International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 6 1209– 1213

Implementation Ids for Web Security Mechanism against Injection and Multiple Attacks

Pranali Mankar
Department of Computer Science and Engineering
TGPCET Nagpur

Prof. Jayant Adhikari
Department of Computer Science and Engineering
TGPCET Nagpur

Abstract:- In this paper we propose a philosophy and a model apparatus to assess web application security instruments. The approach is in view of the thought that infusing sensible Vulnerabilities in a web application and assaulting them naturally can be utilized to bolster the evaluation of existing security systems and apparatuses in custom setup situations. The investigations incorporate the assessment of scope and bogus positives of an interruption recognition framework for SQL Injection assaults and the viability's evaluation of two top business web application defenselessness scanners. Results demonstrate that the infusion of vulnerabilities and assaults is to be sure a viable approach to assess security components and to bring up their shortcomings as well as courses for their change.

Keywords: *SQL Injection, XSS, VAIT.*

*****

I. Introduction

Nowadays there is an increasing dependency on web applications, ranging from individuals to large organizations. Almost everything is stored, available or traded on the web. Web applications can be personal websites, blogs, news, social networks, web mails, bank agencies, forums, e-commerce applications, etc. The omnipresence of web applications in our way of life and in our economy is so important that it makes them a natural target for malicious minds that want to exploit this new streak.

We need means to evaluate the security of web applications and of attack counter measure tools. To handle web application security, new tools need to be developed, and procedures and regulations must be improved, redesigned or invented. Moreover, everyone involved in the development process should be trained properly. All web applications should be thoroughly evaluated, verified and validated before going into production.

II. Literature Review

[1] In this paper they propose a methodology and a prototype tool to evaluate web application security mechanisms. The methodology is based on the idea that injecting realistic vulnerabilities in a web application and attacking them automatically can be used to support the assessment of existing security mechanisms and tools in custom setup scenarios. To provide true to life results, the proposed vulnerability and attack injection methodology relies on the study of a large number of vulnerabilities in real web applications. In addition to the generic methodology, the paper scribes the implementation of the Vulnerability & Attack Injector Tool (VAIT) that allows the automation of the entire process. The drawback of this paper is methods are more complicated and less efficient.

[2]In this methodology has been used to extend a debugging tool aimed at testing fault tolerance protocols developed by BULL France. It has been applied successfully to the injection of faults in the inter-replica protocol that supports the application-level fault tolerance features of the architecture of the ESPRIT-funded Delta4project. The results of these experiments are analyzed in detail [2].

[3]The paper describes a dependability evaluation method based on fault injection that establishes the link between the experimental evaluation of the fault tolerance process and the fault occurrence process. The main characteristics of a fault injection test sequence aimed at evaluating the coverage of the fault tolerance process are presented. The various steps by which the fault occurrence and fault tolerance processes are combined to evaluate dependability measures are identified and their interactions are analyzed [3].

[4]In this paper, due to our increasing reliance on computer systems, security incidents and their causes are important problems that need to be addressed. To contribute to this objective, the paper describes a new tool for the discovery of security vulnerabilities on network connected servers. The AJECT tool uses a specification of the server's communication protocol to automatically generate a large number of attacks accordingly to some predefined test classes. Then, while it performs these attacks through the

network, it monitors the behavior of the server both from a client perspective and inside the target machine[4].

III. Proposed System

To give consistent with life comes about, the proposed helplessness and assault infusion procedure depends on the investigation of an expansive number of vulnerabilities in genuine web applications. Notwithstanding the non-specific approach, the paper portrays the Vulnerability's usage & Attack Injector Tool (VAIT) that permits the whole robotization process. We utilized this instrument to run an arrangement of trials that exhibit the attainability and the viability of the proposed procedure.

The tool will test on top of widely used applications in two scenarios. The first to evaluate the effectiveness of the VAIT in generating a large number of realistic vulnerabilities for the offline assessment of security tools, in particular web application vulnerability scanners. The second to show how it can exploit injected vulnerabilities to launch attacks, allowing the online evaluation of the effectiveness of the counter measure mechanisms installed in the target system, in particular an intrusion detection system.

