IISRC International Conference on Industrial Scientific Research Engineering
International Industrial Scientific Research Corporation
www.worldconferences.org

Evaluation of Web Security Mechanisms using Vulnerability & Attack Injection

SYED NURJA[1], M. HYMAVATHI[2], SYED ABDUL HAQ[3]
[1]PG Scholar, Dept of CSE, QCET, Nellore, AP, India.
[2]Associate Professor, Dept of CSE, QCET, Nellore, AP, India.
[3]HOD, Dept of CSE, QCET, Nellore, AP, India.

Abstract: In this paper we 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 describes the implementation of the Vulnerability & Attack Injector Tool (VAIT) that allows the automation of the entire process. We used this tool to run a set of experiments that demonstrate the feasibility and the effectiveness of the proposed methodology. The experiments include the evaluation of coverage and false positives of an intrusion detection system for SQL Injection attacks and the assessment of the effectiveness of two top commercial web application vulnerability scanners. Results show that the injection of vulnerabilities and attacks is indeed an effective way to evaluate security mechanisms and to point out not only their weaknesses but also ways for their improvement.

Keywords: Fault-Tolerance Algorithms And Mechanisms (FTAM), Vulnerability & Attack Injector Tool (VAIT).

I. INTRODUCTION

The security motivation of web application developers and administrators should reflect the magnitude and relevance of the assets they are supposed to protect. Although there is an increasing concern about security (often being subject to regulations from governments and corporations), there are significant factors that make securing web applications a difficult task to achieve:

- The web application market is growing fast, resulting in a huge proliferation of web applications, based on different languages, frameworks, and protocols, largely fuelled by the (apparent) simplicity one can develop and maintain such applications.
- Web applications are highly exposed to attacks from anywhere in the world, which can be conducted by using widely available and simple tools like a web browser.
- It is common to find web application developers, administrators and power users without the required knowledge or experience in the area of security.
- Web applications provide the means to access valuable enterprise assets. Many times they are the main interface to the information stored in backend databases; other times they are the path to the inside of the enterprise network and computers.

The main objective of this work is to propose a methodology and a tool to inject vulnerabilities and attacks in web applications. The proposed methodology is based on the idea that we can assess different attributes of existing web application security mechanisms by injecting realistic vulnerabilities in a web application and attacking them automatically. This follows a procedure inspired on the fault injection technique that has been used for decades in the dependability area. In our case, the set of “vulnerability” þ “attack” represents the space of the “faults” injected in a web application, and the “intrusion” is the result of the successful “attack” of a “vulnerability” causing the application to enter in an “error” state . In practice, security “vulnerability” is a weakness (an internal “fault”) that may be exploited to cause harm, but its presence does not cause harm by itself. 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 , and are injected according to a set of representative restrictions and rules.

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. These experiments illustrate how the proposed methodology can be used in practice, not only to uncover existing weaknesses of the tools analyzed, but also to help improve them. It is accomplished by making easy to understand screens for the information section to handle huge volume of information. The objective of planning info is to make information section less demanding and to be free from blunders. The information passage screen is outlined in a manner that all

the information controls can be performed. It additionally gives record seeing offices. At the point when the information is entered it will check for its legitimacy. Information can be entered with the assistance of screens. Fitting messages are given as when required so that the client won't be in maize of moment. In this way the goal of information configuration is to make a data format that is anything but difficult to take after.

II. LITERATURE SURVEY

A. "Precise Alias Analysis for Static Detection of Web Application Vulnerabilities,"

AUTHORS: N. Jovanovic, C. Kruegel, and E. Kirda

The number and the significance of web applications have expanded quickly in the course of the most recent years. In the meantime, the amount and effect of security vulnerabilities in such applications have developed also. Since manual code surveys are drawn out, blunder inclined and immoderate, the requirement for robotized arrangements has ended up apparent. In this paper, we address the issue of helpless web applications by method for static source code examination. To this end, we show a novel, exact false name examination focused at the special reference semantics usually found in scripting dialects. Also, we upgrade the quality and amount of the produced helplessness reports by utilizing a novel, iterative two-stage calculation for quick and exact determination of record considerations. We incorporated the displayed ideas into Pixy~\cite {jova novic06:pixy_short}, a high-exactness static investigation device went for identifying cross-site scripting vulnerabilities in PHP scripts. To show the viability of our methods, we examined three web applications and found 106 vulnerabilities. Both the high analysis speed as well as the low number of generated false positives shows that our techniques can be used for conducting effective security audits.

