Against Code Injection Attacks

Authors: HUSSEIN ALNABULSI 1 , RAFIQUL ISLAM

Code injection attacks (CIAs) exploit security vulnerabilities and computer bugs that are caused by processing invalid codes. CIA is a problem which hackers attempt to introduce to any new method, their objective being to bypass the protection system. In this paper, we present a tool called GMSA, developed to detect a variety of CIAs, for example, cross-site scripting (XSS) attack, SQL injection attack, shell injection attack (command injection attack), and file inclusion attack. The latter consists of local file inclusion and remote file inclusion. Our empirical analysis reveals that compared with existing research, gathering multiple signatures approach (GMSA) executes a precision performance (accuracy of the proposed algorithm is 99.45%). The false positive rate (FPR) of GMSA is 0.59%, which is low compared with what other research has reported. The low FPR is the most important factor. Ideally, the defense algorithm should balance between the FPR and true positive rate (TPR) because with existing methodologies, security experts can defend against a broad range of CIAs with uncomplicated security software. Typical protection methods yield a high FPR. Our method results in high TPR while minimizing the resources needed to address the false positive. GMSA can detect four types of CIA. This is more comprehensive than other research techniques that are restricted to only two major types of CIA, namely, SQL injection and XSS attacks.

3) **Large Scale Text Classification Using Map Reduce and Naive Bayes Algorithm for Domain Specified Ontology Building**

Authors: J.Santosh

Internet that covers a large information gives an opportunity to obtain knowledge from it. Internet contains large unstructured and unorganized data such as text, video, and image. Problems arise on how to organize large amount of data and obtain a useful information from it. This information can be used as knowledge in the intelligent computer system. Ontology as one of knowledge representation covers a large

area topic. For constructing domain specified ontology, we use large text dataset on Internet and organize it into specified domain before ontology building process is done. We try to implement naive bayes text classifier using map reduce programming model in our research for organizing our large text dataset. In this experiment, we use animal and plant domain article in Wikipedia online encyclopedia as our dataset. Our proposed method can achieve highest accuracy with score about 98.8%. This experiment shows that our proposed method provides a robust system and good accuracy for classifying document into specified domain in preprocessing phase for domain specified ontology building

4) SQL Injection Detection for Web Applications Based on Elastic-Pooling CNN

Authors: Chunhui Ren and Yusheng Fu

An enterprise's data can be one of its most important assets and often critical to the firm's development and survival. SQL injection attack is ranked first in the top ten risks to network applications by the Open Web Application Security Project (OWASP). Its harmfulness, universality, and severe situation are self-evident. This paper presents a method of SQL injection detection based on Elastic-Pooling CNN (EP-CNN) and compares it with traditional detection methods. This method can output a fixed two-dimensional matrix without truncating data and effectively detects the SQL injection of web applications. Based on the irregular matching characteristics, it can identify new attacks and is harder to bypass.

3. EXISTING SYSTEM

In current days web applications are applications that can be accessed over the Internet by using any Web browser that runs on any operating system and architecture. They have become ubiquitous due to the convenience, flexibility, availability, and interoperability that they provide. Unfortunately, Web applications are also vulnerable to a variety of new security threats. SQL Injection Attacks (SQLIAs) are one of the most significant of such threats. SQLIAs have become increasingly frequent and pose very serious security risks because they can give attackers unrestricted access to the databases that underlie Web applications. The following are the limitations that take place due to these SQLIAs.

LIMITATIONS OF EXISTING SYSTEM

1) In the existing days there is no automated approach for dynamic detection and prevention of SQLIAs.

2) There is no concept to identify the difference between “Trusted “Strings and Injected Strings.

3) There is no mechanism which can identify the injection of some special operators or keywords during data insertion in a dynamic manner.

4) All the existing schemes failed to detect the SQL injections for web applications in accurate manner.

4. PROPOSED SYSTEM

This paper presents a method of SQL injection detection based on SEPTIC protocol and compares it with traditional detection methods. This method can output a fixed two-dimensional matrix without truncating data and effectively detects the SQL injection of web applications. Based on the irregular matching characteristics, it can identify new attacks and is harder to bypass. Here we try to use SEPTIC protocol which require can able to identify the SQL injections present in the database and can able to differentiate normal queries and injection queries. In this proposed application we try to construct key/signature generator and signature comparator. By using these two modules we can able to download the data under secure manner and can also recover the original file, if there is any attack occurred for the input data.

