Presentation of a Pattern to Counteract the Attacks of XSS Malware

Abbas Ali Najjari
Department of Computer,
Zanjan Branch
Islamic Azad University,
Zanjan, Iran

Nasser Modiri
Department of Computer,
Zanjan Branch
Islamic Azad University
,Zanjan, Iran

ABSTRACT
Cross Site programming (XSS) is the script attack in web pages, and it is accounted as one of the most dangerous problems of web applications. The researchers of security have investigated on different problems and they have found that the XSS vulnerability exists in many of known websites. The vulnerability is applied when an attacker reaches to an authorized user's web explorer optionally and he/she might do cookie theft, develop destructive software, thieve the session and change the path of destruction. The validation of the user's input is the first obstacle to protect the web applications against this vulnerability. The main aim of improving the security of web applications is improvement in the quality of user's input validation. Unfortunately, the web application developers usually forget the user's input validation and/or implement a weak validation. In this paper, it is attempted to present a pattern to validate the user's input correctly in the web applications, and the obtained results are compared with the tools of scanning the existing vulnerability.

Keywords
web vulnerabilities, input validation, XSS malware

1. INTRODUCTION
Using the web applications for social communications, hygienic services and financial operations is growing. Unfortunately, the software vulnerability has become a vital issue for web applications. According to the latest security report statistics of websites, 47% of the websites have XSS security problems [15]. In year 2015, the open web application security project (OWASP) [1] and the common weakness enumeration (CWE) [3] reported the XSS vulnerability as the most prevalent vulnerability of web-based applications. Abusing this vulnerability may have severe influence on the organizations. The main reason for XSS vulnerability may be due to weakness in the application code which is because of the weakness existing in the programming language or lack of correct authorization of user's input or lack of observing the security standards in coding by the software developers [14]. Each time the destructive inputs run without required validation by the application, this vulnerability occurs and it causes that the hackers may thieve the user account, obtain the cookie, send arcane information or disconnect the service and perform many other destructive activities through entering destructive scripts into the input. According to the variety of programming languages and lack of problem comprehension development by the programmers and their unfamiliarity with secure coding methods, this vulnerability exists in most of the web-based applications. Therefore, validating the user input is the first obstacle for protection of web applications against the vulnerability. Improvement in the quality of validation of user input may be the main objective of improvement of web applications. Unfortunately, the web application developers usually forget the validation of user input or implement a weak validation[6].

2. LITERATURE REVIEW
In this section, the studies, the methods and the tools presented for counteracting these attacks are reviewed.
Only the cases of testing with sufficient vectors of attack may force the main and mutant to do different behaviors. Hossein Shahriari and Mohammad Zakeri created the error-based testing tool (MUTEC) which changes the sensitive phrases in program sentences through operator mutation and reduces the generated error during running [5].

The static analysis method recognizes the polluted inputs from the external data resources; tracking the polluted data flow and investigation on this point that whether the sink data, e.g. SQL phrases have reached to output HTML sentences or not. Benjamin and Monica Lahm were using binary decision making diagram for analyzing the considered points in the scripts of the server side. Their approach is to determine the pattern of vulnerability in inquiry programs [9].

Static method-based analysis has caused inability in recognizing the incorrect functions in the protection. Diode et al developed Saner tool which investigates on the accuracy of protection functions to counteract the XSS attacks. This method is a replacement for PIXY method which proceeds to recognize the error methods potentially in the protection through using static analysis like what was suggested by Weserman and Sue, and then it proceeds to counteract this type of attacks in a time in which the injection occurs using a series of inputs including string attack and the method of checking it. Static-based analysis may prevent from XSS attacks in many cases and its defect is that it often causes generating many incorrect positive cases [4].

Web pages which are scripted by ASP>NET might have a series of vulnerability which is not observable by its owner. Adneasa et al have presented an algorithm for recognizing the security vulnerabilities in pages. This algorithm performs the exploration operations in websites and applications. This vulnerability scanning tool works on the applications written based on ASP.NET and having C# and VB programming language codes. For this purpose, the scripted program determines a report of defects and weaknesses and vulnerabilities in webpages based on this point that which files and which texts have them. This algorithm may help the organizations rectify the vulnerabilities and raise the security of their programs [2].