B."Fault Injection for Formal Testing of Fault Tolerance,"

AUTHORS:D.Avresky,J.Arlat,J.C.Laprie,andY. Crouzet

This study addresses the use of fault injection for explicitly removing design/implementation faults in complex fault-tolerance algorithms and mechanisms (FTAM), viz, fault-tolerance deficiency faults. Formalism is acquainted with speak to the FTAM by an arrangement of affirmations. This formalism empowers an execution tree to be created, where every way from the root to a leaf of the tree is an all around characterized recipe. The arrangement of all around characterized equations constitutes a helpful structure that completely describes the test succession. The data examples of the test succession (issue and enactment spaces) then are resolved to less particular basic criteria over the execution tree (actuation of legitimate arrangements of ways). This gives a system to creating an utilitarian deterministic test for projects that execute complex FTAM. This technique has been utilized to amplify an investigating apparatus went for testing adaptation to internal failure conventions created by BULL France. It has been connected effectively to the infusion of shortcomings in the between copy convention that backings the application-level adaptation to non-critical failure elements of the construction modeling of the ESPRIT-subsidized Delta-4 undertaking. The consequences of these examinations are investigated in point of interest. Specifically, despite the fact that the objective convention had been freely confirmed formally, the utilization of the proposed testing method uncovered two adaptation to non-critical failure inadequacy flaws.

C. "Mapping Software Faults with Web Security Vulnerabilities,"

AUTHORS: J. Fonseca and M. Vieira

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

D. "Fault Injection and Dependability Evaluation of Fault-Tolerant Systems",

AUTHORS; J. Arlat, A. Costes, Y. Crouzet, J.-C. Laprie, and D. Powell,

The authors describe a dependability evaluation method based on fault injection that establishes the link between the experimental assessment of the adaptation to internal failure procedure and the issue event process. The principle qualities of a flaw infusion test succession went for assessing the scope of the adaptation to non-critical failure procedure are exhibited. Accentuation is given to the deduction of exploratory measures. The different strides by which the shortcoming event and adaptation to non-critical failure procedures are consolidated to assess steadfastness measures are recognized and their associations are dissected. The system is represented by an application to the reliability assessment of the circulated shortcoming tolerant building design of the Esprit Delta-4 Project.

III. EXISTING SYSTEM

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. However, these best practices are unfeasible to apply to the hundreds of millions of existing legacy web applications, so they should be constantly audited and protected by security tools during their lifetime. This is particularly relevant due to the extreme dynamicity of the security scenario, with new vulnerabilities and ways of exploitation being discovered every day.

Disadvantages of Existing System:

- Clearly, security technology is not good enough to stop web application attacks and practitioners should be concerned with the evaluation and the assurance of their success.
- In practice, there is a need for new ways to effectively test existing web application security mechanisms in order to evaluate and improve them.

IV. 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 as shown in Fig.1. 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.

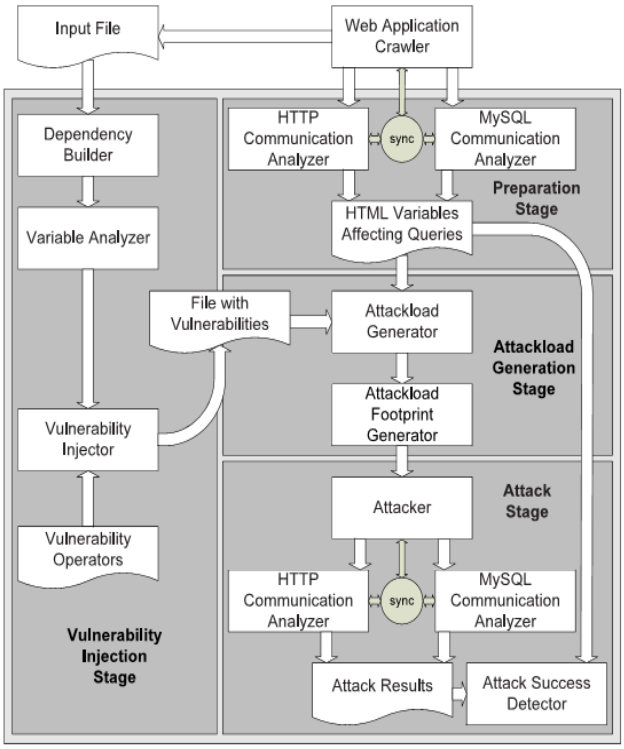

Fig.1. System Architecture.