ADVANTAGES OF PROPOSED SYSTEM

1. All the attackers information is recorded in a log file
2. If there is any malicious query injected into the database, our SEPTIC protocol will trace that and identify easily.
3. The SEPTIC protocol can easily identify the intruders who try to create attack on sensitive information

The SEPTIC protocol can generate no false negative data because the false data can be easily recovered into original data.

5. SYSTEM MODULES

Implementation is the stage where the theoretical design is converted into programmatically manner. In this stage we will divide the application into a number of modules and then coded for deployment. The front end of the application takes JSP, HTML and Java Beans and as a Back-End Data base we took My SQL data base. The application is divided mainly into following 5 modules. They are as follows:

1) Owner Module

2) User Module

3) Application Server Module

4) Signature Generator Module

5) Signature Comparator Module

5.1 OWNER MODULE

In this module, the data owner uploads their data in the cloud server. For the security purpose the data owner encrypts the file and the index name and then store in the cloud. The data encryptor can have capable deleting of a specific file. And also he can view the transactions based on the files he uploaded to cloud

5.2 USER MODULE

In this module, user logs in by using his/her user name and password. After Login user do some operations like View your profile ,Request secret key from application server and view Response, Search data by its keyword and view all details and take secret key automatically if permission is given, Download the file by verifying signature. If signature is wrong then don't download

5.3 APPLICATION SERVER MODULE

The Application server manages a cloud to provide data storage service do some operations like View all Data Owners and authorize View all end users and authorize, View all data contents with rank and ratings with digital signature, View all data contents with rank and ratings without digital signature, View users search transactions, View all SQL Injection Intruders with date and time and IP Address, View all docs rank in chart, View all intruders and give link to view in chart(No. Of Attacked Documents name).

5.4 SIGNATURE GENERATOR MODULE

The signature Generator is the one who generates the generate Digital signature and do following operations like Login, View all owner documents and give option to generate Digital signature, View all data contents with rank and give option to generate Secret key using RSA

5.5 SIGNATURE COMPARATOR MODULE

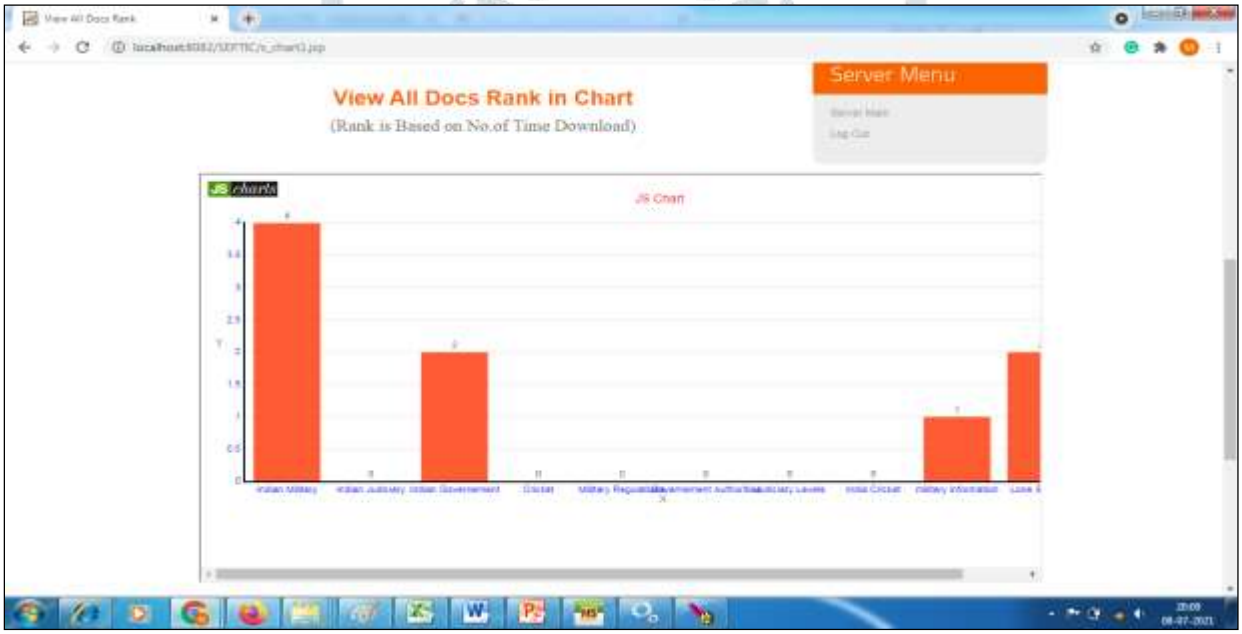
Here signature comparator is one who try to compare the signatures and then tell which documents are genuine and which are modified by the sql injection attacker.