The aim of security test is to recognize defects which may cause abusing in order to perform attacks. In this paper, a method is presented which acts in recognizing the

vulnerabilities in web pages using the practical tests. In this order, at first, it saves the HTML pages which have no vulnerabilities, and then using valid data, it saves the pattern of valid pages, and then the HTML output of each input entered by the user is compared with valid patterns. If it is not similar to the pattern, it is stated as an invalid and vulnerable input. Among the disadvantages of this method, abundant incorrect positive production can be mentioned [12].

Presenting an approach to recognize the vulnerabilities created through injecting code in web pages is a method presented in this paper through incorporating the deductive model into evolutionary fuzzy logic. The deductive model is used to obtain the behavior of the web-based application and based on its findings, different inputs are produced using genetic model. The genetic model produces the considered inputs automatically through automatic learning to be recognized through vulnerable input [13].
Due to lack of filtering the user input which enters the web pages, the XSS vulnerability is generated. It is widely used by the software developers and the inspectors. Studying in this field is utilized as an approach to check and investigate the program code which causes changing the incorrect code into the coding of the secure model. Also some guides are presented to improve the coding of the program model into the model counteracting the XSS attacks [8].

The input validation test may detect and neutralize the XSS vulnerability in the input programs. The IVT features-based method, creates the test cases with the aim of applying a combination of valid or invalid inputs in their features. In order to prevent the unique dependencies in the characteristics, Neoli et al attempted to be aware of valid input conditions through analyzing the input fields and their surrounding texts in the scripts of the receiver side. The code-based IVT analysis method applies the static analysis of server-side codes to extract the valid or invalid input conditions. Generally, to a great extent, the effect of both code-based criteria and methods relies on the evolutionary of features or the sufficiency of the collection produced in the test for detecting the XSS vulnerabilities in the source code. Among the disadvantages of this method, lack of preventing the incorrect positive valid inputs can be mentioned [10].

3. TOOLS OF SCANNING THE VULNERABILITY OF WEB PAGES

There are different tools to detect the vulnerabilities of web pages, but generally, the available scanning tools which perform the validation of the user input in web pages are divided into two classes based on crawl and proxy in the view of their performance [1].

In tools which perform based on crawl in the web pages such as Nikito2, Wikito, Acunetix Web Vulnerability Scanner tools, the web pages initialize the inputs automatically based on initial definitions and they translate their results, and no control is carried out on the validity of the input data which whether the input format is entered based on the definitions or not.

In the opposite, the tools which work based on proxy, e.g. (Fiddler, Burp Proxy, Tamperie) tools, allow the developers edit the input in the web pages directly, but they do not give them any help with respect to production of different input tests. In the following table, the features of these tools are compared with each other [10].

According to table.1, it can be concluded that crawl-based tools are appropriate to counteract the XSS attacks. In this project, Acunetix web vulnerability scanning tool is used in order to compare with the suggested pattern. This tool is used in most of the studies as a pattern for detecting the XSS vulnerabilities [7].

Table 1. Comparison between the tools of scanning the vulnerabilities of web-based programs

Characteristic	Crawl Base	Proxy Base
Determining the type of the input field	Yes	No
Production and edition of experimental inputs	No	Yes
Counteracting SQL injection	Yes	No
XSS with predetermined input	Yes	No
Producing invalid input based on valid input	For entering the system	No
Predetermined test	SQL,XSS	No
Content test	No	No

4. PROPOSED MODEL

In this model, an approach is presented through benefitting from the advantages of the current method which prevents from occurrence and creation of this type of attacks through accurate validation of inputs and creation of black and white list. The steps of the suggested model is as below:

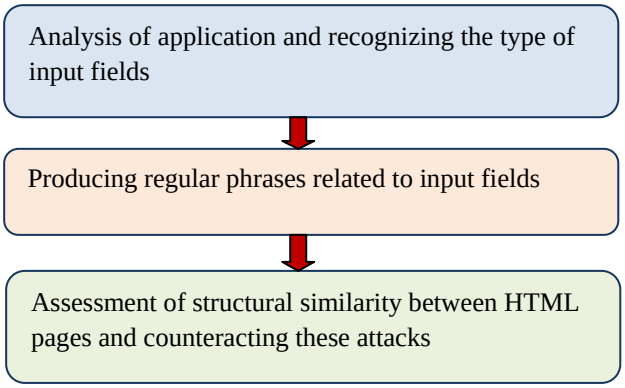

Fig 1: Overall procedure of suggested model

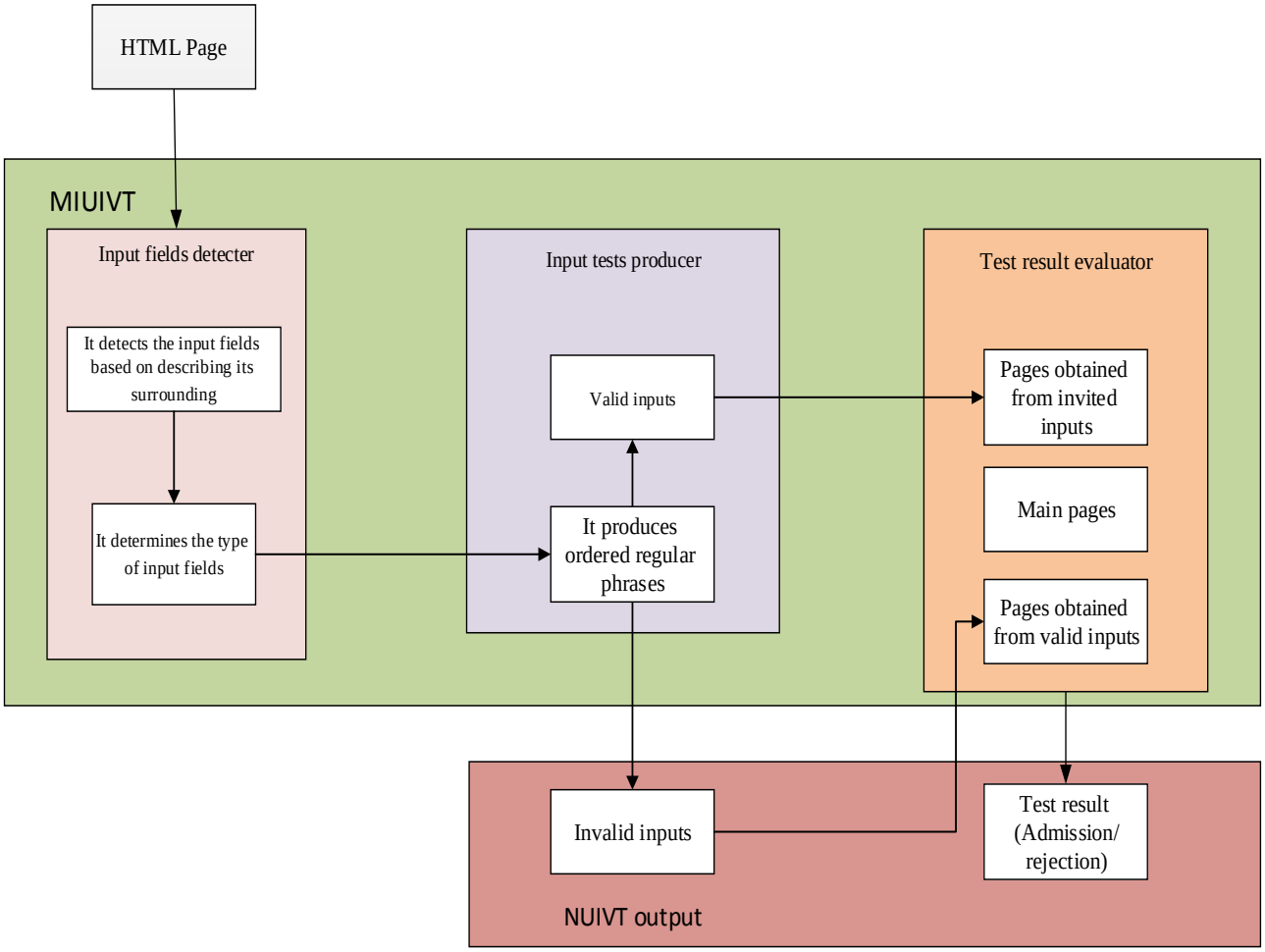

Fig 2: Steps of NUIVT algorithm

5. SUGGESTED METHOD

In this section, the suggested algorithm is presented. Figure 1 represents the general procedure of the suggested pattern. For easiness, the suggested algorithm is named NUIVT (Novel User Input Validation Test). The steps of function of the algorithm is described in the next section.

