ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

Improve Web Service Robustness: New Security Testing Approach

Jaydeep J. Mangle, Prof. Vina M. Lomte

M.E. Student, Dept. of Computer Engineering, R.M.D Sinhgad School of Engineering, Pune, Maharashtra, India

Assistant Professor, Dept. of Computer Engineering, R.M.D Sinhgad School of Engineering, Pune, Maharashtra, India

ABSTRACT: To have flexibility of providing services available across various platform, we need to expose web services due to its open nature. Widely grown web services use raises new security challenges. Increasing demand has increased challenges on information security; it becomes important to provide robustness to the web services. The various web services attacks such as XML injection, XPath Injection, Cross-site scripting (XSS), may become harmful to the organization. Studies has shown that current different testing available techniques such as penetration testing and fuzzy scanning- generates several false positive and negative indications. However, fault injection technique seems more robust as it provides greater flexibility to modify the test cases and find software bugs. This work describes the fault injection technique with WS-Security (UsernameToken) and development of set of rules to determine vulnerability analysis, resulting on the improvement of vulnerability detector accuracy..

KEYWORDS: Web services; cross-site scripting; XSS attack; penetration testing; fault injection; WS-Security; WSS; Security Token; soapUI; WSInject.

I. INTRODUCTION

Web security is the biggest issue in the corporate world. There is need to use internet in 0the day to day life. The "availability" of Internet and its services means that the information, the computing systems, and the security controls are all accessible and operable in committed state at some random point of time; however, the inherent vulnerabilities of the Internet architecture provide opportunities for a lot of attacks on its infrastructure and services. [10] XSS, SQL injection, Sniffing, Request Encoding and DOS attacks which poses an immense threat to the availability of the Internet. An occurrence of these attacks on the web degrades or completely disrupts services to legitimate users by expending communication and/or computational resources of the target. The web services provide platform for communication between web applications, it allows interoperable nodes communication through World Wide Web. Thus providing confidentiality and integrity for data which has been transferred over the web service communication, is the important aspect. Theft of communicated information will cause biggest financial or infrastructural losses to the organization. There must be testing mechanisms, which provides the robustness to the web applications. This paper discusses the need and operations of such security mechanisms.

Nodes in MANET have limited battery power and these batteries cannot be replaced or recharged in complex scenarios. To prolong or maximize the network lifetime these batteries should be used efficiently. The energy consumption of each node varies according to its communication state: transmitting, receiving, listening or sleeping modes. Researchers and industries both are working on the mechanism to prolong the lifetime of the node's battery. But routing algorithms plays an important role in energy efficiency because routing algorithm will decide which node has to be selected for communication.

The main purpose of energy efficient algorithm is to maximize the network lifetime. These algorithms are not just related to maximize the total energy consumption of the route but also to maximize the life time of each node in the network to increase the network lifetime. Energy efficient algorithms can be based on the two metrics: i) Minimizing total transmission energy ii) maximizing network lifetime. The first metric focuses on the total transmission energy used to send the packets from source to destination by selecting the large number of hops criteria. Second metric focuses on the residual batter energy level of entire network or individual battery energy of a node [1].

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

II. RELATED WORK

A. Vulnerabilities in Web Services

In the Service Oriented Architecture (SOA) implementation, Web Services are in constant communication with other services. Their clients make requests for services through of a communication channel such as the Internet, sending and receiving information simultaneously. Another benefit is the possibility to develop web services in different languages and platforms. This technology transmits their information using two protocols, XML and HTML.

In [3], the author defines the main challenges related to standards and interoperability in Web Services. This research emphasizes the relative immaturity of this technology on security threats, quality of service (QoS), and scalability, among others. In [8], the authors classify the security challenges involving threats, attacks and security problems in this technology. We describe them as follows:

- Services level threats: attacks against WSDL and UDDI, injection of malicious code, phishing, denial of service, spoofing XML schemas and kidnapping/stealing session.
- Message level threats: Injection attacks, forwarding messages, attacks of message validation, interception and loss of message confidentiality.