6. OUTPUT RESULTS

1) ADMIN VIEW LIST OF OWNERS AND USERS

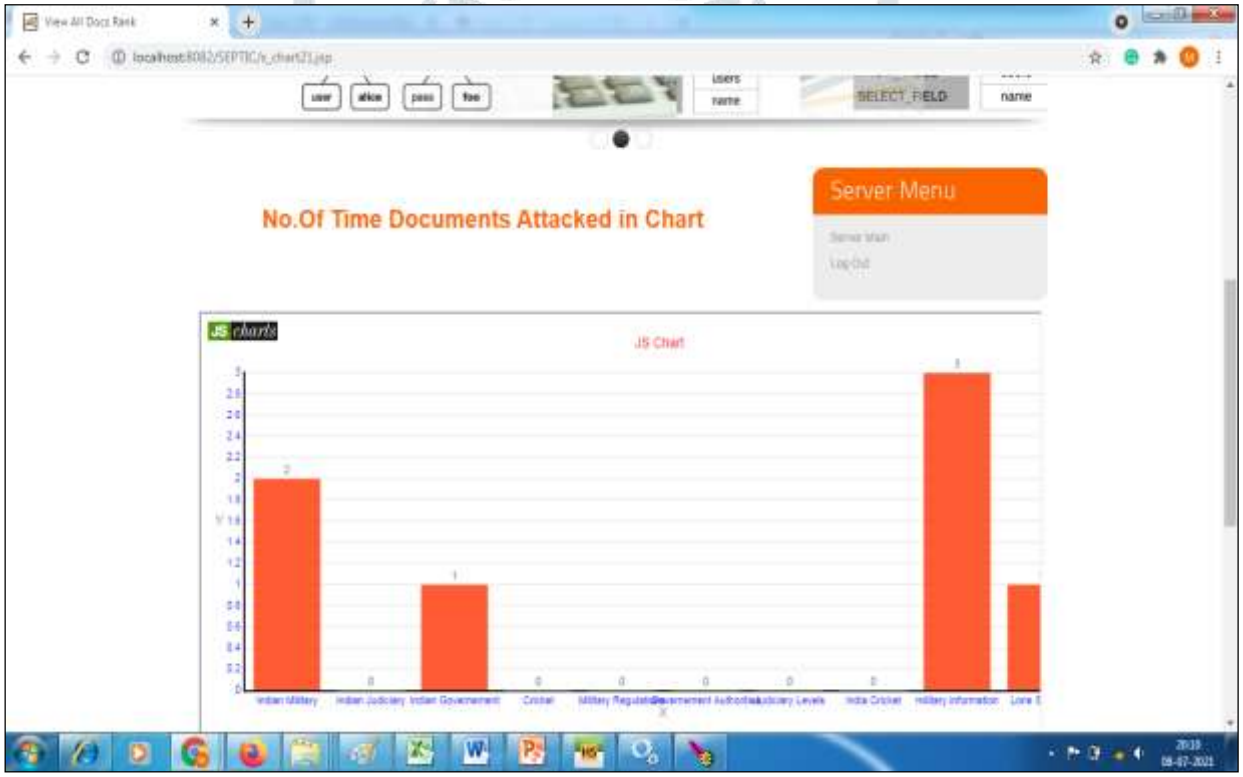
2) ADMIN VIEWS ALL SQL INJECTION USERS

3. ADMIN CAN VIEW THE LIST OF DOCUMENTS IN RANKED MANNER

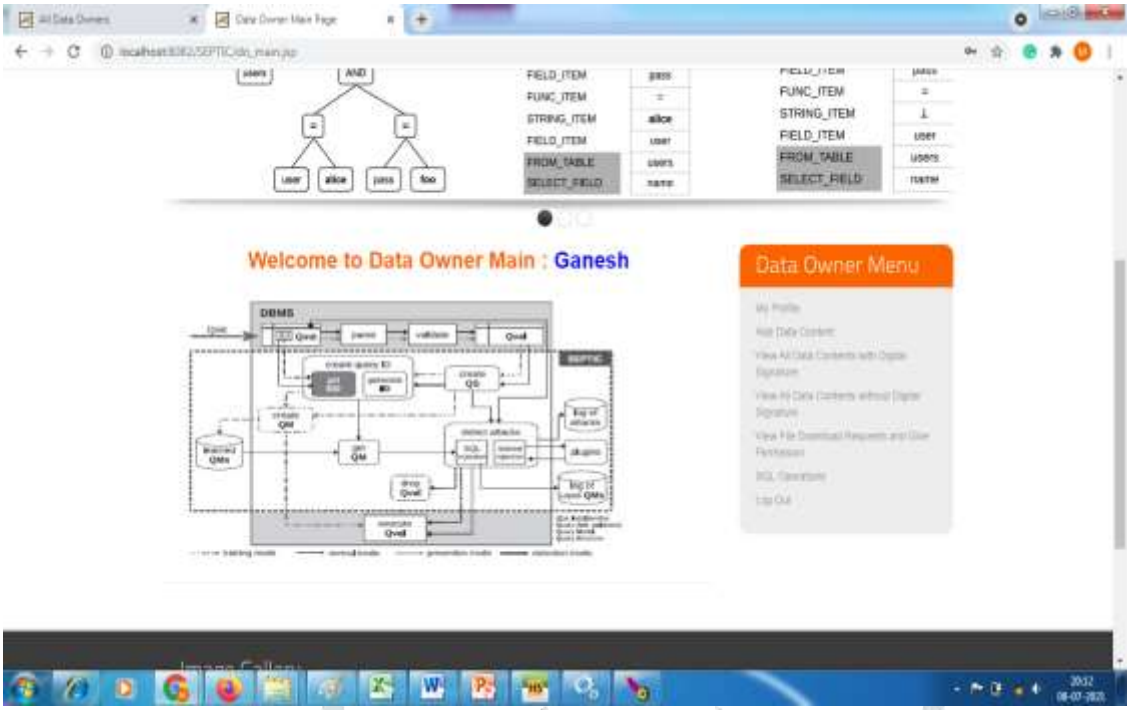


4) ADMIN CAN VIEW LIST OF ALL INTRUDERS

5) ADMIN VIEWS NUMBER OF DOCUMENTS IN RANKED MANNER

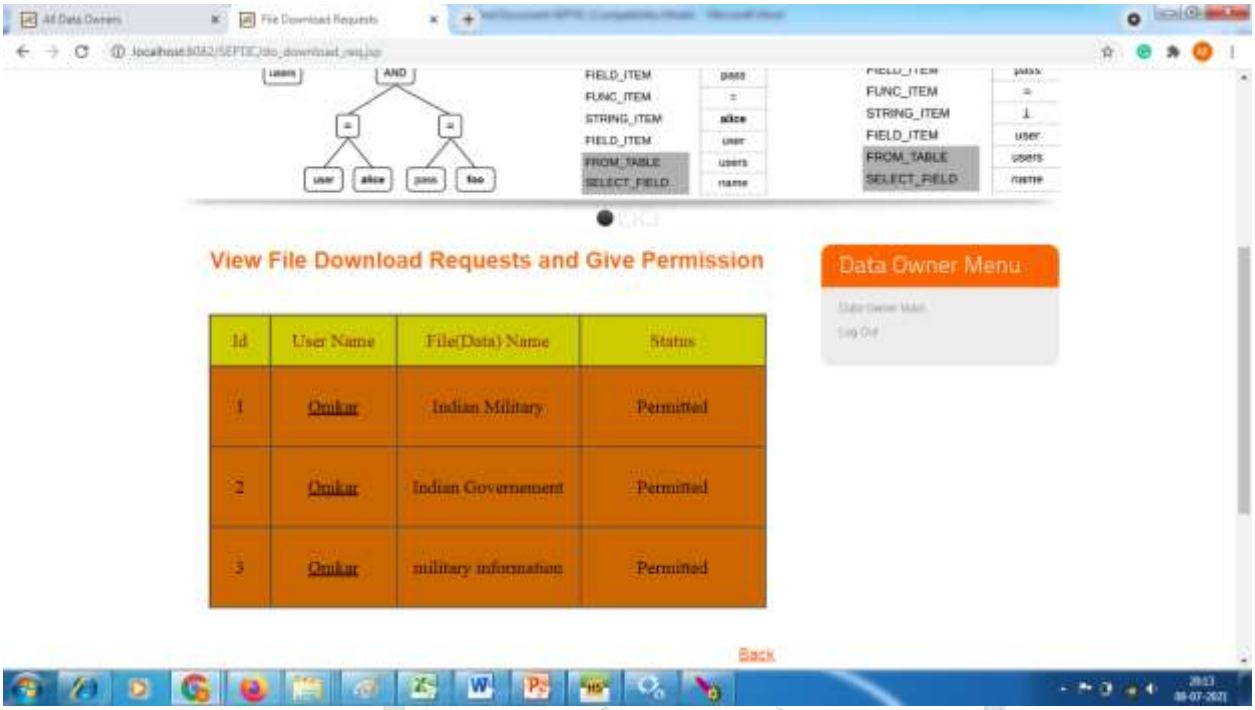


6) DATA OWNER LOGIN



7) ADMIN TRY TO ENTER SQL QUERY

8) ADMIN CAN VIEW THE FILE DOWNLOAD REQUEST AND GIVE PERMISSIONS



9) USER CAN VIEW HOME PAGE

10) USER CAN VIEW THE DOWNLOAD DATA

7. CONCLUSION

This project explored a new form of protection from attacks against web and business application databases. It presented the idea of catching attacks inside the DBMS, letting it protected from SQLI and stored injection attacks. Moreover, by putting protection inside the DBMS, we showed that it is possible to detect and block sophisticated attacks, including those related with the semantic mismatch problem. As a second idea, it presented a form of identifying vulnerabilities in application code, when attacks were detected. This paper also presented SEPTIC, a mechanism implemented inside MySQL. In order to do detection, SEPTIC resorts to a learning phase, and quarantine and aging processes that deal with models of queries, creating and managing them. The mechanism