5.1 NUIVT algorithm

The NUIVT algorithm includes three distinct steps:
The first step includes analyzing the user input fields and detection of the filed types in the input forms.
The second step includes types of input based on regular phrases and based on the filed type defined by the user.
The third step is testing types of invalid inputs in order to detect the vulnerability and counteracting it based on comparison between the structure of the pages translated from valid and invalid inputs.

5.2 Steps of algorithm

The main objective of the NUIVT algorithm is to produce invalid inputs in order to assess the results obtained from the validation test of user input for web pages in the side of the service-recipient. The NUIVT input is a HTML page and the NUIVT output includes invalid inputs and validation test of user input.

Figure 2 represents three main elements of NUIVT with relative details. In the first step, the input fields searcher, detects them and analyzes the texts surrounding the input fields in the web pages, then determines the type of input field based on analyzing the keywords in the description surrounding it through detecting the input points. In the second step, in order to define valid inputs for types of input field based on user's definition, the type of input field is corresponded to a regular phrase. Then, the test inputs producer causes producing invalid inputs through disordering the regular phrase. Also, the test inputs producer produces valid inputs according to regular phrases.

In the third step, the test results evaluator compares the structural similarities between the page obtained from the results with invalid inputs and the page obtained from valid inputs and also the main HTML page, and it specifies the test results to see if there is any vulnerability.

5.3 Input tests producer

After detecting the input fields and describing them, in this section, the method of producing the test inputs is carried out for the input fields based on their description. Based on description of the input fields, the type of input field may be detected. The postal code and the email address are examples of these fields.

Based on the input field type, the input test producer, correlates each input field with a corresponding regular phrase which defines the valid inputs for each input field based on it

in interaction with the user. For facilitating the work, at first a regular default phrase is considered for each field, then the input tests producer obtains the invalid inputs through disassembling the regular phrase in valid inputs. Using invalid inputs test, the vulnerability of lack of correct validation of user input may be detected for web application at the test time. If the vulnerability of user input validation exists in under-test web-based applications, the invalid inputs cause producing unconventional behavior of web-based applications.

It is why in order to produce the invalid inputs, after corresponding an input field with a regular phrase which defines valid inputs for the input field, the input tests are produced through disassembling the regular phrases using the below laws:

The definitions of operation laws RU1 to RU6 are:

RU1: removes the compulsory and obligatory phrases from regular phrases.

RU2: the order of rowing of the phrases is disassembled.

RU3: the number of iteration of the elements in the selected collections is changed.

RU4: selects elements from the complementary collections in the collection of regular phrases, particularly, it adds the characters to it in the input field domain after the boundary values.

RU5: invalid and dangerous characters such as empty string, strings started with greater period, and very long strings are added to the regular phrases.

RU6: adds the particular pattern of XSS to the regular phrases.

If our input field is our email address, then based on the library of regular phrases, the corresponding regular phrase would be:

[\w-\.]+@([\w-]+\.)+[\w-]{2,4}

If any of the disorder laws from RU1 to RU6 runs once in the corresponding regular phrase, six disordered phrases are obtained as follows:

RU1: [\w-\.]+ ([\w-])+[\w-]{2,4}

RU2: @[\w-\.]+([\w-]+\.)+[\w-]{2,4}

RU3: [\w-\.]+@([\w-]+\.)*+[\w-]{5,}

RU4: \d+@([\w-]+\.)+[\w-]{2,4}

RU5: [\w-\.]*@([\w-]+\.)*[\w-]{2,4}

RU6:[\w-\.]+@([\w-]+\.)+[\w-]{2,4}+{XSSExpression}

In order to obtain that which one of the vulnerabilities of user input validation exists in web-based applications, the disordered phrase is applied separately, and it is recorded in each step of the obtained results.