Advantages of Proposed 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. This assessment of security tools can be done online by executing the attack injector while the security tool is also running; or offline by injecting a representative set of vulnerabilities that can be used as a testbed for evaluating a security tool.

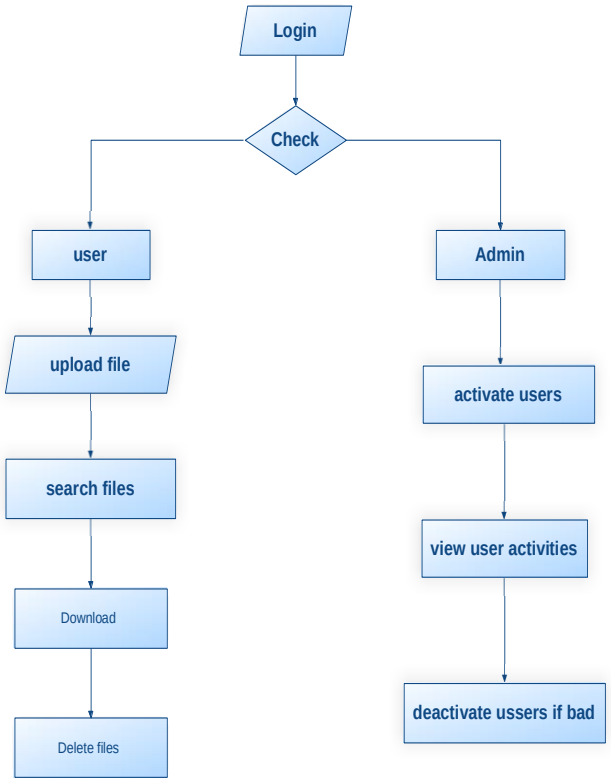

Fig.2. Flow Diagram.

V. MODULES

- Web Crawled Information
- Vulnerability Injection
- Generate Attack Load for Malicious Node
- Synchronization Process with HTTP and MySql

A. Web Crawled Information

The web application is cooperated (creeped) executing every one of the functionalities that should be tried. In the interim, both HTTP and SQL interchanges are caught by the two tests and prepared for later utilize. The connection with the web application is constantly done from the customer's perspective the HTTP variables that convey them and their particular source code documents, and its utilization in the structure of the database inquiries sent to the back-end database.

B. Helplessness Injection

It is in this helplessness infusion arrange that vulnerabilities are infused into the web application. For this reason, it needs data about which include variables convey pertinent data that can be utilized to execute assaults to the web application. This stage begins by dissecting the source code of the web application records hunting down areas where vulnerabilities can be infused (Fig. 2). The injection of vulnerabilities is done by removing the protection of the target variables, like the call to a sanitizing function.

C. Generate Attack Load for Malicious Node

After having the set of copies of the web application source code files with vulnerabilities injected we need to

generate the collection of malicious interactions (attack loads) that will be used to attack each vulnerability. This is done in the attack load generation stage. The attack load is the malicious activity data needed to attack a given vulnerability. This data is built around the interaction patterns derived from the preparation stage, by tweaking the input values of the vulnerable variables.

D. Synchronization Process with HTTP and MySql

The next component is the Variable Analyzer. Because SQLi vulnerabilities rely on vulnerable variables that can be exploited, we have to analyze all the variables that are used to build SQL queries. This component gathers all the PHP variables from the source code and builds a mesh of dependencies related to each other. Then, it searches for PHP variables present in SQL query strings. This is important as one variable may be used only as input (POST or GET HTTP parameters) and the result is passed to another variable that is the one that is in the SQL query string. All the other variables are discarded.

