Original Research Paper

Volume : 5 | Issue : 11 | November 2016 • ISSN No 2277 - 8179 | IF : 3.508 | IC Value : 69.48

Automated Testing for Vulnerabilities reduction in Preventing Structured Query Language Injection, Cross Site Scripting and Request Forgery Attacks in Web Application

Engineering

KEYWORDS: Fuzzy inference, Selenium Tool, Validation, Vulnerabilities, Web Application Attacks.

I.Shahanaz begum — Department of Computer Science and Engineering

G.Geetharamani — Department of Mathematics, 9597502104 BIT Campus, Anna University, Trichirappalli-24, India.

ABSTRACT

Current E-Commerce applications are vulnerable to many Web based Attacks. Many vulnerabilities exist in Web Application to perform SQL Injection, XSS and CSRF Attacks. The form fields of the Website are prone to such vulnerabilities. Testing the form fields is an involved process and hence this work attempts to propose an automated process for the testing by generating testcases. TestNG in the Selenium Testing Tool generated testcases to provide validations for the form fields of the Forum Website and they are executed in parallel making use of the Multithreading support provided by this tool to prevent the mounting of the related attacks. The execution time for these testcases is compared with that of the testcases not including Multithreading support and the results are found to be very encouraging. The validations may be provided at the Client side, Web and DB Server side. Session tracking techniques are also provided to prevent some of the attacks. The results are validated with the help of Sugeno Fuzzy Inference System. The experiments are carried out on the Web pages of a Forum Website that includes the detection of the Vulnerabilities injected into the Web application with the help of a Web Application Vulnerability Scanner.

1. INTRODUCTION

Security mechanisms are to be in-place for the Web applications deployed for implementing the activities dealing with educational, financial or any business organization. The attacks targeting the Application layer are not addressed by firewalls [4]. A survey on various threats and subsequent attacks on the Web Application has been given by Mary et al [5]. More than 80% of security attacks exploit the vulnerabilities at the application layer[2]: As per the report given in 2015 by Open Web Application Security Project(OWASP) web application vulnerabilities have become the cause for introducing the privacy breaches. As per the Website security statistics report, 56% of the vulnerabilities are due to information leakage and 47% of attacks are due to Cross Site scripting and 24% of the attacks are due to Cross site request forgery, 11% insufficient authorization, 11% session fixation and 16% URL Redirector abuse [1].

Jose Fonseca et al., deal with injection of vulnerabilities [6]. The technology such as Transport Layer Security [7] has been attempted to provide secure communication between the client and the Web application but it is impossible to prevent sniffing and man-in-the middle attacks.

To achieve secure design Diansxiang Xu et al., [8] uses Predicate/ Transition nets for modelling system functions, security threats and security features. XSS the serious attack appearing atop the list of attacks is not addressed.

Angelo et al., deals with the dynamic Websites creation including the set of objects such as HTML tags, script functions, hyperlinks and other advanced features allowing malicious codes injection [9]. This work does not deal with other attack possibilities and with the session monitoring.

Meixing Le et al., describes how machine learning technique reduces the false positives to a greater extent while testing a static Web application [10], but there was an issue in testing dynamic Web applications.

In this present work, not only the vulnerabilities are injected into the Web application but also the countermeasures are provided to prevent the attacks from occurring by automatic testing and execution of the testcases of different pages of the Website. Testing can also be carried out in the private Cloud [11].

Testing for security is a difficult process as it is impossible to test a real-world program against all invalid inputs [3,13]. Hence, it is highly desirable to develop automated cost- effective testing techniques for detecting software vulnerabilities.

Cross Site Scripting(XSS), Cross Site Request Forgery(CSRF), Structured Query Language (SQL) Injection and forceful browsing etc. are some of the most common and serious Website Attacks.

Testing techniques prevent the attacks by fixing the vulnerabilities and the proactive session handling methods aid in complete eradication of the various attacks.

TestNG generates and executes the testcases to prevent the bypassing of validation checks. DB Side validation is specifically done with the help of Analytical Tool R. The results were established with the help of Fuzzy Inference System(FIS) [12].

The validation check is provided at the client side, Web and DB Server side of an application. There are less chances of dealing with false positives as all the testcases are generated dealing with vulnerabilities. This current paper deals with the dynamic Web application in which the generated testcases provide countermeasures against SQL Injection and XSS Attacks.