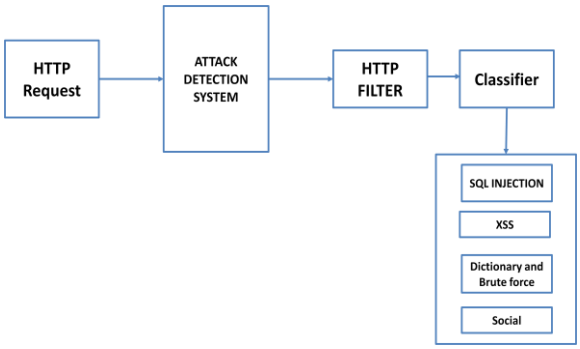

Fig1. System Architecture

Http Request:

Request-Line

The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by space SP characters.

Request Method

The request method indicates the method to be performed on the resource identified by the given Request-URI. The method is case-sensitive and should always be mentioned in uppercase. The following table lists all the supported methods in HTTP/1.1.

Request-URI

The Request-URI is a Uniform Resource Identifier and identifies the resource upon which to apply the request. Following are the most commonly used forms to specify an URI.

Request Header Fields

We will study General-header and Entity-header in a separate chapter when we will learn HTTP header fields.

Attack detection system:

Intrusion detection is the act of detecting unwanted traffic on a network or a device. A intrusion detection system (IDS) provides a layer of defense which monitors network traffic for predefined suspicious activity or patterns, and alert system administrators when potential hostile traffic is detected.

Http Filter:

A filter is an object that is invoked at the preprocessing and postprocessing of a request. It is mainly used to perform filtering tasks such as conversion, logging, compression, encryption and decryption, input validation etc.

Attack detection system:

Intrusion detection is the act of detecting unwanted traffic on a network or a device. A intrusion detection system (IDS) provides a layer of defense which monitors network traffic for predefined suspicious activity or patterns, and alert system administrators when potential hostile traffic is detected.

Intrusion detection faces a number of challenges; an intrusion detection system must reliably detect malicious activities in a network and must perform efficiently to cope with the large amount of network traffic.Network based intrusion detection are the most deployed IDS. An IDS can be a piece of installed software or a physical appliance. Many IDS tools will also store a detected event in a log to be reviewed at a later date or will combine events with other data to make decisions regarding policies or damage control. This paper discusses the various types of attacks that can be detected in a simulated network environment. The different types of attacks are Probe attacks, R2L, Dos and U2R attacks.

Several types of IDS technologies exist due to the variance of network configurations. Each type has advantages and disadvantage in detection, configuration, and cost. NIDS (Network Intrusion Detection Systems)

Network Intrusion Detection Systems are placed at a strategic point or points within the network to monitor traffic to and from all devices on the network. Ideally one would scan all inbound and outbound traffic. NIDS analyzes network traffic at all layers of the Open Systems Interconnection (OSI) model and makes decisions about the purpose of the traffic, analyzing for suspicious activity. Most NIDSs are easy to deploy on a network and can often view traffic from many systems at once. HIDS (Host Intrusion Detection Systems) Host Intrusion Detection Systems are run on individual hosts or devices on the network. A HIDS monitors the inbound and outbound packets from the device only and will alert the user or administrator if suspicious activity is detected. HIDS analyze network traffic and system-specific settings such as software calls, local security policy, local log audits, and more. A HIDS must be installed on each machine and requires configuration specific to that operating system.

Classifier:

A classifier sometimes called a counter word is a word or affix that is used to accompany nouns and can be considered to "classify" the noun depending on the type of its referent.

In practice, the use of both static and dynamic analysis is a key feature of the methodology that allows increasing the overall performance and effectiveness, as it provides the means to inject more vulnerability that can be successfully attacked and discarded those that cannot.

The proposed methodology provides a practical environment that can be used to test countermeasure mechanisms (such as intrusion detection systems (IDSs), web application vulnerability scanners, web application fire-walls, static code analyzers, etc.), train and evaluate security teams, help estimate security measures (like the number of vulnerabilities present in the code), among others.

SQL INJECTION ATTACK

SQL injection is a code injection technique, used to attack data-driven applications, in which nefarious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker). SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server. This form of SQL injection occurs when user input is not filtered for escape characters and is then passed into an SQL statement. This results in the potential manipulation of the statements performed on the database by the end-user of the application.

