A Detection Framework for SQL Injections and Cross Site Scripting

[1]Vamsi Mohan V, [2]Dr. Sandeep Malik
[1,2]Department of Computer Science,
[1,2]School of Engineering and Technology, Raffles University, Neemrana, India

***Abstract*: Designing secure web applications are most important aspect to avoid SQL injections and Cross Site Scripting (XSS) attacks. XSS vulnerabilities are classified into three types. i.e., Reflected XSS, Stored XSS and Dynamic XSS. From these types of XSS, DOM XSS is different from the two others. There are many researches and detection methods proposed for Reflected XSS and Stored XSS. However, it is not suitable for Dynamic XSS. Due to increase of web applications, the threats are getting increased. XSS often included in OWASP top-10 list from the last decade and hence an appropriate XSS detection method is necessary. In this paper, we propose a detection framework for SQLI and XSS. We introduced Regression Neural Network (RNN). It provides accurate and quick solution to regression, approximation, classification and fitting problems. RNN can be used in system identification of dynamic systems as well as control of dynamic systems. Here we are integrating multi-objective optimization which involves integration of objective formulation from dragon fly optimization (DA) and Genetic algorithm (G A). Integrating of optimization algorithm, crossover and mutation is used instead of alignment process of DA.**

***Index Terms*: SQL Injections, Cross Site Scripting, Regression Neural Network, Static Analysis, Dynamic Analysis.**

I. INTRODUCTION

SQL injection attacks have been known for decades now, but still hearing new stories where SQL injections were used for nefarious purposes. According to the register magazine, "SQL injection played a major role in a hacking incident during the 2016 US presidential election". FBI identified a Structured Query Language (SQL) injection (SQLi) vulnerability, used SQLmap to target the state's Board of Election website for vulnerabilities using Acunetix. A UK based telecommunication company suffered a data breach in the year 2015 due to SQL injections.

Cross Site Scripting is one in the top-10 OWASP list. It will lead to spread vulnerability and cause serious privacy issues and even infected with worms. DOM-XSS detection is classified into three categories. i.e., black box fuzzing, static analysis and dynamic analysis. Usually, black box fuzzing and static analysis suffer from high false positive and negative rates. Dynamic analysis is a complex and costlier to analyze even though it can obtain more accurate and efficient results. According to 2017 Web Application Attack Statistics, cross-site scripting attacks are used in 31% of all web attacks. The next most common cyber attacking technique, SQL injection, is responsible for just over 20% of web attacks.

In this paper, we have narrated our analysis into different sections. The first section discusses about the relevant work done in this area. Other sections describe proposed approach of the framework for detecting SQLI and XSS, Solution prototype of proposed system for SQLI and XSS detection. The last section describes the conclusion.

II. RELATED WORK

Because of the advancement of JavaScript, most of the validations and functions moved to the client side. These client-side scripts become more popular and powerful after introduction of HTML5. DOM-XSS is client-side security vulnerability, which is known as third XSS. It triggers XSS by the DOM parsing of the browsers. It doesn't involve in the analytical response of the server. Compare to the server-side applications, web client applications are facing greater challenges in the static and dynamic security testing.

Administrators can control the server-side code. However, the client-side code will run at the client side in their machines. According to K.Z.Snow, and et.al., (2016) compared with C# and Java commonly used by the server, JavaScript code often regards string data as executable code by eval or other API. Also, most of the modern frameworks, JavaScript use third party script code, which is hidden and difficult to control.

Hydra et. al. (2014), analyzed XSS attacks and found 0.9% DOM-XSS only till the year 2013. Vogt et. al., (2007) proposed a static analysis and dynamic information flow tracing method to reduce the risk of XSS. In their proposed method, they focused on sensitive information like user cookies, potential information leakage, and not on tracking unreliable data. Ra is a plug in for Firefox browser for detecting DOM-XSS, which is useful for detecting black box fuzzing. SpiderMonkey Javascript is modified and introduced a tool called Dominator for detecting DOM-XSS based vulnerability tracking.

Saxena et.al., proposed in their "Systematic discovery of client-side validation vulnerabilities in rich web applications" a method of black box fuzzing. In their work, dynamic taint analysis is combined with automatic random Fuzzing technology. At the same time, the corresponding prototype tool FLAX was developed. Criscione (2013) proposed an automation tool based on black box to find Cross Site Scripting (XSS), and the tool used a web browser to test the vulnerabilities in the code.

Lekies (2015) et al. implemented a detection technique for DOM-XSS and verification method based on dynamic taint tracking by modifying in the Chromium browser. DEXTERJS is a testing platform implemented by Parameshwaran (2015).

It is a type of dynamic taint tracking method based on byte level to obtain and verify the potential dataflow of vulnerabilities caused by the Web page. It is quite simple and in order to patch DOM-XSS, Parameshwaran proposed a scheme (2015), where code is not needed to modify in the browser. There is another tool called DOMXSS Scanner (2016) to check web pages source code with DOM XSS sources.

