A new algorithm for detecting SQL injection attack in Web application

Ouarda Lounis, Salah Eddine Bouhouita Guermeche, Lalia Saoudi, Salah Eddine Benaicha
Computer Science Department
University of Mohamed Boudiaf of M'Sila
M'Sila, Algeria

***Abstract*—Nowadays, the security of applications and Web servers is a new trend that finds its need on the Web. The number of vulnerabilities identified in this type of applications is constantly increasing especially SQL injection attack. It is therefore necessary to regularly audit Web applications to verify the presence of exploitable vulnerabilities. Web vulnerability scanner WASAPY is one of the audit tool, it uses an algorithm which bases on a classification techniques of pages obtained by sending HTTP requests especially formatted. We propose in this paper a new algorithm which was built in a vision to improve rather to supplement the logic followed in modeling WASAPY tool. The tool was supplemented by a new class reflecting the legitimate appearance or referential, therefore, the detection mechanism was solidly built on a statistic in a fairly clear mathematical framework described by a simple geometric representation or interpretation.**

***Keywords*—SQL injection attack; scanner Web; Web Application; Web vulnerabilities; security**

I. INTRODUCTION

The Web server security is now a recurring problem. The number of vulnerabilities identified in this type of software is constantly increasing, as described in particular in the document "The OWASP Ten Most Critical Web Application Security Risks" [4]. They can be explained by several reasons: the increasing complexity of Web technologies, deadlines ever tennis marketing software, sometimes limited skills and lack of safety culture developers. As a result, many of these applications contain multiple vulnerabilities that can be exploited by hackers. These attacks can allow them, for example, to obtain confidential data (credit card numbers, passwords, etc..) that are manipulated by the application, or even alter or destroy some of these data. The complexity of the technologies used today (Java, JavaScript, PHP, Ruby, J2E, etc..) to create Web applications makes it particularly difficult 1) to prevent the introduction of vulnerabilities in these applications and 2) estimate or predict their presence. In addition, network security and installation firewall does not provide adequate protection against Web attacks as these applications are accessible to all. It is therefore necessary to regularly audit Web applications to verify the presence of exploitable vulnerabilities and this can be realized by vulnerabilities Web scanners.

There are two main classes of approaches adopted by most of the vulnerability Web scanners:

- Approach based on recognition of error messages in response pages,
- Approach based on studying similarity of pages returned by the server.

These two approaches will be explained in section III.

In our project, we realize an approach for the detection of SQL injection attacks in Web applications, based on sending HTTP requests and analyzing the responses of the latter. This approach is based on two techniques: technique of recognition of error messages in response pages, and the study of similarity of pages returned by the server.

The proposed algorithm is a modified and improved approach of Rim's Akrout [6] algorithm, it is carried out in a spirit of logical completeness expressing a desire to specify a new model completing the WASAPY model. It involves sending HTTP requests to the Web server. These requests are structured into three classes: class containing syntactically valid requests, class containing syntactically invalid requests and class containing random requests. The responses of each request will be recorded and compared to a fourth class that contains only pages of references to examine the similarity between them using the Levenshtein algorithm [5], and following tests, we determine what are the requests that have well used to exploit the SQL injection vulnerability.

The LS model provide a detection mechanism built on a solid statistics in a clear mathematical framework covering the whole crowd resulting data. Detection of security following two phases 1) detect if the page is secured or not, and 2) if the page is not secure, a search request injection is performed.

Our scanner is experienced using vulnerable Web sites we implemented to calculate the performance of the approach implemented, according to three parameters: the detection rate, false positive, false negative. Graphs are drawn to properly express the results of experiments realized. The used request were extracted from OWASP (SQL Cheat Sheet) project [4].

II. Related Work

A. using the recognition of error messages in response pages approache

W3af [1] was created by Andres Riancho in 2006 and is considered one of the most powerful scanners, it is written in Python. Its modular architecture allows users to import and easily modify the different modules that compose it. W3af sends three HTTP requests to test the presence of a vulnerability in a page, the three responses associated with these queries are then analyzed. If they contain SQL error messages, w3af informs the user that the application is vulnerable to SQL injection. In spite of its power, it has no additional mechanism that is implemented to verify if the vulnerability actually exists or not, that is to say, if it is actually exploitable, this is its major default.

Wapiti [2] is another example which follows the same principle. This tool developed in Python, is able to detect SQL injection, XSS injections, mishandling of files, LDAP injection and execution of operating system commands from a URL. To identify SQL injections, it sends two requests. Vulnerability is declared present if an error is identified in the response pages. The effectiveness of this approach is related to the completeness of the knowledge base regrouping the error messages.