The Intent-of-Attack and Sufficient Number of Validations are chosen, as the parameters of this present work and experimental results show that high risk causing vulnerabilities for performing various Web Application Attacks are prevented. The results are justified with the help of FIS. There are other soft-computing techniques to evaluate the performance [14].

TestNG supports execution of testcases in parallel for fixing the vulnerabilities causing various types of Attacks. The multithreaded support extended by TestNG is utilized in the present work which also reduces the execution time in handling the different attacks. The time complexity for Stored XSS Prevention algorithms is O(1), for Reflected XSS Prevention algorithm it is O(n), for SQL Injection prevention it is O(1) and for session management it is O(log n). These results are found to be appreciable compared to the testing system where in one TestSuite all the testcases are grouped and executed. The performance of the attack injection, detection and prevention are also validated.

The paper is organized as follows: Section 2 discusses the techniques utilized in the present work. Section 3 sketches the formation of Fuzzy rule base for the Web Application to derive the output from the experiments and also introduces the algorithms for validating the inputs received through various vulnerable fields. Section 4 discusses

the experimental results and the performance of this work with the help of popular Web application vulnerability scanner. It also evaluates the impact of adequate validation techniques on the Website's vulnerable fields. Section 5 is on conclusion.

2. Present Work

TestNG feature in the Selenium Tool generates testcases to prevent the serious attacks targeting the Web application. The generated testcases cover all the possibilities of exploitation events and the testcases are associated with form fields of different Web pages of the Forum Website. These testcases prevent bypassing of validation checks at the Client, Web and DB Server sides. The methodology involved in this work has identified two input parameters namely Intent-of-Attack and Formfield validations. The related output variables are identified as ranking of the application, trustworthiness of the application, reduction in vulnerabilities, risk involved in the application and information leakage.

The following Algorithms (1 – 4) are developed for the prevention of XSS and SQL Injection Attacks and for Efficient session Management.

Algorithm 1. For prevention of stored XSS attack
Input: Form field values 'f' as input, Records 'r' in db
Output: Message

For every Webpage of the site, repeat the following steps.
1: Get input from the form fields
2: Do client side validation using

JavaScript and regex
i) If special characters exist, end the process and show block message.
ii) Else proceed.

3: Do server side validation
i) Get values from client side and proper validation done using prepared statement.
ii) If executed, DB operations are performed successfully.
iii) Else show block message.

4: Analyze entire database using R script for existence of any malicious scripts
i) If exists , delete that specific row and update the table
ii) Else update the table

Time complexity involved in this algorithm is **O(1).**

Algorithm 2. For prevention of Reflected XSS attack
Input: Url parameter 'u' or search textbox value 's'
Output: Block Message/Result.

For every Webpage of the site, repeat the following steps.
1: Get Current URL of the page.
2: Validate URL contents

i) If any script tags or java script present Then
Block harmful navigation/unexpected navigation.
ii) Else Navigate to next page.

3: Get value entered in search box
4: Validate at the client side
i) If value contains script tags then show block message.
ii) Else execute and show result.

Time Complexity involved in this algorithm is **O(n).**

Algorithm 3. For prevention of SQL injection attack
Input: login Form fields 'f' as input / Url parameter 'u' / search box value 'v'
Output: message

1: Get value from login page/ form fields.
2: Do client side validation
i) If value contains special characters (piggy backing, tautologies) then Show block message.
ii) Else execute and show result.

3: Do server side validation
i) Execute prepared statement/Stored procedure
ii) If executed, db operations are performed successfully.
iii) Else show block message.

4: Get Current URL content after "?'

5: Validate at the client side
i) If query strings exist then post back to previous page.
ii) Else execute and show result.

6: Decode the url contents
i) If contained query strings then go back to previous page.
ii) Else execute and show result.

