ISSN 2319-2720
Volume 7, No.2, April – June 2018
International Journal of Computing, Communications and Networking
Available Online at http://www.warse.org/ijccn/static/pdf/file/ijccn40722018.pdf
https://doi.org/10.30534/ijccn/2018/40722018


# Evaluation of Web Security Mechanisms Using Vulnerabilities and Attack Injection

**Syeda Sheeba Tabassum[1], Pratheeka M[2] , Megha P[3], Dr. Arun Biradar[4]**
[1]Student, EWIT India. syedasheeba.2018@gmail.com
[2] Student, EWIT, India. prathikamogaveera@gmail.com
[3] Student, EWIT India. meghaputtaswamy@gmail.com
[4] Professor & Head Department of CSE, EWIT, India.hodcsea@gmail.com

.

**ABSTRACT**

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: Security, fault injection, internet applications, review and evaluation.**

## 1.INTRODUCTION

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. 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:

1. The web application market is growing fast, resulting in a huge proliferation of web applications, based on different languages, frameworks, and protocols, largely fueled by the (apparent) simplicity one can develop and maintain such applications.

2. 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.

3.It is common to find web application developers, administrators and power users Without the required knowledge or experience in the area of security.

4.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.

To fight this scenario 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. 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. This paper proposes 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. In practice, a 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 and are injected according to a set of representative restrictions and rules. 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. 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 vulnerabilities that can be successfully attacked and discarded those that cannot. Although this methodology can be applied to various types of vulnerabilities, we focus on two of the most widely exploited and serious web application vulnerabilities that are SQL Injection (SQLi) and Cross Site Scripting (XSS). 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). 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 firewalls, 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. 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. 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. The structure of the paper is as follows. The next section presents related research. Section 3 describes the proposed attack injection methodology.

## 2.RELATED WORK

In this paper we addressed the problem of providing transaction security in decentralized SG energy trading without reliance on trusted third party[1].

We have developed a secure PUF based authentication and certificate proofs for the protocol have been formulated under the SK security and UC framework[2].

Fault injection techniques have traditionally been used to inject physical (i.e., hardware) faults [18], [19]. In fact, initial fault injection techniques used hardware-based approaches such as pin-level injection or heavy-ion radiation. The increasing complexity of systems has lead to the replacement of hardware-based techniques by software implemented fault injection (SWIFI), in which hardware faults are emulated by software. xception [20] and NFTAPE [21] are examples of SWIFI tools.

## 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.

## 3 SYSTEM ARCHITECTURE

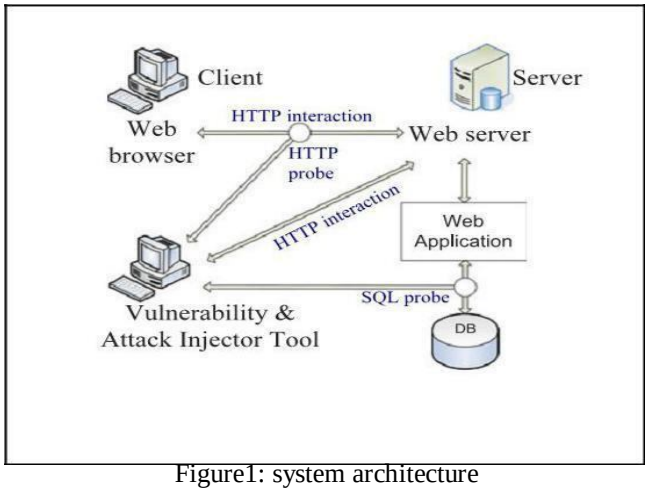


Figure1: system architecture

The vulnerabilities are injected in the web application following a realistic pattern derived . The information about what was injected is fed to the injection mechanism in order to improve the attack success rate. As shown in Fig. 1, the attack injection uses two external probes: one for the HTTP communication and other for the database communication. These probes monitor the HTTP and SQL data exchanged, and send a copy to be analyzed by the attack injection mechanism. This is a key aspect of the methodology to obtain the user interaction and the results produced by such interaction for analysis, so they can be used to prepare the attack. Therefore, the attack injection mechanism is aware of important inner workings of the application while it is running. For example, this provides insights on what piece of information supplied to

a HTML FORM is really used to build the correlated SQL query and in which part of the query it is going to be inserted. The entire process is performed automatically, without human intervention. For example, let‟s consider the evaluation of an IDS: during the attack stage, when the IDS inspects the SQL query sent to the database, the VAIT also monitors the output of the IDS to identify if the attack has been detected by the IDS or not. We just have to collect the final results of the attack injection, which also contains, in this case, the IDS detection output.