was experimented both with synthetic code with vulnerabilities inserted on purpose and with open-source PHP web applications, and other type of applications. This evaluation suggested that the mechanism could detect and block the attacks it was programmed to handle, performing better that all other tools in the literature and the WAF most used in practice, and can identify the vulnerabilities in code of applications, when the attacks attempted exploit them. The performance overhead evaluation of SEPTIC inside MySQL shows an impact of around 2.2%, suggesting that our approach can be used in real systems.

8. REFERENCES

[1]. injection attacks," *Adv. Comput. Syst. Secur.*, vol. 395, pp. 49–64, 2015.

[2] Akamai Technologie, Cambridge, MA, USA, "Q1 2016 state of the Internet/security report," Tech. Rep., vol. 3, no. 1, Jun. 2016.

[3] S. Bandhakavi, P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan, "CANDID: Preventing SQL injection attacks using dynamic candidate evaluations," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*," Oct. 2007, pp. 12–24.

[4] C. A. Bell, *Expert MySQL*. New York, NY, USA: Apress, 2007.

[5] S. W. Boyd and A. D. Keromytis, "SQLrand: Preventing SQL injection attacks," in *Proc. 2nd Appl. Cryptography Netw. Secur. Conf.*, 2004, pp. 292–302.

[6] G. T. Buehrer, B. W. Weide, and P. Sivilotti, "Using parse tree validation to prevent SQL injection attacks," in *Proc. 5th Int. Workshop Softw. Eng.Middleware*, Sep. 2005, pp. 106–113.

[7] M. Ceccato, C. D. Nguyen, D. Appelt, and L. C. Briand, "SOFIA: An automated security oracle for black-box testing of SQL-injection vulnerabilities,"in *Proc. 31st IEEE/ACMInt. Conf. Autom. Softw. Eng.*, Sep. 2016,pp. 167–177.

[8] E. Cecchet, V. Udayabhanu, T. Wood, and P. Shenoy, "BenchLab: An open testbed for realistic benchmarking of web applications," in *Proc. 2nd USENIX Conf. Web Appl. Develop.*, 2011, pp. 37–48.

[9] J. Clarke, *SQL Injection Attacks and Defense*. Rockland, MA, USA: Syngress,2009.

[10] *Common Vulnerabilities and Exposures*, 2014. [Online]. Available: http://cve.mitre.org

[11] *SolidIT: DB-Engines Ranking*, Aug. 2015. [Online]. Available: http://db-engines.com/en/ranking\

[12] A. Douglen, "SQL smuggling or, the attack that wasn't there," COMSEC Consulting, Inf. Secur., London, U.K., Tech. Rep., 2007.

[13] M. Dowd, J. Mcdonald, and J. Schuh, *Art of Software Security Assessment*.London, U.K.: Pearson Edu., 2006.

[14] J. Fonseca, N. Seixas, M.Vieira, and H. Madeira, "Analysis of field data on web security vulnerabilities," *Trans. Dependable Secure Comput.*, vol. 11,no. 2, pp. 89–100, Mar./Apr. 2014.

[15] *Gambas*, 2015. [Online]. Available: http://gambas.sourceforge.net/