B. using the analysis similarity of pages returned by the server approache

Skipfish [3] was developed by Google to detect vulnerabilities on Web servers. It proceeds in two steps. In a first step, he analyses the Web application and collect all pages that appear to be stable. The others are ignored. To detect whether a page is stable Skipfish sends 15 queries and several tests are performed on the page. The main shortcoming of this scanner is that the distance used for the study of similarity considers the frequency of words regardless of the order of words in a text. Ignore the word order can lead to ignore the semantics of a page and again can lead to misjudge if two pages are identical or not. For example, the following pages share the same vocabulary, but they correspond to a successful and failed authentication respectively:

Your are authenticated, you have-nots has Entered wrong login.

Your are not authenticated, you-have Entered a wrong login.

III. Explanation and Criticism of Approaches Adopted by Web Scanner

A. Approache based on recognition of error messages in response pages

To identify SQL injections, this approach sends requests of a particular format and look for specific patterns in responses such as error messages database. The basic idea is that the presence of an SQL error message in an HTML page response means that the corresponding request has not been verified by the Web application before being sent to the server database. Therefore, the fact that the request was sent unchanged to the SQL server reveals the presence of a vulnerability. Scanners such as w3af (SQLI module) andWapiti adopt such an approach.

The effectiveness of the approach by recognition of error messages is related to the completeness of the knowledge base regrouping the error messages that may result from the execution of queries submitted to the Web application. Generally, as in the case of w3af we consider mainly the error messages from the database. However, the error messages that are included in the HTML response pages do not necessarily come from the server database. The error message may also be generated by the application that can also reformulate the error message from the server, for example to make it understandable to the client. Moreover, even if the message is generated by the database server, the receipt of this message is not sufficient to say that SQL injection is possible. Indeed, this message means that for this particular query, entries have not been cleaned, but this does not support the conclusion in relation to other SQL requests, particularly those correspond to successful attacks.

B. Approache based on studying similarity of pages returned by the server.

The principle of this approach is to send various requests specific to the type of vulnerabilities and to study the similarity of the responses returned by the application using a textual distance. Based on the results obtained and well-defined criteria, we conclude on the existence or not of a vulnerability.

Concerning this approach, it is based on the assumption that the contents of rejection page is generally different from the contents of execution page. For this comparison, however, to be effective, it is important to ensure a wide coverage of different types of pages of rejection that could be generated by the application. This can be achieved by generating a large number of queries to enable the largest possible number of different pages of rejection. However, existing implementations of this approach, especially in Skipfish generate too few queries. Skipfish only uses 3 queries. If the answers correspond to different pages of rejection, he incorrectly concludes that the vulnerability is present leading to a false positive. Moreover, this approach, as in any classification problem, the choice of the distance is very important. That used in Skipfish do not take into account the order of words in a text. However, this order generally defines the semantics of the page. It is therefore important to consider the order of words in text.

These analyzes clearly show the need to develop new approaches to improve the effectiveness of vulnerability tools detection. The work presented in this paper adopt a new approach.

IV. Global Preview of Our Approch

Security is a concept which requires a certain reflection especially if we want to give it a definition which allows us to develop methods or techniques allowing to handle certain quantities which capture as faithfully as possible the intrinsic aspects of the security of the system in question.

Our study is interested in the security of the Web application, of which it is necessary to specify the necessary measures which can reflect the state of its safety.

To do this, it seems to us essential to have a model vision of the security of Web application generally.

To talk about safety is to call upon the legitimacy and thus security and legitimacy are two equivalent notion in the following optic:

To be secured is to allow only the legitimate attitude which can in this case of Web application, be seen as a natural navigation in the Web application such as designed. So, a navigation is defined as a sequence of legitimate Web pages which can be grouped within a single class called class Ref.

And therefore the legitimacy can be defined as a passage of a page of reference to another reference page, in this vision of things, a secure Web application generate only legitimate pages and thus it belongs in the reference class. All in all, there is a unique class being invoked. And thus legitimacy or security is modelled by the unity described by a unique class containing all the reference pages of the Web application.

Conversely, the illegitimate has a strong ability to generate non-legitimate pages, evidently they don't belong to the class of reference pages Ref, and so it proposes a new classes. This fact, the illegitimate which captures the state of a not secure Web application; is described by an explosion of the unique class in new classes.

These new classes are created according to a criterion merging three criteria simultaneously. These criteria stemming from the syntactic correction and the sematic of the SQL.

- Syntactic aspect is controlled by the SGBD which generates errors messages suited to the produced syntactic violation.
- Values of legitimate attributes generate pages slightly different from references, while those with illigimate values produce error messages generated by the SGBD.
- The semantic aspect is present with the insertion at the request of tautologies or antilogies.

The fission which reflects the non-secure of the Web application considered produces two new classes:

- Class Aleat: contain the pages returned by the random requests.

Example : Considering the URL following with the parameter: id_suj

http://localhost/forum/ajouterreponse.php?id_suj=254

here, the id-suj value is randomly chosen.

- Inval class: contains the pages returned by the invalid requests.

Example:

http://localhost/forum/ajouterreponse.php?id_suj=1union privilege_type FROM information_schema.user_privileges WHERE privilege_type = 'SUPER' AND 1=1

The definition of classes is based on two fundamental concepts:

- The inside-class coupling: the elements of single class should be very close one of the other one, what reflects the intuition "those who are alike flock".
- The between-classes decoupling: the elements of two different classes should be remote one of the other one, what reflects the intuition "the remoteness disadvantage the union".