5.4 Results evaluator

In general, the behavior of web-based applications against different inputs can be divided into three types of processes in the below:

Defensive: if the web application detects and rejects the invalid input, the web application is introduced as defensive, and the test result is acceptable.

Non-sensitive: if the web application accepts invalid input, the web application is introduced as non-sensitive or failed, and the test result is unacceptable.

Defenseless or falling: if the web application represents obvious error against the invalid input, such as The Page Not found, it is falling or defenseless.

In practice, the web applications are defensive. If the application can detect the invalid input, three types of pages are translated by defensive web applications.

Its type-1 page is similar to the main page along with a series of guidance, e.g. enter the email address correctly.

The page type-II does not translate any forms and it only contains a series of guidance.

Its type-3 page conducts the user to another page.

In the second state, the translated page is very different with the main page and even the page translated from valid results. In the third state, it cannot be detected accurately and automatically, because the web application may refer to each one of the pages on the server. The web application is non-sensitive when the page translated from invalid input is similar to the page translated with valid input. Two HTML pages are similar if in terms of structure, when they are opened on the explorers, they are similar. The application under web has fallen when the translated page has a series of error with codes such as The page not found 404. In such cases, the web application accepts the invalid input and it opens a page representing leakage of information from the configuration or the internal function of the program which opens a text error. This information may be a leverage to perform dangerous attacks even automatically which all of it is given in OWASP.

Detection of the similarity of two webpages on the explorer through observing them is not easy. For this purpose, in order to detect the similarity between the web pages, they are opened based on the order of the tags, and their similarity is detected based on the arrangement of the tags, because the similarity of two webpages depends on their placement style and the order of their tags. For this purpose, a mechanism incorporating XSLT language and LCTS (Long Common Tag Sequence) LCTS algorithm is used. Before assessing the invalid inputs, based on the regular phrase related to the valid inputs, the structure of the pages translated from these inputs and the structure of the main pages are saved in the related data platform. After this step, the invalid inputs test begins. The pattern which is used for testing these inputs is given in figure. 3.