Majority of contribution focus at the web applications is on defensive coding, vulnerability testing and preventive actions. Programmers has to write defensive coding in a particular manner. Mcclure and Kruger (2005) designed SQL-DOM method, a set of classes strongly related to the database schema to generate safe SQL statements. This method is a complex activity and time consuming. Whenever there is a need of changing the database schema, there is a need to regeneration and recompilation of the classes.

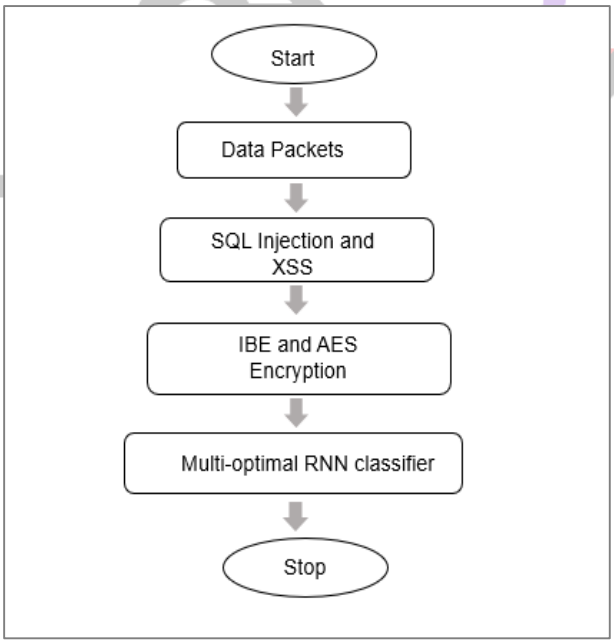
Thomas et al. (2009) and Bisht et al. (2010c) proposed methods to generate prepared statements in the application code automatically. whenever the source code is modified, the prepared statements need to be regenerated. Vulnerability testing-based approaches rely on testing web applications to discover injection hotspots to fix them before the production release. Benedikt et al. (2002) designed VeriWeb and Jovanovic et al. (2006) developed Pixy to automatically test and discover the SQL injection vulnerabilities in the code. Shin et al. (2006), Wassermann et al. (2008), and Ruse et al. (2010) proposed auto matic generation of test cases. Bisht et al. (2010a) designed NoTamper, a black-box testing method for detecting server-side vulnerabilities.

Effectiveness of these methods and approaches is limited by their ability to discover all possible security vulnerability issues. Prevention based approaches consist of preparing a model of queries to detect anomalies at runtime. Halfond and Orso (2005) designed AMNESIA combining static analysis of source code and runtime monitoring tool for vulnerability checking. Buehrer et al. (2005) proposed SQLGuard which compares parse tree of queries before and after user inputs.

III. SOLUTION PROTOTYPE OF PROPOSED SYSTEM FOR SQLI AND XSS DETECTION

In order to detect the SQL injections and Cross Site Scripting vulnerabilities in the web application code, this paper simulates the process. Here we have chosen the Advanced Encryption Standard (AES), which is at least six time faster than triple DES. AES is an iterative process than Feistel cipher. It is based on 'substitution–permutation network'. It is a combination of a series of linked operations, for some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

We used IBE encryption (ID-based encryption, or identity-based encryption), which is a primitive of ID-based cryptography technique. It is a public-key encryption in which the public key of a user is some unique information about the identity of the user (e.g. user's email address or date of birth). This means that a sender, who has access to the public parameters of the system can encrypt a message using receiver's identity as a key. The decryption key obtained by the receiver from a central authority, which needs to be trusted as it generates secret keys for every user.



In the process of detection SQL injections and Cross Site Scripting, we collected the data packets, which has attacked by sql injections and cross site scripting. We encrypted the text using IBE (ID-based encryption, or identity-based encryption) and AES techniques and classified through the multi-optimal RNN (Regression Neural Network)

IV. PROPOSED APPROACH OF THE FRAMEWORK FOR DETECTING SQLIA AND XSS

1. *Pseudo Code for AES*

```
state=M
AddRound Key(state, & r[0])
for i=1 step 1 to 9
SubBytes (state)
ShiftRows (state)
MixColumns (state)
AddRound Key (state, & r[i*4])
endfor
SubBytes(state)
ShiftRows(state)
AddRound Key(state, & r[40])
```

2. *Pseudo Code for IBE*

```
Input: Message m, receiver's public key QB
Output: U, C,tag
Set random u Zp
Compute U= u.G
Compute S( xs, ys) = u.QB
Generate (kENC, kMAC) =KDF(xs)
Encrypt C= ENC(m, kENC)
Generate tag = HMAC(C, kMAC).
```

3. *Multiobjective optimization-based Regression Neural Network*

RNN provides accurate and quick solution to regression, approximation, classification and fitting problems. RNN can be used in system identification of dynamic systems as well as control of dynamic systems. Here we are integrating multiobjective optimization which involves integration of objective formulation from dragon fly optimization (DA) and Genetic algorithm (GA). Integrating of optimization algorithm, crossover and mutation is used instead of alignment process of DA.