The coupling and the decoupling are defined on the notion of distance. In fact, this distance capture the degree of similarity between elements. The similarity is defined mathematical as a function acting on two HTML pages and returns a real number in the interval [0,1]. Thus, the mathematical sign of the similarity is as follows:

Sim: C_1 x C_2 [0,1]
Sim: [P_1 ,P_2] s
Sim (P_1 ,P_2) = s
Sim: Similaritv

Fig. 1. Similarity between two pages

Many researches were deployed for the definition of functions of similarity between two strings. Their computation strategies differ according to their vision of the string:

- *Character by character:* The matching involves both a character of the first with the character of the same order but took the second page. Among the most famous is the Levenshtein distance
- *Line by Line:* The matching involves two lines each time, each taken from one of the pages, one of the most famous is the diff Linux.
- *Word by Word:* The matching is done by comparing a word of the first page with the corresponding word in order in the second page. Taking into account the order of the words in each page define two variants of this strategy:
  - With consideration of the scheduling of words.
  - Without consideration of their mutual scheduling.

In our case, we used Lenvenshtein algorithm [5] which will be described in the fifth section.

The vulnerability if it exists, it sounds from two points called injection points:

- URL: is identified by the presence of URL parameter.
- Formulary: is identified by text fields such as the login and the password.

V. THE WASAPY MODEL

A model recently proposed within the framework of a Phd thesis in the national institute of applied sciences of Toulouse (INSA Toulouse) [6].

A. Presentation of the model

a) The vision of safety:

The security of Web application is seen as the non-presence of illegitimate pages generated by a malicious behavior. Thus, the model articulates on the illegitimate to explore the security status of the Web application in question. This way of making influences considerably the modelling of the approach.

b) The classes of the model:

The model proposes two classes: the class Aleat of Web pages generated by random requests as well as the class Inval including Web pages turned by invalid requests. We notice well that the choice of the classes reflects the spirit of the approach since the two classes represent the malicious requests. In addition to these two classes, we found a class which presents a syntactically valid requests.

c) The principal of detection:

The detection principle is formed around the clustering technique implies a similarity threshold ε defined as follows:

- The ε algorithm

Set the threshold by the smallest distance between i) The longest distance between two responses in Aleat,

ii) The longest distance between two in the invalid responses.

- The clustering algorithm

Grouped together within the same cluster those queries which the pair wise distance is less than a threshold. [7]

- Detection Algorithm

If ∃ a Grap whose members are only valid requests
Then all these request have allowed to exploit the injection, so the site is not secure.

Fig. 2. Detection algorithm

B. Criticism of WASAPY model

a) The vision of security:

The vision was bounded on the illegitimate part of the queries, which deprived the completeness of a model that could be very beneficial. This gap has greatly influenced the accuracy of the proposed approach. Also, the model classes were limited to two classes reflecting malice only at the expense of the correction.

b) The classes of the model:

The class Aleat and the class Inval should respect the classification ethics which are:

- A coupling factor: as low as possible which expresses assembling similarity. This factor is well respected by tests conducted in the study of WASAPY model.
- A factor of Decoupling: most important factor that justifies the creation of new classes. However, an interesting observation draws our attention to the fact that the decoupling factor is so low that coincides with the coupling factor. This reflects negatively on the quality of classes offered, because of the similarity point of view, the two classes are only one.

c) The choice of threshold ε:

The choice of ε was motivated by the desire to minimize false positives products, and therefore, its design foundation was not mathematically or statistically very clear. The evaluation of the threshold ε was much guided by intuition that needed empirical experiments to acquire a certain degree of confidence.

d) Detection:

Detection is significantly related to the threshold ε, the design of the latter will have an impact, especially, the fact that it was designed with the intent to minimize false positives.

VI. LS MODEL

A. The vision of security

The vision of security is induced by the completeness of legitimate and illegitimate space. And thus, to be secure is to be closer to legitimate than illegitimate.

B. The model classes

a) The classe Ref

Reflects the legitimate website navigation, which is ultimately a set of legitimate pages called "reference pages" that are offered by the site during normal navigation. And so, a page that is not offered by a natural navigation of the website is recognized as a page raising from a malicious attempt to access unauthorized information and so this page is an attack signature that can reveal a vulnerability in the web site design. These pages are the result of some queries which represent a navigation parallel to that offered by the site navigation.

b) The class Aleat

Aleat is the class which queries are generated from words randomly selected from the list [a- zA -Z0 -9]. These queries generate pages rejection [6].

c) The class Inval

Inval is the class of SQL injection, syntactically invalid queries to the injection point. They are constructed so that the SQL Server that interprets these queries generates an error [6].

d) The class Val

Val is the class of queries that generates execution pages. For example, the couple login / pass which respectively have the values ' or'1 ' = '1 - abcd and generates the following SQL query :

SELECT * FROM users WHERE login=' 'or'1'='1--' and pass='abcd'

C. The principle of detection

The detection principle is formulated through two phases :

- Determine if the site is secure or not.
- If the site is not secure , find the maximum injection queries.

a) Phase one :

A secure page allows only legitimate consultation therefore it returned legitimate pages or their ones which are very closer. The neighborhood is defined with the distance which is a similarity function. This is a measure that quantifies the degree of similarity between two pages. So securing entails approximating pages returned by different queries from the three classes of LS.