B. Vulnerabilities Detection Technique

Following the best practices of software testing and standards, there have been developed a lot of tools, languages and techniques in order to analyse and detect vulnerabilities in systems [2]. The security validation for Web Services can be performed in two phases, static and dynamic phase. The static phase tries to find faults inserted during the development phase introduced in the code by possible human errors in the project stage. This phase is analysed as a state not reachable, i.e. it can always be found new faults. In this case, the methods used are Static Analyse (code inspection, static vulnerability analysis) or Theorem Proof, which donor need to run the system. These methods are early detection and carry many benefits such as reduced cost of testing. On the other hand, the dynamic phase focuses on verification of the system during its running, i.e. the code of the system is tested with real entries to verify security mechanisms at runtime.

The Security Testing is applied in this phase. This test looks for vulnerabilities in web applications by sending attack within request message. Among these security techniques, the Penetration Testing and Fault Injection are among them.

C. Fault Injection attack

In software testing, fault injection is a technique for improving the coverage of a test by introducing faults to test code paths, in particular error handling code paths that might otherwise rarely be followed. It is often used with stress testing and is widely considered to be an important part of developing robust software. Robustness testing (also known as Syntax Testing, Fizzing or Fuzz testing) is type of fault injection commonly used to test for vulnerabilities in communication interfaces such as protocols, command line parameters, or APIs.

The propagation of a fault through to an observable failure follows a well-defined cycle. When executed, a fault may cause an error, which is an invalid state within a system boundary. An error may cause further errors within the system boundary, therefore each new error acts as a fault, or it may propagate to the system boundary and be observable. When error states are observed at the system boundary they are termed failures. This mechanism is termed the fault-error failure cycle and is a key mechanism in dependability.

III. TESTING METHODOLOGY

One of the challenges to find vulnerabilities in web Services during the implementation phase is determine which attacks scenarios are appropriate to test for [1]. These scenarios can be obtained from various sources such as Internet, books and papers. However, it is hard to find and set up a database with relevant attacks and automating them according to the testing environment. Our purpose in this section is to use, in part, the Security Testing Methodology [9]. In the following sub-sections, author briefly describe the results of each phase of Security Testing Methodology for Vulnerabilities Detection of XSS in Web Services the implementation of the Security Testing Methodology with XSS. This attack is emulated with WSInject and soapUI [1].

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

A. Identification of the Attacker Objectives:

The Web Service attacker aims to find vulnerabilities using different kinds of methods (e.g. Denial-of-Services, spoofing, injection attack, and man-in-the-middle, among others). In this research, the goal of the attacker is focused on finding vulnerabilities in servers that work with Web Services. The attacker intercepts SOAP messages between the client and the server. His goal is perform unauthorized operations (Violation of integrity) or escalate privileges (violation of control access). For that, he tries to inject various XML tags in order to modify the XML message structure and generate faults in the server.

B. Attacks Modelling:

In this step, author use the SecurITree version 3.4 in order to model XSS attack. This tool, used in several researches helped us to design the attack tree for injecting vulnerabilities inWeb Services. The attack tree was built and structured accordingly to the proposed steps in [1], composed of the following attributes:

i) attacker capability; ii) possibility of emulating the attack by a fault injection tool; iii) the requirements of the attack to be run in the Web Service; and iv) the verification if the WSSecurity protects the Web Services from XSS attack. These four attributes were used to classify the Injection attacks with boolean values, namely < Possible; Impossible >. The output is the creation of the attack tree, which is used by the attacker to look for vulnerabilities in the Web Services. Sample attack tree for XML injection scenarios is described in Fig 1.

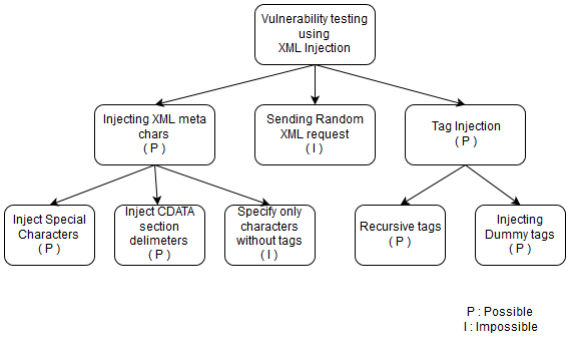

Fig 1. Attack Scenario Tree

C. Attack Scenarios Generation

At this stage, the attack scenarios are produced automatically according to the criteria defined in step 3). The output of this step is the attack scenarios described in the same format of the tree leaves, each one representing the description of an attack. The scenarios can be used to create a useful and reusable library of attacks to test protocols.