Time Complexity involved in this algorithm is **O(1)**

Algorithm 4. For Session Management
1: Session will be created when user logs in.
2: Created session value will be stored in HttpSession Interface
3: Current session values will be stored at the Client side and Server side.
4: If attacker is trying to steal the session using malicious scripts entered in the URL or Search box

4a.Invalidate the user session
4b.Alert the user about session hijacking and start the new session

Worst case Time Complexity involved in this algorithm is **O(log n)**

The testcase reports are generated in Excelsheet with the columns labelled with Testcase-Id, Input field names, the Expected output, Actual output and the Status of the testcase after execution. These reports are generated for every page of the Website. It is expected that the testcase status should be PASS after executing a particular testcase associated with a form-field, which means that the sufficient number of validations are provided to overcome the vulnerable inputs. In all the attack pages of the Forum Website the testcase status should be FAIL from which it is inferred that the expected validations are missing. The aforementioned facts were experimented and achieved the expected results for the Attack instant and Preventive instant of the Vulnerable Website and Protected Website respectively. The PASS testcase status will be obtained when the expected output and the actual output column values match. The actual output column and testcase status columns will be filled up after the successful execution of the respective testcases. These experiments are conducted using Glassfish server in the NetBeans IDE along with the MySQL Server. Then the results are validated with the help of FIS in MATLAB.

2.1 Experiment and Case Study

The experiment was conducted on the Vulnerable and Protected Forum websites. The Stored XSS Attack can be performed against the victim to hijack his session. To prevent the stored XSS Attack whenever the victim retrieves information associated with any Web page got from the db server, it is checked for the presence of any malicious scripts, if found they are filtered by checking against the list of unexpected patterns. By this session stealing is prevented. Even before the script is executed they are filtered and hence the attacker cannot extract the session information.

In the case of Reflected XSS Attack the injection can be done either through the URL in the address bar or through the search input field or any vulnerable input field. Whenever the injection is done in the URL, this information can be extracted by the execution of the

Volume : 5 | Issue : 11 | November 2016 • ISSN No 2277 - 8179 | IF : 3.508 | IC Value : 69.48

associated Testcase using TestNG feature of the Selenium tool. The exploitation of the related vulnerability to succeed in performing the Reflected XSS Attack is prevented.

The attacker injects scripts in vulnerable fields to propagate it to the Web server and the validation is done at that level using prepared statement and stored procedures to filter the unwanted malicious scripts from getting executed, leading to stealing of the session information and to mount the Reflected XSS Attack. As a support of proper session management technique monitoring action is carried out whenever the script is passed to the Web or DB Sever side from the Address bar or through search input box or any equivalent vulnerable field the session is invalidated to prevent misuse of session details by the attacker to subsequently mount the Stored XSS or Reflected XSS Attacks.

For performing CSRF Attack the attacker targets to obtain the victim page URL. To prevent this CSRF Attack the exact page URL is hidden using Web server side coding to make CSRF attack a failure attack. By the server side validation redirection to next page or URL jumping is prevented.

Then the Forum Website form fields are identified where more than one possible type of vulnerability can be exploited to perform more number of attacks such as SQL Injection and XSS. The fields are classified based on the types of vulnerabilities that are identified and the corresponding attacks, and based on that two linguistic variables are obtained to hold the input values such as Intent of Attack and Formfield validations. And also the related output linguistic variables namely the Ranking of the application, Trustworthiness of the application, Reduction in vulnerabilities, Risk involved in the application, Information leakage are identified. With the help of Fuzzy logic the membership equations for the two linguistic variables Intent of Attack and Formfield validations are framed. The form fields i.e where only client side validation, where both client side and server side validations and where the validations are applied at the Client, Web and DB Server sides are identified. TestNG in Selenium Tool generates the testcases to carry out these validation checks. DB Side validation is performed with the help of Analytical Tool R.

The different WebPages / Forms with its different form fields of the Forum Website are analyzed to get the information on different possible types of attacks. The results are observed for the reduction in vulnerabilities of the Website with the sufficient number of validations.

Finally in the MATLAB using FIS appropriate graphs in the Rule Viewer window of MATLAB are obtained justifying that this approach has dealt with most of the atop entries in the OWASP released list of attacks.

The results are evaluated using FIS with the following membership function equations involving two input linguistic variables and five output linguistic variables

Membership function equation for the degree of Intent-of-Attack variable is declared using equations 1 to 4.

$$LI(x) = 1 \quad ; \quad x \leq 5 \qquad (1)$$
$$VHI(x) = x-\alpha/\beta-\alpha \quad ; 10 < x \leq 14 \qquad (2)$$
$$HI(x) = \beta- x/\beta-\gamma \quad ; 12 < x \leq 60 \qquad (3)$$
$$\text{Otherwise} = 0 \quad ; \quad x > 60 \qquad (4)$$

FIS helps in inferring the reduction in the vulnerabilities by providing validation checks at the Client, Web Server and DB Server sides of an application. The objective is 1) to improve the ranking of the Forum Website 2) to reduce vulnerabilities 3) to eliminate information leakage 4) and to improve the trustworthiness of the application by improving the Form-field-Validations.