The following line of code illustrates this vulnerability:

statement = "SELECT * FROM users WHERE name = '" + userName + "';"

This SQL code is designed to pull up the records of the specified username from its table of users. However, if the "userName" variable is crafted in a specific way by a malicious user, the SQL statement may do more than the code author intended. For example, setting the "userName" variable as:

' OR '1'='1

or using comments to even block the rest of the query (there are three types of SQL comments). All three lines have a space at the end:

' OR '1'='1' --

' OR '1'='1' ({

' OR '1'='1' /*

renders one of the following SQL statements by the parent language:

SELECT * FROM users WHERE name = '' OR '1'='1';

SELECT * FROM users WHERE name = '' OR '1'='1' -- ';

If this code were to be used in an authentication procedure then this example could be used to force the selection of every data field (*) from all users rather than from one specific user name as the coder intended, because the evaluation of '1'='1' is always true (short-circuit evaluation).

CROSS-SITE SCRIPTING (XSS) ATTACK

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in web applications. XSS enables attackers to inject client-side scripts into web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access

controls such as the same-origin policy. Cross-site scripting carried out on websites accounted for roughly 84% of all security vulnerabilities documented by Symantec as of 2007. Their effect may range from a petty nuisance to a significant security risk, depending on the sensitivity of the data handled by the vulnerable site and the nature of any security mitigation implemented by the site's owner. Security on the web depends on a variety of mechanisms, including an underlying concept of trust known as the same-origin policy.

Cross-site scripting attacks use known vulnerabilities in web-based applications, their servers, or the plug-in systems on which they rely. Exploiting one of these, attackers fold malicious content into the content being delivered from the compromised site. When the resulting combined content arrives at the client-side web browser, it has all been delivered from the trusted source, and thus operates under the permissions granted to that system. By finding ways of injecting malicious scripts into web pages, an attacker can gain elevated access-privileges to sensitive page content, to session cookies, and to a variety of other information maintained by the browser on behalf of the user. Cross-site scripting attacks represent a special case of code injection.

Microsoft security-engineers introduced the term "cross-site scripting" in January 2000. The expression "cross-site scripting" originally referred to the act of loading the attacked, third-party web application from an unrelated attack-site, in a manner that executes a fragment of JavaScript prepared by the attacker in the security context of the targeted domain (taking advantage of a reflected or non-persistent XSS vulnerability). The definition gradually expanded to encompass other modes of code injection, including persistent and non-JavaScript vectors (including ActiveX, Java, VBScript, Flash, or even HTML scripts), causing some confusion to newcomers to the field of information security.

IV. Experimental Results

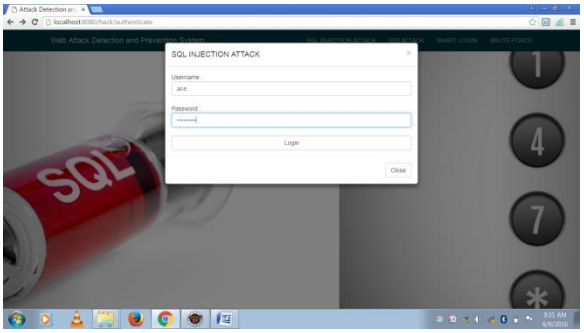

Fig 1. SQL Injection Login Page

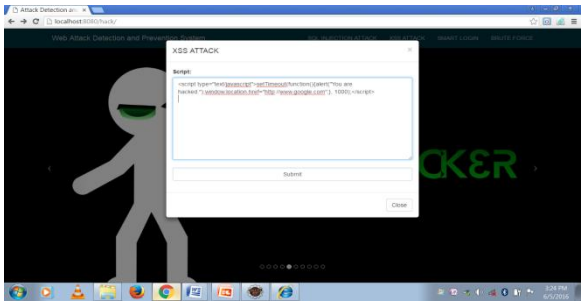

Fig 2. Xss Attack log in page window

V. Conclusion

The SQL - Injection Attacks are tremendously dangerous in association to other types of Web-based attacks, for the reason that here the end result is data manipulation.SQL injection holes can be easily exploited by a technique called SQL Injection Attacks. This proposed integrated approach is an effort to add some more security measures to databases to avoid SQL injection attack. we will try to improve the technique by making it fully secure and efficient for other types of SQL injection attacks also. Then, this technique will be able to prevent SQL Injection Attacks completely.