D. Attack Scenarios Implementation

The attack scenarios, generated in step C), are described in text notation, i.e. at the same level of the attack tree abstraction. This type of description is useful for testing analysts and security experts due to their easy configuration, but not to be processed by an injection tool. In this stage, the analysts must perform a set of refinement steps in order to transform the text notation into executable script by WSInject tool.

IV. PROPOSED SYSTEM

It is necessary to have flexible testing approaches that allow creating multiple scenarios, which could help to determine robustness of the web services. There are several ways to analyse the existence of vulnerabilities in SOA (Service Oriented Architecture), e.g. compare server responses in the presence of attacks and absence of them, sensitive information exposure, XML schema modification request, among others. This step is crucial to reduce the number of false positives or false negatives.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

A. *Proposed System Architecture:*

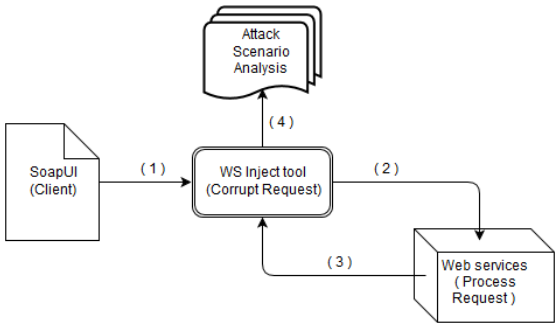

Fig 2. System Architecture

The given architecture provides overview of the request flow and it's use for making various attacks on the target service so that WS Inject tool can analyse it's behaviour. This will help to the more robustness in decision to analyse vulnerability in the target service.

Surveyed approach uses the HTTP status-code in the server response, which describes the behaviour of the Web Service in a not robust environment. For example, when the request is processed by Web Services without detecting the attack, i.e. not generated a message describing the existence of error in the request, it allows to identify the existence of a possible vulnerability found with code 200 OK. If a code 400 Bad Request is received, it can be considered as robust response because the server detected the XSS attack.

Further analysis of the response can be performed by loading given response as client loading it. This could provide threatened impacts, on execution of the same like XSS attacks. Same HTTP response can be observed for different behavioural responses; hence for deriving robustness properties of target web service it is important to carefully analyse the response as well.

B. *Proposed System Algorithm:*

Step 1: Capture the message sent from client (e.g. Soap-UI).

Step 2: Identify the request and modify the content of the request to corrupt it, for given attacking scenario. (Using tool such as WS-Inject)

Step 3: Obtain response from server, observe the impact due to malformed request sent in step 2. This could be analysed with help of HTTP status code, as mentioned in section B (1).

Step 4:

If I. Given attack does not show any vulnerability in targeted environment then modify the test case,

Go to Step 2;

Else

I. Report the bug in the web service with given attack scenario.

Step 5: Stop.

C. *Proposed System Algorithm rules:*

The UDDI Business Registry (UBR) is a single registry for Web Services through its WSDL. The WSDL file allows us to know how the service can be called, what parameters it expects, and what data structures it returns. The fault injection can be performed with help of WSInject which is configured between client and server. This can be configured to WSDL of the service through proxy. Using this one can corrupt the request sent through the client (e.g. SoapUI) and observe its response [1]. This enables to conclude whether targeted system has vulnerabilities in it or not. The difference in behavior can be observed by varying workload and fault load. The evaluation can be performed with help of 8 rules developed as mentioned below [1]:

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

Rule 1. If the message header contains the code 200 OK AND the server ran the SOAP message with any attack, THEN it could be a Vulnerability Found (VF) in the Web Service. OTHERWISE, if the SOAP message describes the existence of a syntax error or warning about the presence of any attack, THEN there is No Vulnerability Found (NVF).

Rule 2. If the message header contains the code 400 Bad request message, e.g. request format is invalid: missing required soap: Body element, THEN there is No Vulnerability Found (NVF).