However, if the page is not secure, then it has vulnerabilities that can be exploited by illegitimate requests, applied to injection points producing illegitimate pages that are significantly distant from legitimate pages. And therefore the none-securing induce removal of pages returned. The speed away of, depend from the classes of the queries.

The notion of proximity is quite relative, which should be quantified in order to allow a specific numerical language much more than descriptive one. The nature of the analysis of such studies provide a huge amount of data , and take into account all the information that its confine is a challenge that can be overcome with a good statistic . The latter should be defined in a way that to get the most information in the most compact possible form.

Descriptive statistics gives us a simple tool which is the arithmetic mean. It can be very effective if properly exploited, especially if it is engaged in a statistics of levels.

The model LS formalizes detection with a triangular vision focuses on the three classes in a space of four classes: Ref, Aleat , Inval and Val . The detection formula is described with highly descriptive geometric language. The following diagram describes geometrically the first phase of detection principle:

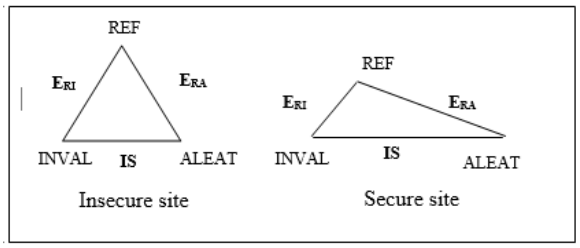

Fig. 3. Triangular representation of secure and non-secure page

$E_{RA:}$ expectation between the similarity between the reference pages and answers random queries.

E_{RI}: expectation between the similarity between the reference pages and answers invalid queries.

IS: Security Index.

The principle of securing is as follow:

If (triangle's base < ½ (sum (ERA , ERI)))
Then the page is not secure
Else it's secure.

Fig. 4. Principle of securing

The detection principle of securing a page is formulated on a strong enough bases borrowed from descriptive statistics. The measure (expectation) is simple but effective to capture the essential and relevant information for our study.

a) Phase Two: injection queries?

In the case where the detection phase determines that the page is not secure, it engages a set of queries to find those that can exploit SQL injection. The principle of detection of injection queries is formalized following the same spirit of reflection already expressed in the first phase detection.

Illegitimate applications of both classes: Aleat and Inval return pages that are clearly distant from legitimate pages of the class Ref. Knowing that the pages of the two classes : Aleat and Inval who share the same offense of illegitimacy making less distant from each other compared with the class Ref. We conclude that a non-secure page is sensitive in its reaction with illegitimate requests without paying much attention to their syntactic correctness.

Similarly, here too, the concept of proximity is formalized with a borrowed geometry in a triangular vision language represented as follows:

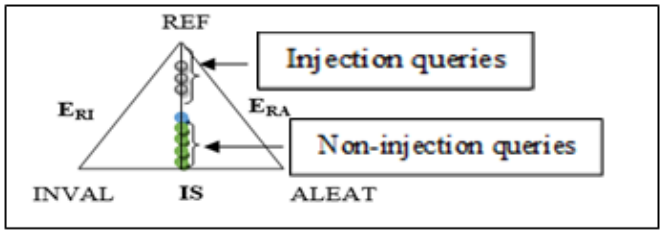

Fig. 5. Triangular representation of the vulnerability detection on non-secure page

D. The singularity of LS Model

The singular point of the model is: IS = 0 which means that the basis of reasoning [Inval, Aleat] disappeared. Therefore, it does not revolve on what to make a decision to secure, because the responses of invalid and random queries are merged into a single class (decoupling factor = 0, coupling factor = 0). From the triangular vision, we go at the linear vision, we don't need references, and the two classes Aleat and Inval are no longer discernible. The role of class Ref became neutral.

And therefore, the need for valid class arises. The geometric description of the model became a line whose ends are fused two points (Aleat , Inval) and the second is (Val) . If it retains its geometric shape of this line expresses the non-secure page and if it shrinks to a single point is to express the security and confirm our starting view that unity expresses security and bursting none securing. This singularity of the model represents the SQL injection applied to authentication forms (POST).

E. Algorithm global of detection

S: is the distance between two textual responses (Levenshtein algorithm).

ReqAleat, ReqInval, Ref, ReqVal: are respectively the random queries, invalid, reference pages and valid requests.