Membership function equation for the validations applied is declared using equations 5 to 8.

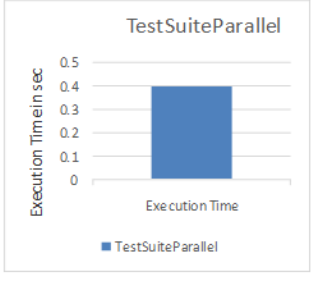
$$CV(x) = 0 \quad ; \quad x < 5 \qquad (5)$$
$$CSV(x) = x-\alpha/\beta-\alpha; 5 < x \leq 24 \qquad (6)$$
$$CSDV(x) = \beta- x/\beta-\gamma; 20 < x \leq 60 \qquad (7)$$
$$\text{Otherwise} = 1 \quad ; \quad x > 60 \qquad (8)$$

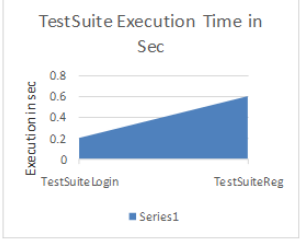
The following Fuzzy rule-base is adopted in implementing the rules to obtain the membership value for targeted output Reduction in Vulnerabilities using the triangular membership function. This is obtained by assigning different degrees(VHI, HI, HI1) to Intent-of-Attack variable and different degrees(CV,CSV,CSDV) to Form field Validations variable.

If(AttackIntent is VHI) and (FFValidation is CV) then (RedVulnerabilities is LOW)
If(AttackIntent is VHI) and (FFValidation is CSV) then (RedVulnerabilities is HIGH)
If(AttackIntent is VHI) and (FFValidation is CSDV) then (RedVulnerabilities is VHIGH)
If(AttackIntent is HI) and (FFValidation is CV) then (RedVulnerabilities is LOW)
If(AttackIntent is HI) and (FFValidation is CSV) then (RedVulnerabilities is HIGH)
If(AttackIntent is HI) and (FFValidation is CSDV) then (RedVulnerabilities is VHIGH)
If(AttackIntent is HI1) and (FFValidation is CV) then (RedVulnerabilities is LOW)
If(AttackIntent is HI1) and (FFValidation is CSV) then (RedVulnerabilities is HIGH)
If(AttackIntent is HI1) and (FFValidation is CSDV) then (RedVulnerabilities is VHIGH)

The Reduction in Vulnerabilities with membership-value of 1 in the Rule Viewer window is obtained which helps in inferring that the vulnerabilities could be controlled to a greater extent by providing maximum validation to all of the form fields of Forum Website. The high membership value is obtained by adding Fuzzy rules to the rulebase in the Rule Editor window of the MATLAB by opting different degrees to input linguistic variables.

The following graphs depict the execution time consumed when the testsuites are executed in parallel and also when all the testcases are grouped in a single testsuite and executed in a sequence Graph 1.





Graph 1a. Execution Time for parallel execution of Testsuites and Graph 1b. Execution Time for Testcases in a single Testsuite executed in a sequence

As TestNG provides support for multithreading, the testcases framed for dealing with each of the Web based attacks are executed in parallel and quickly the targeted attacks are prevented as depicted in Graph 1. The complexity involved in the execution of the algorithms in the best case is O(1). In the worst case the algorithm complexity is O(n).

3. Results and Discussion

Most of the vulnerabilities causing more risk to the functioning of any organization's Web site is removed by this approach. For effective validation of DB Side the analytical Tool R is utilized. To add to the effectiveness of this present work, session management technique is also utilized.

TestNG a feature in Selenium tool favours in carrying out the testing easily and the expected test reports is generated. The multithreading support provided by TestNG helps in parallely executing the testsuites for handling different types of attacks. This parallely executable testcases reduces the execution time which is in the order of O(1) for most of the algorithms utilized in this present work.