Rule 3. If the message header contains the code 500 Internal Server Error AND there are information disclosure in the SOAP message (e.g. it shows path directory, function library and object information, user-names and passwords, database access, among others), THEN there is a Vulnerability Found (VF), OTHERWISE there is No Vulnerability Found (NVF).

Rule 4. i) If in the absence of any attack, the message header contains the code 500 Internal Server Error AND there are information disclosure in the SOAP message; AND ii) if in the presence of any attack, the header contains the code HTTP 200 OK, THEN there is a Vulnerability Found (VF).

Rule 5. i) If in the absence of any attack, the header contains the code 500 Internal Server Error AND there are information disclosure in the SOAP message; AND ii) if in the presence of any attack, the header contains the code 400 Bad request message, THEN there is a Vulnerability Found (VF).
Rule 6. i) If in the absence of any attack, the message header contains the code 500 Internal Server Error AND there are information disclosure in the SOAP message; AND ii) if in the presence of any attack, the header contains the code 500 Internal Server Error too, THEN there is a Vulnerability Found (VF).

Rule 7. If the server does not respond, it is considered as crash, THEN the result is considered Inconclusive, because one cannot guarantee that the error was caused by the attack.

Rule 8. If none of the rules above may be applied, THEN the result is considered Inconclusive, because there is no way to confirm if there really were vulnerabilities in the Web Service.
After performing various attacks made on selected web service, with security tokens (i.e. UsernameToken) and without security token results observed as, the use of UsernameTokens reduces the percentage of Web Services vulnerabilities from 92% to 72% but not enough. This happens because these services use password to authenticate the user. However, these techniques do not protect the integrity of SOAP message and its confidentiality.
XML-Enc or XML-Sig may be used to provide confidentiality and integrity of both the UsernameToken and the entire message body.
Furthermore, the use of this specification forces the programmers to create more robust Web Services. i.e. use message timestamps, nonces, and caching, the password should be digested for protect against eavesdropping attacks, as well as other application-specific tracking mechanisms.

V. MATHEMATICAL MODEL

Let S be the whole system S= I, P, O I-Input P-Procedure O-Output.

Input I=REQ, MEC, S-Token

Where,
REQ- Request that could be obtained from client (e.g. soap-UI),
MEC - Request which is obtained in the form of string could be corrupted with certain mechanism suitable to attack scenario,
S-Token - Security tokens may needs to be supplied with request to improve authentication.

Procedure (P), The REQ needs to be analyzed in the system, which could be corrupted by applying MEC suits to given attack scenario, forward the REQ to the web service and wait for the response.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

As mentioned in the algorithm steps, analyze the response message header information. It usually contains HTTP status code. Further security can be enhanced with S-Token mechanism add-on to the REQ.

A. Module Description:

1. **Fault injection:** To test the robustness of the service, multiple attack scenarios needs to be performed by injecting the malformed code to the REQ made, through clients.
2. **Attack scenario:** various test cases can be generated to test the robustness of the service by varying the attack scenarios previously made. There are set of parameters need to pass to the service to perform desired operation, its value or parameter itself can be modified to approach certain corrupting MEC.
3. **Response analysis:** The malformed request which has been sent to the service may provide in turn response from it. Its header information (e.g. HTTP status) could help to analyze the response.
4. **Test case decision:** For the given attack scenario based on the response observed, one can conclude whether system is vulnerable to the given attack or not.

VI. COMPARISON RESULT

Following table shows the comparative study of different schemes based on different parameters. From below Tables shows that, the schemes which are based on the way response analysis provided properties like are better than any other schemes. The proposed system has tried to meet the essential analysis requirements and provide the functionality as compared to other schemes.

1. Attack scenarios
   i. Multiplicity: Performing predefined attacks from the system.
   ii. Request change control: The Request could be updated to accomplish custom attack scenario which was not there in the predefined list.
   iii. Group of attack technique: Instead of running each test case manually one by one attack can done over the set of scenarios mentioned in the script.
2. Content Analysis
   i. Iteration: Parsing over the response content and reach up to the depth level of analysis to understand the thread.
   ii. Expression: Directly applying set of patterns to scan the response and check whether certain kind of attack has been performed or not.
3. Header Analysis
   When server process any request it provide header information what is the impact of given request processing at server end.
   i. Status code: Status code from the response provides us additional hint over the successful execution at server end.