RepAleat, RepInval, RepRef, RepVal: are respectively random, invalid, reference pages and valid responses.

```
Integer i:= 0;
ReqAleat, ReqInval, Ref, ReqVal;
Read responss of each type of the queries: RepAleat,
RepInval, RepRef, RepVal;
Vector B:= Dis (RepAleat, RepInval); Vector A1:=
Dis (RepRef, RepInval;
Vector A2:= Dis (RepRef, RepAleat); Vector C1:=
Dis (RepVal, RepAleat);
Vector C2:= Dis (RepVal, RepInval); Vector D:= Dis
(RepVal, RepRef);
Calculus : IS = avg(B); Calculer : x:= avg(A1,A2);
If (x < IS) the page is secure End If
If (x > IS)
      Loop
        Calculate c:= avg(C1i,C2i);
        If (c > IS) the ith query of the valid queries
       has   successes to exploit un SQL injection.
        Else the query has not permit the exploitation
       of   the injection.
        i:= i+1;
      Until  i= NbrReqVal
End If
If (IS = 0)
      If (C1 =0 AND C2 =0)   the page is secure
      Else the page is not secure.
End If
```

Fig. 6. Global algorithm of detection of the LS model

```
integer DistanceOfLevenshtein(string chaine1, string
chaine2)

      // i and j iterate over  chaine1 et chaine2
      Integer  i, j, cost, lengthChaine1, lengthChaine2
      lengthChaine1 := lengthr(Chaine1)
      lengthChaine2 := length(Chaine2)
      // d is a table of  l engthChaine1 +1 rowws and
lengthChaine2 +1 columns
      Integer d[0..lengthChaine1, 0..lengthChaine2]

      for i from 0 to lengthChaine1
        d[i, 0] := i
      for j from 0 to lengthrChaine2
        d[0, j] := j

      for i from 1to lengthChaine1
        for j from 1to lengthChaine2
          if chaine1[i - 1] = chaine2[j - 1] then cost := 0
          else cost := 1
          d[i, j] := minimum(
                        d[i-1, j  ] + 1,     // supression
                        d[i,   j-1] + 1,     // insertion
                        d[i-1, j-1] + coût   // substitution
                      )

      return d[lengthrChaine1, lengthChaine2]
```

Fig. 7. Levenshtein algorithm

The first two cases (x> IS) and (x <IS) formalize the detection of SQL injection when the injection point is an URL (GET), and the third case, it is the SQL injection detection in authentication forms (POST).

VII. The Distance Used

To analyze the similarity between two HTML pages, we need a distance to evaluate the deference between two strings. The order of words in page has a great importance. Thus, to calculate the distance between two pages we have selected Levenshtein algorithm [5]. The principle of this algorithm is described in figure 7.

The final distance is calculate as follow, and the result will be between [0,1] :

$$Dis = \frac{DistanceOfLevenshtein(Chaine1, Chaine2)}{lengthChaine1 + lengthChaine2}$$

VIII. Results and Experiments

We conducted a series of experiments on six Web applications that we have developed, containing seven secure pages and other seven unsecured, in total, in order to illustrate the effectiveness of our algorithm.

The main objective of these experiments is to characterize the ability of the model to deal with SQL injection attack and specifically to evaluate its effectiveness for detection. Such efficiency is generally characterized by the evaluation of the detection rate, false negative and false positive rate.

Note:

- We say that two pages a and b are similar if the distance between them is closer to the 0, and they are not similar otherwise.
- All you need is a single valid query that successfully exploits the flaw to say that the page is vulnerable to a SQL injection attack.
- The Model contains the following cardinality:

Cardinality Inval: 10 queries

Cardinality Aleat: 10 queries

Cardinality Val : 42 queries

In what follows, we will present the curve of each case.

A. A non-secure Web page

a) URL non secured

For a non-secure web page (URL), we have the following curves:

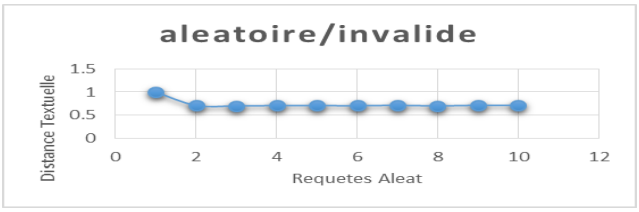

Fig. 8. Curve of the distance between invalid and random

Fig. 9. Curve of the distance between reference and random

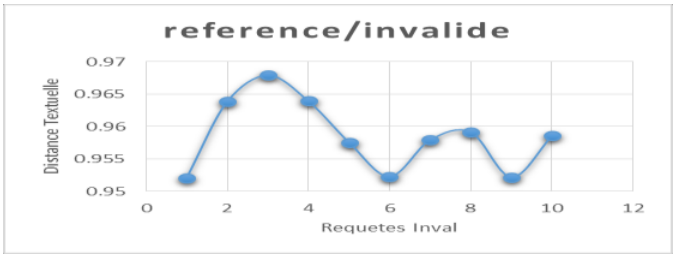

Fig. 10. Curve of the distance between reference and invalid

For a non-secure Web page (URL), we have the following geometric representation:

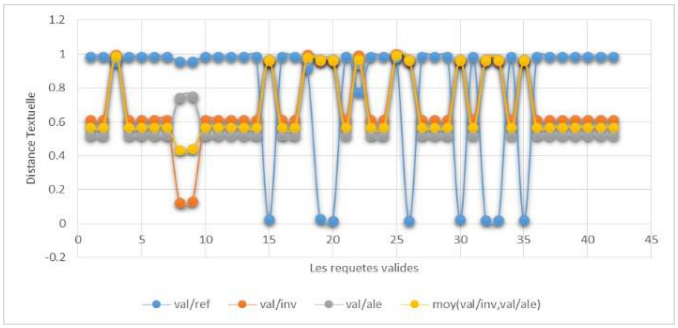

Fig. 11. Curve of the result of a non-secure page (URL)

Figure 12 shows the distance between the three classes (Ref, Inval, Aleat) described in the green square. Inside each circle, we have the green dot that expresses mathematical average written in Pink Square and the mathematical variance written in the yellow square.

The distance between the classes is very high which means that the decoupling between them is very strong, the variance of each class is very small, which means that the values of each class are identical.

a) *Unsecured formulary*

For a non-secure formulary, we have the following curves:

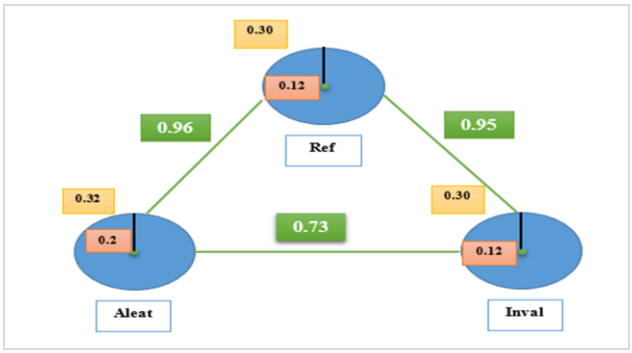

Fig. 12. Geometric representation (triangular) of a non-secure page (URL)

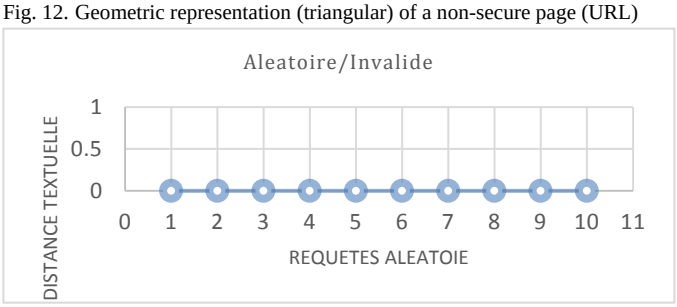

Fig. 13. Curve of the distance between random and invalide

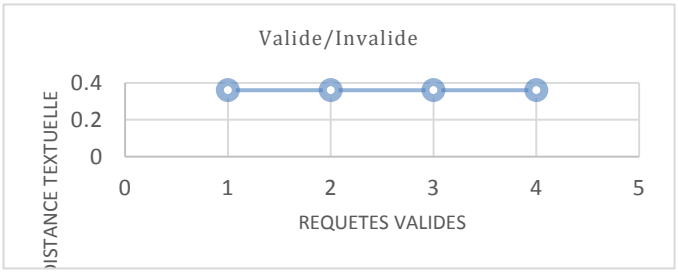

Fig. 14. Curve of the distance between valid and invalid

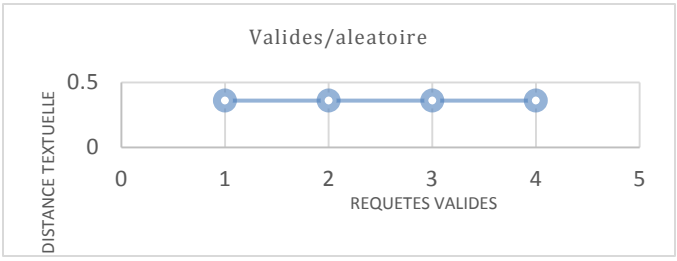

Fig. 15. Curve of the distance between valid and random

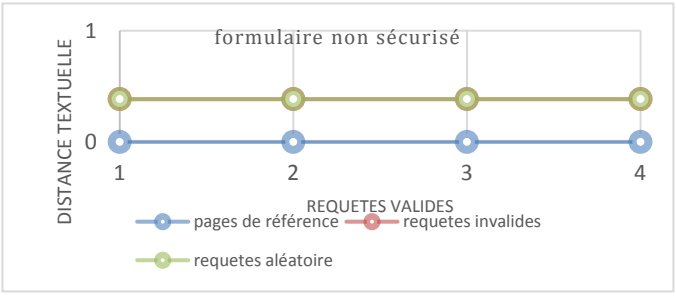

Fig. 16. Curve of the result of a non-secure page (formulary)

Fig. 17. Geometric representation (line) of a non-secure page (formulary)

In figure 13, we see that the distance of Inval/Aleat is 0 which is our algorithm singularity point (authentication form).