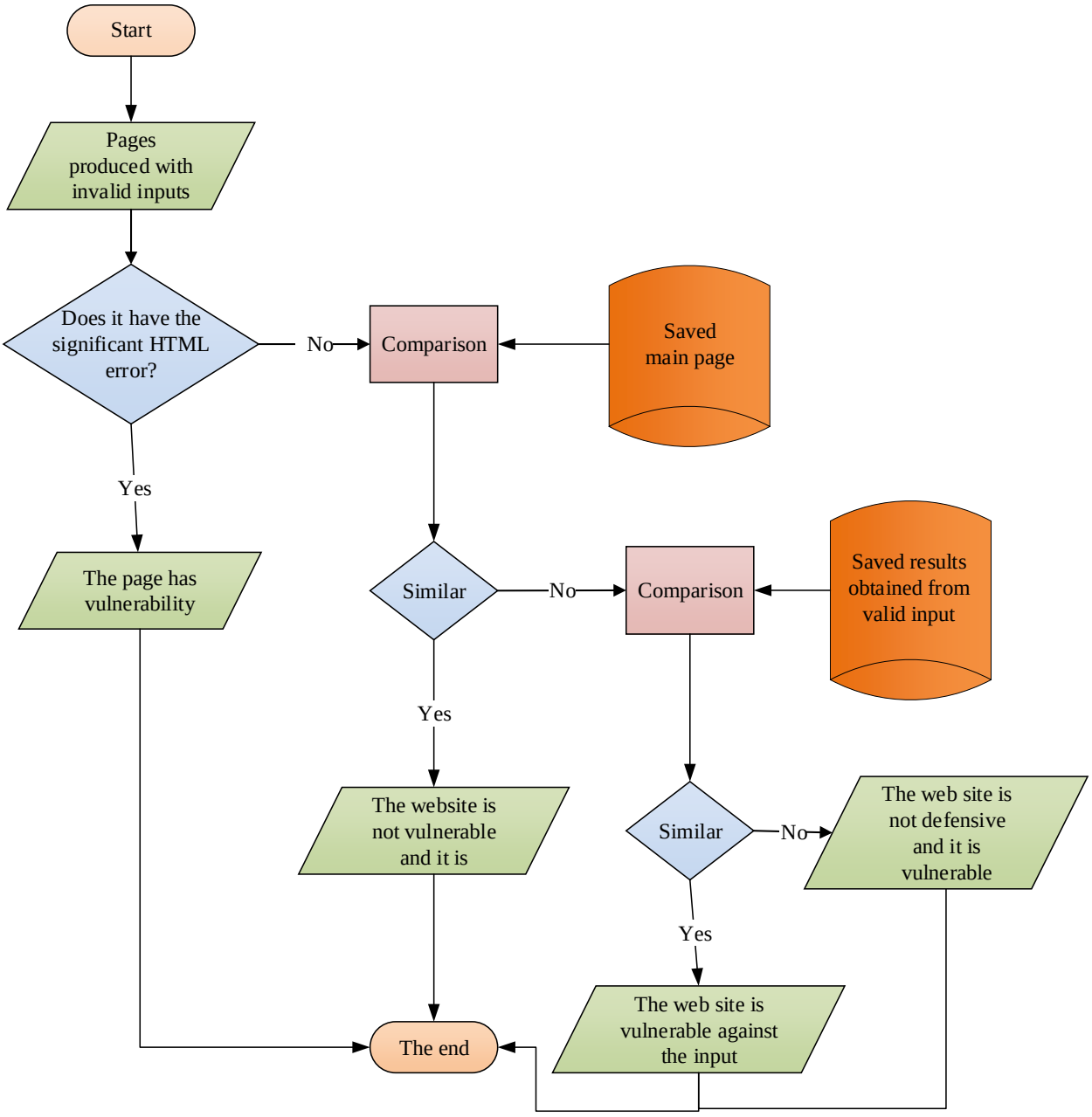

Figure.3. Evaluator of similarity between HTML pages

At first, it is controlled whether the HTML page translated from invalid inputs has predetermined errors of HTTP or not, such as "The Page not found". If the answer is yes, the application falls and it is vulnerable. If the answer is no, the page translated from invalid input (black list) is compared with the main page and the valid inputs (white list). If the page translated from invalid inputs is similar to the main page structurally, the invalid input is prevented and the application is defensive. If the page translated from the invalid input is similar to the page translated with the valid input, there are two possibilities:

1) The application is not sensitive, because it has accepted an invalid input.

2) The application is defensive and it is sufficiently strong and it has switched to a page having a valid input. In these conditions, if the page translated from the invalid input is different with the page translated from valid input and the main page, the web application is known as defensive. In each step, after completing the related process, the inputs which are obtained based on the results are saved in order to raise the intelligence of the system in white and black lists in order to prevent from repetition of the comparisons.

6. EVALUATION OF THE TEST RESULTS

In order to illustrate the influence of the input test based on NUIVT, in order to recognize the vulnerabilities of the test website for detecting the vulnerabilities of the test website, 22 vulnerabilities of user input were designed according to table no.2 and the tests required in the test website were carried out through applying the laws RU1-RU6, and the scanning tools of vulnerability of web pages, e.g. Acunetix Web Vulnerability Scanner 10, and the obtained results were compared. The obtained results represented that the presented

approach has completed the Acunetix vulnerability scanning tool through testing the greater range of inputs and correct validation of user input.

Table 2. Results of the obtained test in comparison to the Acuntix scanner

Type of input submission	Type of generated error	Detection with RUs	Percentage of error detection	
			NUIVT	Acuntix
Length of invalid input	Invalid sub-field	RU1,RU5	2/2	0/2
	Pointing to empty value	RU1		
	Overflowing of buffer	RU1		
	Out of the range	RU1		
Invalid type	Submission of invalid and out-of-range numbers and strings and	RU1-RU5	4/4	0/4
Invalid value	Overflowing of memory or buffer	RU5	7/9	0/9
	Error of lack of file	RU1-RU5		
	Submission of false output key	RU5		
	network disconnection error (false port or submission of false name of server)	RU5		
XSS injection	Running un-expectable scripts	RU6	7/7	7/7

Invalid inputs which are produced based on RU1-RU6 laws, are divided into four classes of invalid input length, invalid type, invalid value and injection of particular code of XSS which are placed in column 1 of table 2. The second column of table 2 represents the type of the generated error. The third column represents the law used to produce the applied input tests and which has caused error. The two last columns represent the number of vulnerabilities which are recognized in both approaches. This number includes number of vulnerabilities in type out of the detected vulnerabilities.