References

[1] Jose Fonseca, Marco Vieira, and Henrique Madeira “Evaluation of Web Security MechanismsUsing Vulnerability & Attack Injection”-IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 11, NO. 5, SEPTEMBER/OCTOBER 2014

[2] D. Avresky, J. Arlat, J.C. Laprie, and Y. Crouzet, “Fault Injection for Formal Testing of Fault Tolerance,” IEEE Trans. Reliability, vol. 45, no. 3, pp. 443-455, Sept. 2011\

[3] J. Arlat, A. Costes, Y. Crouzet, J.-C.Laprie, and D. Powell, “Fault Injection and Dependability Evaluation of Fault-Tolerant Systems,” IEEE Trans. Computers, vol. 42, no. 8, pp. 913-923, Aug. 2011.

[4] N. Neves, J. Antunes, M. Correia, P. Ver_ıssimo, and R. Neves, “Using Attack Injection to Discover New Vulnerabilities,” Proc. IEEE/IFIP Int’l Conf. Dependable Systems and Networks, 2006.

[5] N. Jovanovic, C. Kruegel, and E. Kirda, “Precise Alias Analysis for Static Detection of Web Application Vulnerabilities,” Proc. IEEE Symp. Security Privacy, 2006.

[6] IBM Global Technology Services “IBM Internet Security Systems X-Force 2012 Trend & Risk Report,” IBM Corp., Mar. 2013.

[7] J. Fonseca and M. Vieira, “Mapping Software Faults with Web Security Vulnerabilities,” Proc. IEEE/IFIP Int’l. Conf. Dependable Systems and Networks, June 2008

[8] J. Fonseca, M. Vieira, and H. Madeira, “Training Security Assurance Teams using Vulnerability

Injection," Proc. IEEE Pacific Rim Dependable Computing Conf., Dec. 2008.

[9] J. Carreira, H. Madeira, and J.G. Silva, "Xception: Software Fault Injection and Monitoring in Processor Functional Units," IEEE Trans. Software Eng., vol. 24, no. 2, Feb. 1998.

[10] D.T. Stott, B. Floering, D. Burke, Z. Kalbarczpk, and R.K. Iyer, "NFTAPE: A Framework for Assessing Dependability in Distributed Systems with Lightweight Fault Injectors," Proc. Computer Performance and Dependability Symp., 2000.

[11] J. Christmansson and R. Chillarege, "Generation of an Error Set that Emulates Software Faults," Proc. IEEE Fault Tolerant Computing Symp., 1996.

[12] H Madeira, M. Vieira, and D. Costa, "On the Emulation of Software Faults by Software Fault Injection," Proc. IEEE/IFIP Int'l Conf. Dependable System and Networks, 2000.

[13] J. Fonseca, M. Vieira, and H. Madeira, "Testing and Comparing Web Vulnerability Scanning Tools for SQLi and XSS Attacks," Proc. IEEE Pacific Rim Int'l Symp. Dependable Computing, Dec. 2007.

[14] J. Dur~aes and H. Madeira, "Emulation of Software Faults: A FieldData Study and a Practical Approach," IEEE Trans. Software Eng., vol. 32, no. 11, pp. 849-867, Nov. 2006.

[15] Ananta Security "Web Vulnerability Scanners Comparison,"anantasec.blogspot.com/2009/01/web-vulnerability-scannerscomparison.html, accessed 1 May 2013, 2009.

[16] J. Fonseca, M. Vieira, and H. Madeira,"The Web Attacker Perspective- A Field Study," Proc. IEEE Int'l. Symp. Software Reliability Eng., Nov. 2010

[17] [17] G. Buehrer, B. Weide, and P. Sivilotti, "Using Parse Tree Validation to Prevent SQLi Attacks," Proc. Int'l Workshop Software Eng. and Middleware, 2005

[18] [18] I. Elia, J. Fonseca, and M. Vieira, "Comparing SQLi Detection Tools Using Attack Injection: An Experimental Study," Proc. IEEE Int'l Symp. Software Reliability Eng., Nov. 2010.

[19] [19] M. Buchler, J. Oudinet, and A. Pretschner, "Semi-Automatic Security Testing of Web Applications from a Secure Model," Proc. Int'l Conf. Software Security and Reliability, 2012.

[20] [20] Y.-W. Huang, S.-K.Huang, T.-P.Lin, and C.-H. Tsai, "Web Application

[21] Security Assessment by Fault Injection and Behavior Monitoring," Proc. Int'l Conf. World Wide Web, pp. 148-159, 2003.

[22] [21] J. Fonseca, M. Vieira, and H. Madeira, "Detecting Malicious SQL,"

[23] Proc. Conf. Trust, Privacy & Security in Digital Business, Sept. 2007