The performance of vulnerablilities injection and detections is evaluated with the help of Commercial Web Application vulnerability scanner namely WebInspect. The results obtained for the current work is compared with the already obtained outputs by scanning the famous forum Websites such as PhpBB and TikiWiki. The Vulnerabilities detection rate is found to be approximately 90% by testing the present work with the WebInspect tool.

In the current work not only the intrusions were injected into the Forum Website but also the countermeasures for preventing them with the help of generated testcases using TestNG in Selenium Tool were provided. The validations are provided at the different layers of the application to prevent the vulnerabilities from getting exploited.

4. Conclusion

A novel methodology is proposed in this paper to detect and prevent the mounting of SQL Injection, XSS and CSRF Attacks on the Forum website by the automated testing using TestNG. The methodology utilized in this present work consists of analyzing each and every input field of the different forms of Forum Website so that the various Web attacks can be detected and prevented. The testcases are generated for performing validation check at each and every form field of the Web application at the Client, Web and DB Server sides. DB side is specifically validated or monitored using R Script. This work is also associated with the proactive monitoring of the Session of any Transaction. The results are established by obtaining a maximum membership value with the FIS in MATLAB. The vulnerabilities injection and detection is evaluated with the help of Open source tool WebInspect. TestNG supports in executing the testcases in parallel for fixing the vulnerabilities causing various types of Attacks. The multithreaded support extended by TestNG is utilized in the present work which reduces the execution time in handling the different attacks. The performance of the present work is validated.

It is planned to utilize the countermeasures that are proposed in this current work for preventing the attacks which may be targeting the Cloud data. It may prevent exploitation of all of the vulnerabilities for mounting the various types of Web related attacks including Session Fixation and Transport layer related attacks.

5. References

1. https://info.whitehatsec.com/rs/whitehatsecurity/images/2015-Stats-Report.pdf.
2. https://www.owasp.org/images/c/c3/f_IAPP_Summit_2015.pdf.
3. H.H. Thompson, "Why Security Testing is Hard?", IEEE Security and Privacy Magazine, vol. 1, no. 4, pp. 83-86, 2003.
4. Cataldo Basile and Antonio Lioy, "Analysis of Application-Layer Filtering Policies with Application to HTTP", IEEE/ACM Transactions on Networking, vol. 23, no.1, February 2015.
5. Virginia Mary Nadar, Leena Jacob and Madhumita Chatterjee, "Different Threats and Attacks on Online Application: A Survey", International Journal of Computer and Internet Security, vol.8, no. 1, pp. 25-33, 2016.
6. Jose Fonseca, Marco Vieira and Hensique Madeira, "Evaluation of Web Security Mechanisms using Vulnerability and Attack Injection", IEEE Transactions on Dependable and Secure Computing, vol. 11, no.5, 2014.
7. http://www.ietf.org/html.charters/tls-charter.html, 2006.
8. Dianxiang Xu, Manghui Tu, Michael Sanford, Lijo Thomas, "Automated Security Test Generation with Formal Threat Models", IEEE Transactions on dependable and Secure Computing, vol. 9, no. 4, 2012.
9. Angelo Edurdo Nunan, Eduardo Souto, Eulanda M. Dos Santos, Eduardo Feitcss, "Automatic classification of Cross-Site Scripting in Web Pages using Document-based and URL-based Features" , ResearchGate, 2012.
10. Meixing Le, Angelos Stavrou and Brent ByungHoon Kang "Double guard: Detecting intrusions in Multi-tier Web Applications, IEEE Transactions on Dependable and Secure Computing, vol. 9, no. 4, 2012.
11. Tomislav Pavic, Ivan Vrbovcan, Marija Sosa Anic Hrvatski, "Testing in the Private Cloud", MIPRO 2015, 25-29, 2015, Croatia.
12. Timothy J.Ross, " Fuzzy logic with Engineering Applications", 3rd Ed, 2010 John Wiley and Sons, Ltd.
13. J.Thompson and J. Whittaker, "Testing for Software Security", Dr. Dobbs J., pp. 24-34, 2002.
14. Rui Wang, Xiaoqi Jia and Qinleti Li, " Machine Learning based Cross-site Scripting Detection in Online Social Network", IEEE International Conference on High Performance Computing and Communications, 2014.