VI. CONCLUSION & FUTURE ENHANCEMENT

This paper proposed a novel methodology to automatically inject realistic attacks in web applications. This methodology consists of analyzing the web application and generating a set of potential vulnerabilities. Each vulnerability is then injected and various attacks are mounted over each one. The success of each attack is automatically assessed and reported. The realism of the vulnerabilities injected derives from the use of the results of a large field study on real security vulnerabilities in widely used web applications. This is, in fact, a key aspect of the methodology, because it intends to attack true to life vulnerabilities. To broaden the boundaries of the methodology, we can use up to date field data on a wider range of vulnerabilities and also on a wider range and variety of web applications. To demonstrate the feasibility of the methodology, we developed a tool that automates the whole process: the VAIT. Although it is only a prototype, it highlights and overcomes implementation specific issues. It emphasized the need to match the results of the dynamic analysis and the static analysis of the web application and the need to synchronize the outputs of the HTTP and SQL probes, which can be executed as independent processes and in different computers. All these results must produce a single analysis log containing both the input and the output interaction results. The VAIT prototype focused on the most important fault type, the MFCE (vulnerabilities caused by a missing function protecting a variable), generating SQLi vulnerabilities. Although this fault type represents the large majority of all the faults classified in the field study and can be considered representative, other fault types can also be implemented, namely those that come next concerning their relevance.

Future Enhancement: The experiments have shown that the proposed methodology can effectively be used to evaluate security mechanisms like the IDS, providing at the same time indications of what could be improved. By injecting vulnerabilities and attacking them automatically the VAIT could find weaknesses in the IDS. These results were very important in developing bug fixes (that are already applied to the IDS software helping in delivering a better product). The VAIT was also used to evaluate two commercial and widely used web application vulnerability scanners, concerning their ability to detect SQLi vulnerabilities in web applications. These scanners were unable to detect most of the vulnerabilities injected, in spite of the fact that some of them seemed to easily be probed and confirmed by the scanners. The results clearly show that there is room for improvement in the SQLi detection capabilities of these scanners.

VII. REFERENCES

[1] USA, "Sarbanes-Oxley Act," 2002.
[2] Payment Card Industry (PCI) Data Security Standard, PCI Security Standards Council, 2010.
[3] S. Christey and R. Martin, "Helplessness Type Distributions in CVE," Miter Report, May 2007.
[4] S. Zanero, L. Carettoni, and M. Zanchetta, "Programmed Detection of Web Application Security Flaws," Black Hat Briefings, 2005.
[5] N. Jovanovic, C. Kruegel, and E. Kirda, "Exact Alias Analysis for Static Detection of Web Application Vulnerabilities," Proc. IEEE Symp. Security Privacy, 2006.
[6] J. Williams and D. Wichers, "OWASP Top 10," OWASP Foundation, Feb. 2013.
[7] IBM Global Technology Services "IBM Internet Security Systems X-Force 2012 Trend & Risk Report," IBM Corp., Mar. 2013.
[8] Verizon "2011 Data Breach Investigations Report," 2011.
[9]ThePrivacyRightsClearinghousewww.privacyrights.org/databreach, Accessed 1 May 2013, Apr. 2012.
[10] M. Fossi, et al., "Symantec Internet Security Threat Report: Trends for 2010," Symantec Enterprise Security, 2011.
[11] M. Fossi, et al., "Symantec Report on the Underground Economy, Symantec Security Response," 2008.
[12] R. Richardson and S. Dwindles, "2010/2011 CSI Computer Crime & Security Survey," Computer Security Inst., 2011.
[13] D. Avresky, J. Arlat, J.C. Laprie, and Y. Crouzet, "Flaw Injection for Formal Testing of Fault Tolerance," IEEE Trans. Unwavering quality, vol. 45, no. 3, pp. 443-455, Sept. 1996.
[14] D. Powell and R. Stroud, "Theoretical Model and Architecture of MAFTIA," Project MAFTIA, Deliverable D21, 2003.
[15] V. Krsul, "Programming Vulnerability Analysis," PhD proposition, Purdue Univ., West Lafayette, IN 1998.
[16] J. Fonseca and M. Vieira, "Mapping Software Faults with Web Security Vulnerabilities," Proc. IEEE/IFIP Int'l. Conf. Tried and true Systems and Networks, June 2008.