In figure 14 and 15, we see that the distance Val/Aleat, Val/Inval do not equal 0, this means that the field texts in this authentication formulary are vulnerable. The distance calculated in figure 16 prove that.

For this non-secure formulary, we have a geometries representation showed in figure 17.

B. *A secure Web page*

a) *Secured URL*

For a secure web page (URL), we have the following curves:

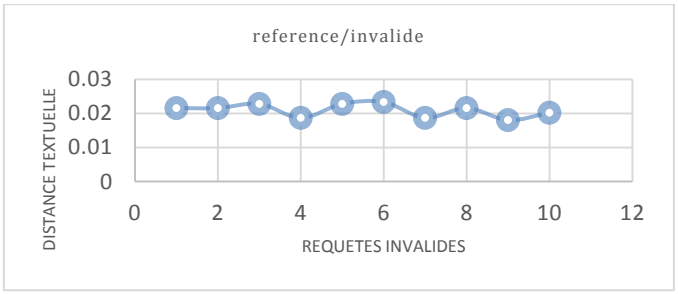

Fig. 18. Curve of the distance between reference and invalid

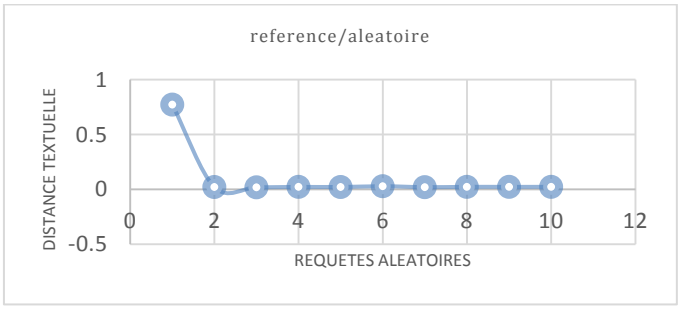

Fig. 19. Curve of the distance between reference and random

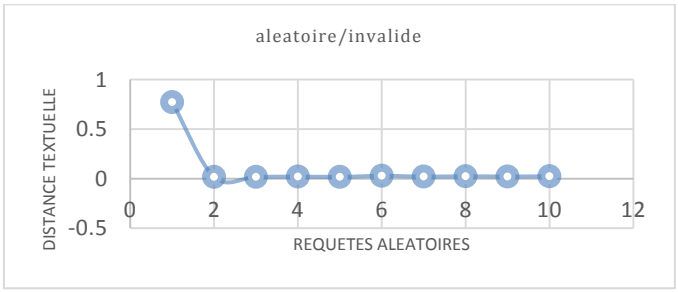

Fig. 20. Curve of the distance between random and invalid

Figures 18,19,20 and 21 shows the distance between all classes, this distance is near 0. This express that all classes are approached one to another wich mean that this unity lead to the security of Web page.

b) *Secured authentication form*

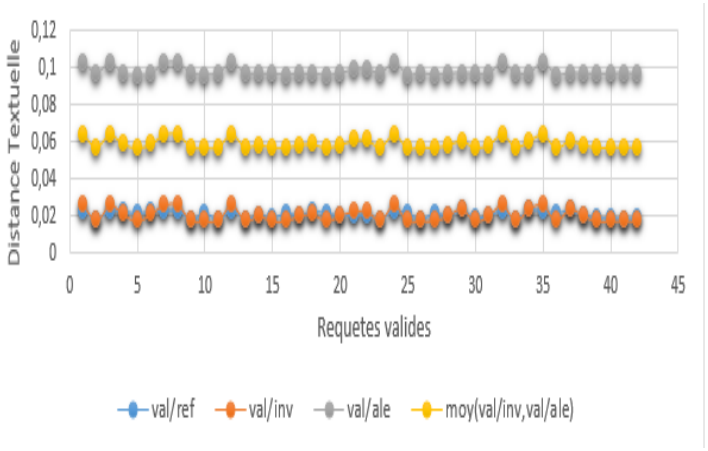

Fig. 21. Curve of the result of secure Web page (URL)