Finally, the total number of vulnerable are counted manually and the results are controlled.

7. CONCLUSION

As it was stated, XSS is one of the ten main vulnerabilities of web applications according to OWASP report. With the growing trend of using the web applications which are mainly developed from web programming languages in HTML pages, not only the vulnerability is not annihilated, but the hackers could plan dangerous attacks by this malware and influence on many websites through new swindles and exploiting from the defects or programmers' mistakes in coding. In this research, a particular look is taken to the validation of the user input. This approach produces a wide range of inputs in web-based applications. In most of the tests carried out, usually the comparison with the white list or the same valid inputs is considered, and the main factor which is not considered is that an invalid input might exist having the same valid input reaction due to weakness in the security, hence, in these conditions, comparison with the white list has no validity and it causes incorrect positive production. In this research, the range of input tests is developed through considering the type of input fields by defining the regular phrases corresponding to these fields, and all valid inputs are produced and then based on them, invalid inputs are produced and comparison with black and white list has become possible. The tag-based presented approach analyzes the HTML pages produced from different inputs which this trend causes accurate distinction between valid and invalid inputs.

8. REFERENCES

[1] Andrews, A., Offutt, J., Alexander, R. "Testing web applications by modeling with fsms". Software Syst. Model. 4 (3), 326–345, 2005.

[2] Avancini, M. Ceccato, F.B. Kessler, "Grammar Based Oracle for SecurityTesting of Web Applications", in: 7th International Workshop on Automation of Software Test (AST), no. line 13, pp. 15–21, 2012.

[3] Common Vulnerabilities and Exposures (The Standard for Information Security Vulnerability Names) http://cwe.mitre.org/

[4] D. Balzarotti et al., "Saner: Composing Static and Dynamic Analysis to Validate Sanitization in Web Applications," Proc. 29th IEEE Symp. Security and Privacy, IEEE CS, pp. 387-401, 2008.

[5] H. Shahriar and M. Zulkernine, "MUTEC: Mutation-Based Testing of Cross Site Scripting," Proc. 5th Int'l Workshop Software Eng. for Secure Systems (SESS 09), IEEE, pp. 47-53, 2009.

[6] Isatou Hydara, Abu Bakar Md. Sultan, Current state of research on cross-site scripting (XSS) – A systematic literature review, Elsevier, Volume 58, February, Pages 170–186, 2015.

[7] José Fonseca, Marco Vieira, Henrique Madeira, "Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks", in: 13th IEEE International Symposium on Pacific Rim Dependable Computing.p.365,372,2007.

[8] L.K. Shar, H.B.K. Tan, "Predicting common web application vulnerabilities from input validation and

sanitization code patterns", in: Proc. 27th IEEE/ACM Int. Conf. Autom. Softw. Eng. – ASE, p. 310, 2012

[9] M.S. Lam et al., "Securing Web Applications with Static and Dynamic Information Flow Tracking," Proc. 2008 ACM SIGPLAN Symp. Partial Evaluation and Semantics-Based Program Manipulation (PEPM 08), ACM, pp. 3-12, 2008

[10] N. Li et al., "Perturbation-Based User-Input-Validation Testing of Web Applications," J. Systems and Software, Nov. 2010, pp. 2263-2274

[11] Open Web Application Security Project, XSS (Cross-Site Scripting), Prevention Cheat Sheet, 2015; https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_ Prevention_Cheat_Sheet.

[12] R.B. Brinhosa, C.M. Westphall, C.B. Westphall, "Proposal and Development of the Web Services Input Validation Model", in: IEEE Network Operations and Management Symposium (NOMS), pp. 643–646, 2012.

[13] R. Komiya, I. Paik, M. Hisada, "Classification of malicious web code by machine learning", in: 3rd International Conference on Awareness Science and Technology iCAST, pp. 406–411, 2011.

[14] S. Fogie et al., XSS Attacks: Cross Site Scripting Exploits and Defense, Syngress, 2007.

[15] White Hat Security Website Stats Report 2015, https://info.whitehatsec.com/Website-Stats-Report-2015.html