Preliminary steps are,

1. Collect Data
2. Separate into Training and Test Sets
3. Define a Network Structure
4. Select a Learning Algorithm
5. Set Parameters, Values, Initialize Weights

a. Weight selection/ initialization carried out using Multiobjective optimization
b. DA with genetic operators are employed here

6. Transform Data to Networks Inputs
7. Start Training, and Determine and Revise Weights
8. Stop and Test
9. Implementation: Use the Network with New Cases

4. *Pseudo-codes of the DA-GA algorithm:*

```
Initialize the dragonflies population Xi (i = 1, 2, ..., n)
Initialize step vectors ΔXi (i = 1, 2, ..., n)Xi (i = 1, 2, ..., n)
while the end condition is not satisfied
Calculate the objective values of all dragonflies
Update the food source and enemy
Update w, s, a, c, f, and e
Calculate S, A, C, F, and E using above Equations
Update neighbouring radius
if a dragonfly has at least one neighbouring dragonfly
Update by Two point Crossover,
Calculate Mutation rate
else
Update the neighbour radius as new source
end if
Check and correct the chromosome based on the
```

boundaries of variables
end while

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a framework for detecting SQLIA and XSS based on AES and IBE encryptions. Further, it is optimized based on the regression neural networks. In addition, we implemented the prototype detection framework for SQLIA and XSS. During the detection process, our framework analyses pages which may cause XSS and obtains vulnerability traces. According to these traces, our method can generate attack vectors automatically, which is quite significant for developers and testers to detect and resolve the vulnerabilities in the code.

REFERENCES

[1] Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (h)ibe in the standard model. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 553–572. Springer (2010)

[2] Apon, D., Fan, X., Liu, F.H.: Compact identity-based encryption from lwe. IACR Cryptology ePrint Archive (2016)

[3] Brannigan, S., Smyth, N., Oder, T., Valencia, F., O'Sullivan, E., G¨uneysu, T., Regazzoni, F.: An investigation of sources of randomness within discrete Gaussian sampling. Cryptology ePrint Archive, Report 2017/298 (2017), http://eprint.iacr.org/2017/298

[4] C. Criscione, firing-range, Google, https://github.com/google/firingrange (2016).

[5] D. Balzarotti, M. Cova, V. Felmetsger, N. Jovanovic, E. Kirda, C. Kruegel and G. Vigna, "Saner: Composing Static and Dynamic Analysis to Validate Sanitization in Web Applications," In IEEE symposium on Security and Privacy, (2008).

[6] Google, Ra.2, Google, https://code.google.com/archive/p/ra2-dom-xssscanner/issues (2012).

[7] G. Wassermann and Z. Su, "Static detection of cross-site Scripting vulnerabilities," In Proceeding of the 30th International Conference on Software Engineering, (2008)May.

[8] I. Hydara, A. B. M. Sultan, H. Zulzalil, N. Admodisastro, Current state of research on cross-site scripting (xss) c a systematic literature review, Information & Software Technology 58 (2014) 170–186.

[9] I. Parameshwaran, E. Budianto, S. Shinde, H. Dang, A. Sadhu, P. Saxena, Dexterjs: robust testing platform for dom-based xss vulnerabilities, in: Joint Meeting, 2015, pp. 946–949.

[10] I. Parameshwaran, E. Budianto, S. Shinde, H. Dang, A. Sadhu, P. Saxena, Auto-patching dom-based xss at scale, in: Joint Meeting, 2015, pp. 272–283.

[11] K. Z. Snow, R. Rogowski, J.Werner, H. Koo, F. Monrose, M. Polychronakis, Return to the zombie gadgets: Undermining destructive code reads via code inference attacks, in: Security and Privacy (SP), 2016 IEEE Symposium on, IEEE, 2016, pp. 954–968.

[12] M. Sutton, A. Greene, P. Amini, Fuzzing: Brute force vulnerability discovery, 2007.

[13] P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Krgel, G. Vigna, Cross site scripting prevention with dynamic data tainting and static analysis., in: Network and Distributed System Security Symposium, NDSS 2007, San Diego, California, Usa, February - March, 2007.

[14] P. Saxena, S. Hanna, P. Poosankam, D. Song, Flax: Systematic discovery of client-side validation vulnerabilities in rich web applications., in: Network and Distributed System Security Symposium, NDSS 2010, San Diego, California, Usa, February - March, 2010.

[15] R. Gomez, DOMXSS Scanner, https://github.com/yaph/domxssscanner (2016).

[16] Ritu Gaur, Ravi Bhushan; "Protection against SQL Injection Attack on Web Applications Using AES and Stored Procedure", IJARCSSE, Vol 4, Issue 5, 2014

[17] S. Lekies, B. Stock, M. Johns, 25 million flows later - large-scale detection of dom-based xss, in: ACM SIGSAC Conference on Computer & Communications Security, 2013, pp. 1193–1204.

[18] V.B. Livshits and M. S. Lam, "Finding security errors in Java programs with static analysis," In proceedings of the 14th Usenix security symposium, (2005)August, pp. 271-286.