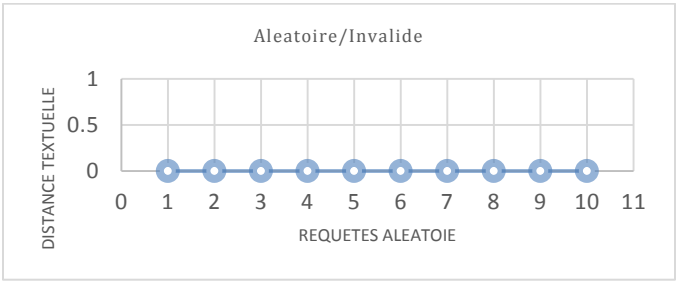

Fig. 22. Curve of the distance between random and invalid

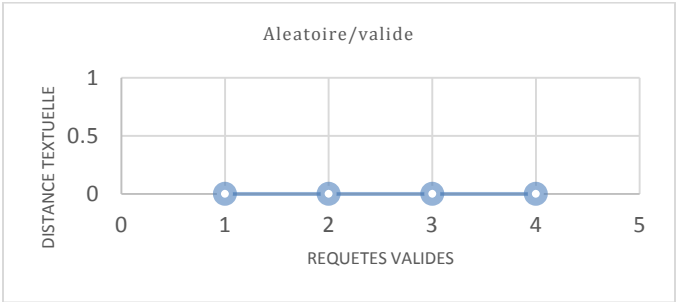

Fig. 23. Curve of the distance between random and valid

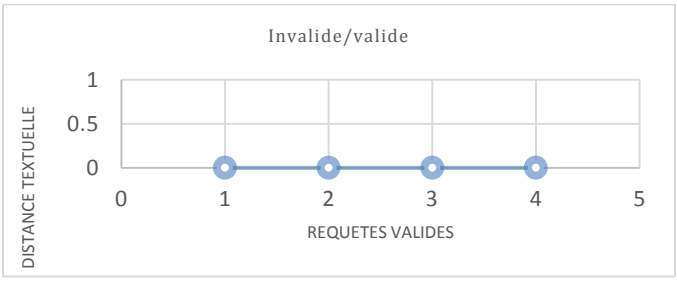

Fig. 24. Curve of the distance between invalid and valid

Fig. 25. Geometric representation (point) of a secure formulary

We apply our algorithm and the one described in the WASAPY model on Web application secure and non-secure, we had the following results (for a non-secure Web page URL):

TABLE I. WASAPY AND LS MODEL RESULT

		WASAPY	LS	
		ε=0.75	IS=0.73	x=0.955
Valid queries	R_{15}, R_{19}, R_{20}, R_{26}, R30, R_{32}, R_{33}, R_{35}	✓	✓	
	R_3, R_{18}, R_{22}, R_{25}	✗	✓	
Number of detection		8	12	
Number of false positive		0	0	
Number of false negative		4	0	

For this page, we see that our algorithm detects all requests that must be detected with a 0 number of false positive and false negative while WASAPY model does not detect certain queries.

And for each non-secure Web application, we had the same result described by the graph below:

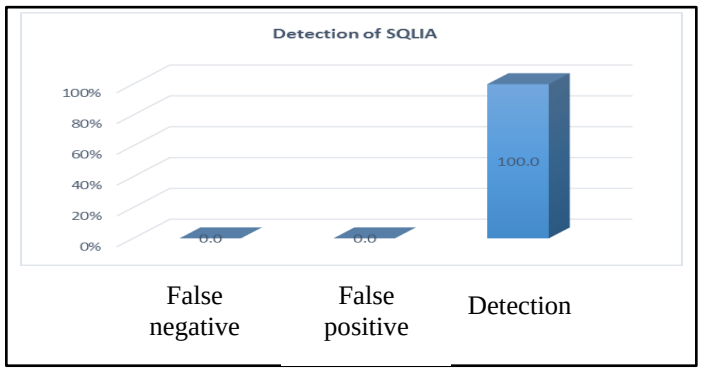

Fig. 26. Curve of the result of a non-secure Web application

IX. CONCLUSION

We used the mathematical variance characterize the dispersion of our pattern. it shows how the statistical series or random variable is dispersed around its average. A variance of zero indicates that all values are identical. A small variance is a sign that the values are close to each other while a high variance is a sign that they are very open. We used the notion of variance to see the coupling factor of a class.

The idea of adding a fourth class that contains only legitimate pages for SQL injection detection in web applications has been successful which is proved by the results obtained and discussed.

The algorithm has a detection performance so interesting that it is wise to conduct an intensive study to confirm its validity, especially since it can detect all potential applications that leverage SQL injection. It should be noted that the model specifically detects requests injection successful that WASAPY approach don't detect. And so, the proposed Model LS, is a good step in website securing field.

REFERENCES

[1] Sectools, http://sectools.org/web-scanners.html, consulté le 20/05/2013

[2] Sourceforge, http://w3af.sourceforge.net, consulté le 22/04/2013.

[3] Google, http://code.google.com/p/skipfish, consulté le 02/05/2013.

[4] OWASP, www.owasp.org, accessed in 05/05/2013.

[5] V. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals", Soviet Physics Doklady, 1966, pages 707-710.

[6] R. Akrout, Analyse de Vulnérabilité et Evaluation de Système de Détection d'Intrusion pour les Applications Web, Doctorat, Université de Toulouse, 2012.

[7] R. Akrout, Analyse de Vulnérabilité et Evaluation de Système de Détection d'Intrusion pour les Applications Web, Doctorat, Université de Toulouse, 2012, pages 53-54.