International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 10, October 2015
Available at http://internationaljournalofresearch.org

Designing Ids for Web Security Mechanism against Injection and Multiple Attacks

Ms. Aboli; Prof. D. M. Sable & Prof V.R. Wadhankar
Computer Science and Engineering, RTMNU University, ACE Wardha, Maharashtra, India

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. 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's robotization process. We utilized this instrument to run an arrangement of trials that exhibit the attainability and the viability of the proposed procedure. 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: SL Injection; XSS; VAIT

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.

Conceptually, the attack injection consists of the introduction of realistic vulnerabilities that are afterwards automatically exploited (attacked). Vulnerabilities are considered realistic because they are derived from the extensive field study on real web application vulnerabilities presented in [16], and are injected according to a set of representative restrictions and rules defined in [17].

The attack injection methodology is based on the dynamic analysis of information obtained from the runtime monitoring of the web application behavior and of the interaction with

external resources, such as the backend database. This information, complemented with the static analysis of the source code of the application, allows the effective injection of vulnerabilities that are similar to those found in the real world.

Although this methodology can be applied to various types of vulnerabilities, we focus on of the most widely exploited and serious web application vulnerabilities that are SQL Injection (SQLi) and Cross Site Scripting (XSS) [3], [6]. Attacks to these vulnerabilities basically take advantage of improper coded applications due to unchecked input fields at user interface. This allows the attacker to change the SQL commands that are sent to the database (SQLi) or through the input of HTML and scripting languages (XSS).

A Brute-Force Attack, or exhaustive key search, is a cryptanalytic attack that can, in theory, be used against any encrypted data (except for data encrypted in an information-theoretically secure manner). Such an attack might be used when it is not possible to take advantage of other weaknesses in an encryption system (if any exist) that would make the task easier. It consists of systematically checking all possible keys or passwords until the correct one is found. In the worst case, this would involve traversing the entire search space. When password guessing, this method is very fast when used to check all short passwords, but for longer passwords other methods such as the dictionary attack are used because of the time a brute-force search takes. The resources required for a brute-force attack grow exponentially with increasing key size, not linearly. Although US export regulations historically restricted key lengths to 56-bit symmetric keys (e.g. Data Encryption Standard), these restrictions are no longer in place, so modern symmetric algorithms typically use computationally stronger 128- to 256-bit keys. There is a physical argument that a 128-bit symmetric key is computationally secure against brute-force attack.

Literature Review

[1]Jose Fonseca, Marco Vieira, and Henrique Madeira "Evaluation of Web Security Mechanisms Using Vulnerability And Attack Injection" Transactions On Dependable And Secure Computing, Vol. 11, No. 5, September/October 2014.

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 [1].

[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

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]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

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. Emphasis is given to the derivation of experimental measures. 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] 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.

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. The observation of an incorrect behavior indicates a successful attack and the potential existence of a vulnerability. To demonstrate the usefulness of this approach, a considerable number of experiments were carried out with several IMAP servers[4].

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

In this methodology, the number and the importance of web applications have increased rapidly over the last years. At the same time, the quantity and impact of security vulnerabilities in such applications have grown as well. Since manual code reviews are time-consuming, error prone and costly, the need for automated solutions has become evident. In this paper, we address the problem of vulnerable web applications by means of static source code analysis. To this end, we present a novel, precise alias analysis targeted at the unique reference semantics commonly found in scripting languages. Moreover, we enhance the quality and quantity of the generated vulnerability reports by employing a novel, iterative two-phase algorithm for fast and precise resolution of file inclusions [5].

[6] J. Fonseca and M. Vieira, "Mapping Software Faults with WebSecurity Vulnerabilities," Proc. IEEE/IFIP Int'l. Conf. DependableSystems and Networks, June 2008.

Web applications are typically developed with hard time constraints and are often deployed with critical software bugs, making them

vulnerable to attacks. The classification and knowledge of the typical software bugs that lead to security vulnerabilities is of utmost importance. This paper presents a field study analyzing 655 security patches of six widely used web applications. Results are compared against other field studies on general software faults (i.e., faults not specifically related to security), showing that only a small subset of software fault types is related to security. Furthermore, the detailed analysis of the code of the patches has shown that web application vulnerabilities result from software bugs affecting a restricted collection of statements. A detailed analysis of the conditions/locations where each fault was observed in our field study is presented allowing future definition of realistic fault models that cause security vulnerabilities in web applications, which is the key element to design a realistic attack injector.

[7] J. Fonseca, M. Vieira, and H. Madeira, "Training Security Assurance
Teams using Vulnerability Injection," Proc. IEEE Pacific RimDependable Computing Conf., Dec. 2008.