Comparison	Requirement	[1]	Proposed Work
Attack Scenarios	Multiplicity	No	Yes
	Request Change Control	Yes	Yes
	Group of attack technique	Yes	Yes
Content Analysis	Iteration	No	Yes
	Expression	No	Yes
Header Analysis	Status code	Yes	Yes

Table 1. Comparison Result

VII. CONCLUSION AND FUTURE WORK

This paper provides survey on new approach to analyze the robustness of Web Services by Fault Injection with WSInject. This tool allows emulation and generation of attacks, however, the process is delayed and often not automated. This survey focused on emulation of Cross-site Scripting (XSS) attack. This is a fairly frequent attack, according to the research cited, whose effects can be quite devastating for servers and users of Web Services.

The results of the Penetration Testing phase help to develop the rules for vulnerabilities analysis.

However, the results obtained by soapUI show a large percentage of false positives and false negatives. One advantage of the proposed approach is that it relies on the use of a fault injector of general purpose, which can be used

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

to emulate several types of attacks and may generate variants of the same, which is usually limited in the tools commonly used for security testing, as the vulnerabilities scanners.

REFERENCES

1. Marcelo Invert Palma Salas, Paulo Lcio de Geus, Eliane Martinsl, "Security Testing Methodology for Evaluation of Web Services Robustness - Case: XML Injection", Institute of Computing, UNICAMP, Campinas, Brazil, 2015.
2. Cachin, C., and J. Camenisch, "Malicious and Accidental-Fault Tolerance in Internet Applications: Reference Model and Use Cases", LAAS, MAFTIA, 2000.
3. Holgersson, J., and E. Soderstrom, "Web Service Security-Vulnerabilities and Threats within the Context of WS Security", SIIT 2005, ITU.
4. Williams J., and D.Wichers, OWASP Top 10, OWASP Foundation,2010,URL: https://www.owasp.org/.
5. Meiko J., G. Nils, H. Ralph, "A Survey of Attacks on Web Services, Computer Science - Research and Development", Heidelberg; ISSN: 1865-2034,volume 24, Edio 4. 2009.
6. Dolev D., A. Yao, On the "Security of Public Key Protocols, In IEEE Transactions on Information Theory", IEEE Computer Society Press, Mar 1983.
7. Morais A, and E. Martins, "Injeo de Ataques Baseados em Modelo para Teste de Protocolos de Segurana", Thesis (Master in Computer Science), Institute of Computing, UNICAMP, State University of Campinas, Brazil, 15, May 2009.
8. Ladan MI, "Web services: Security Challenges, in Proceedings of the World Congress on Internet Security", 2011, WorldCIS11, IEEE Press, Londres, Reino Unido, 21-23, Feb 2011.
9. Eastlake D. , "XML Signature Syntax and Processing", W3C Recommendation, 2002.
10. Zhao G., W. Zheng, J. Zhao, and H. Chen, "An Heuristic Method for Web-Service Program Security Testing", In Proceedings of the 2009 Fourth ChinaGrid Annual Conference, CHINAGRID '09, IEEE Computer Society Press, Yantai, China, 21-22, Aug 2009.
11. Cristian F., H. Aghili, R. Strong, and D. Volev, "Atomic Broadcast: From Simple Message Diffusion to Byzabtube Agreement", In Proceedings of the Twenty-Fifth International Symposium on Fault-Tolerant Computing, IEEE Computer Society Press, Pasadena-CA, USA, 27-30, Jun 1995.
12. Hsueh MC, TK Tsai, and RK Iyer , "Fault Injection Techniques and Tools" , IEEE Computer Society Press, Computer; Volume 30, Ed. 4, Apr 1997.

BIOGRAPHY

Mr. Jaydeep J. Mangle is pursuing his Masters of Engineering in the Computer Science Department, RMD SSOE College, Savitribai Phule University, Pune. He has received Bachelor of Engineering degree in Computer Science from University Of Mumbai, Ratnagiri, India.

Prof. Vina M. Lomte is the HOD of Computer Dept. at RMD SSOE College, Pune, having more than 10+ years of experience in the field of teaching and research. The domains of her research are Software Testing, Software Engineering and Web Security.