Writing secure Web applications is a complex task. In fact, a vast majority of Web applications are likely to have security vulnerabilities that can be exploited using simple tools like a common Web browser. This represents a great danger as the attacks may have disastrous consequences to organizations, harming their assets and reputation. To mitigate these vulnerabilities, security code inspections and penetration tests must be conducted by well-trained teams during the development of the application. However, effective code inspections and testing takes time and cost a lot of money, even before any business revenue. Furthermore, software quality assurance teams typically lack the knowledge required to effectively detect security problems. In this paper we propose an approach to quickly and effectively train security assurance teams in the context of web application development. The approach combines a novel vulnerability injection technique with relevant guidance information about the most common security vulnerabilities to provide a realistic training scenario. Our experimental results show that a short training period is sufficient to clearly improve the ability of security assurance teams to detect vulnerabilities during both code inspections and penetration tests.

[8] J. Carreira, H. Madeira, and J.G. Silva, "Xception: Software FaultInjection and Monitoring in Processor Functional Units," IEEETrans. Software Eng., vol. 24, no. 2, Feb. 1998.

This paper presents Xception, a software faultinjection and monitoring environment. Xception usesthe advanced debugging and performance monitoringfeatures existing in most of the modern processors toinject more realistic faults by software, and tomonitor the activation of the faults and their impacton the target system behaviour in detail. Faults areinjected with minimum interference with the targetapplication. The target application is not modified,no software traps are inserted, and it is not necessaryto execute it in special trace mode (the application isexecuted at full speed). Xception provides acomprehensive set of fault triggers, including spatialand temporal fault triggers, and triggers related tothe manipulation of data in memory. Faults injectedby Xception can affect any process running on the starget system including the operating system.

[9] 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.

Many fault injection tools are available for dependability assessment. Although these tools are good at injecting
a single fault model into a single system, they suffer from two main limitations for use in distributed systems: (1) nosingle tool is sufficient for injecting all necessary fault models; (2) it is difficult to port these tools to new systems.NFTAPE, a tool for composing automated fault injection experiments from available lightweight fault injectors,triggers, monitors, and other components, helps to solve these problems.
We have conducted experiments using NFTAPE with several types of lightweight fault injectors, includingdriver-based, debugger-based, target-specific, simulation-based, hardware-based, and performance-fault injections.Two example experiments are described in this paper. The first uses a hardware fault injector with aMyrinet LAN; the other uses a Software Implemented Fault Injection (SWIFI) fault injector to target a spaceimagingapplication.

[10] J. Christmansson and R. Chillarege, "Generation of an Error Setthat Emulates Software Faults," Proc. IEEE Fault Tolerant ComputingSymp., 1996.

A significant issue in fault injection experiments is thatthe injected faults are representative of software faultsobserved in the field. Another important issue is the time
used, as we want experiments to be conducted withoutexcessive time spent waiting for the consequences of afault. An approach to accelerate the failure process wouldbe to inject errors instead of faults, but this would requirea mapping between representative software faults andinjectable errors. Furthermore, it must be assured that theinjected errors emulate software faults and not hardwarefaults.

Proposed system

The methodology proposed was implemented in a concrete Vulnerability & Attack Injector Tool (VAIT) for web applications. The tool was tested 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.

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.

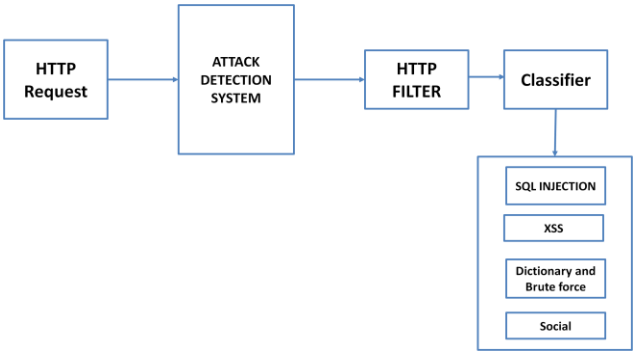

Fig1. System Architecture

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.

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 Field
Data 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] 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] 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] 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] Y.-W. Huang, S.-K. Huang, T.-P. Lin, and C.-H. Tsai, "Web Application
Security Assessment by Fault Injection and Behavior Monitoring," Proc. Int'l Conf. World Wide Web, pp. 148-159, 2003.

[21] J. Fonseca, M. Vieira, and H. Madeira, "Detecting Malicious SQL,"
Proc. Conf. Trust, Privacy & Security in Digital Business, Sept. 2007.