**4. ALGORITHM :**

Table1: Algorithm

```
Pseudo-code of the InfoSpiders algorithm
{initialize each agent‟s genotype,energy and
starting page} PAGES-maximum no of pages to
visit While number of visited pages<PAGES
do While for each agent a do
{Pick and visit an out-link from the current agent‟s page}
{update the energy estimating benefit()-cost()}
{update the genotype as a function of the current benefit}
if agent‟s energy>THRESHOLD then
{apply the generic operators to
produce offspring} else
{Kill the agent}
else if
 end while
end while
```

**5. MODULES:**

**5.1 Overview of the Methodology**

Our Vulnerability & Attack Injection methodology for SQLi and XSS can be applied to a variety of setups and technologies, but the following description uses as reference a typical web application, with a web front-end andThe entire process is performed automatically, without human intervention. For example, let‟s consider the evaluation of an IDS: during the attack stage, when the IDS inspects the SQL query sent to the database, the VAIT also monitors the output of the IDS to identify if the attack has been detected by the IDS or not. We just have to collect the final results of the attack injection, which also contains, in this case, the IDS detection output. The automated attack of a web application is a multistage procedure that includes: preparation stage, vulnerability injection stage, attackload generation stage, and attack stage. These stages are described in the next sections

**5.2 Preparation Stage**

In the preparation stage, the web application is interacted (crawled) executing all the functionalities that need to be tested (Fig. 2). Meanwhile, both HTTP and SQL communications are captured by the two probes and processed for later use. The interaction with the web application is always done from the client‟s point of view (the web browser). The outcome of this stage is the correlation of the input values, the HTTP variables that carry them and their respective source code files, and its use in the structure of the database queries sent to the back-end database (for SQLi) or displayed back to the web browser (for XSS). Later on, in the attack stage, the malicious activity applied.

**5.3 Vulnerability Injection Stage**

It is in this vulnerability injection stage that vulnerabilities are injected into the web application. For this purpose, it needs information about which input variables carry relevant information that can be used to execute attacks to the web application. This stage starts by analyzing the source code of the web application files searching for locations where vulnerabilities can be injected (Fig. 2). The injection of vulnerabilities is done by removing the protection of the target variables, like the call to a sanitizing function. This process follows the realistic patterns resulting from the field study presented in [16]. Once it finds a possible location, it performs a specific code mutation in order to inject one vulnerability in that particular location. The change in the code follows the rules derived from [16], which are described and implemented as a set of Vulnerability Operators presented in [17]. The Vulnerability Operators are built upon a pair of attributes: the Location Pattern and the Vulnerability Code Change.

**5.4 AttackLoad Generation Stage**

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 (attackloads) that will be used to attack each vulnerability. This is done in the attackload generation stage. The attackload is the malicious activity data needed to attack a given vulnerability. patterns stage, by tweaking derived from the preparation the input values of the vulnerable variables.

**CONCLUSION**

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.

**REFERENCES**

[1]UrbiChatterjee,Vidya Govindan,Rajat Sadhukhan "Buuiliding LUF Based Authentication"IEEE Trans May-2018.

[2] Nurzhan Zhumabekuly Aizhan,Davor Suetinovic Security and privacy in decentralized energy"IEEE Trans Oct-2016.

[3] V. Krsul, "Software Vulnerability Analysis," PhD thesis, Purdue Univ., West Lafayette, IN 1998.

[4] J. Fonseca and M. Vieira, "Mapping Software Faults with Web Security Vulnerabilities," Proc. IEEE/IFIP Int‟l. Conf. Dependable Systems and Networks, June 2008. https://doi.org/10.1109/DSN.2008.4630094

[5] J. Fonseca, M. Vieira, and H. Madeira, "Training Security Assurance Teams using Vulnerability Injection," Proc. IEEE
Pacific Rim Dependable Computing Conf., Dec. 2008. https://doi.org/10.1109/PRDC.2008.43

[6] 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. 1993. https://doi.org/10.1109/12.238482

[7] R. Iyer, "Experimental Evaluation," Proc. IEEE Symp. Fault Tolerant Computing, pp. 115-132, Special Issue FTCS-25 Silver Jubilee, 1995.

[8] 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.

[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 https://doi.org/10.1109/IPDS.2000.839467.

[10] J. Christmansson and R. Chillarege, "Generation of an Error Set that Emulates Software Faults," Proc. IEEE Fault Tolerant Computing Symp., 1996. https://doi.org/10.1109/FTCS.1996.534615

[11] 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. https://doi.org/10.1109/ICDSN.2000.857571

[12] 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. https://doi.org/10.1109/TSE.2006.113 SQLBlock.com,

[13] 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.

[14] 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.

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

[16] 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.

[17]cgisecurity.net,www.cgisecurity.com/articles/csrf
-faq.sht ml# what is, Dec. 2008.

[18] Sam NG. CISA, CISSP.
www.owasp.org/images/d/Advanced_To
pics_on_SQL_Injection_Protection.ppt,
2006.

[19] S. McConnell, "Gauging Software Readiness with Defect Tracking," IEEE Software, vol. 14, no. 3, May/June 1997. https://doi.org/10.1109/52.589257

[20] 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.

[21] 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. https://doi.org/10.1109/IPDS.2000.839467

Syeda Sheeba Tabassum: pursuing B.E in CSE, EWIT (VTU), Bengaluru. Her areas of interest are Computer Security, Databases, Computer Networks, Software Engineering, Python Programming etc.

Pratheeka M: pursuing B.E in CSE, EWIT (VTU), Bengaluru. Her areas of interest are Databases, Computer Networks, Computer Graphics, Software Engineering, Internet of Things etc.

Megha P: pursuing B.E in CSE, EWIT (VTU), Bengaluru. Her areas of interest are Web Security, Databases, Computer Networks, Internet of Things , Programming the Web, Software Engineering, Java language,Computer Graphics, etc.

Dr. Arun Biradar: Professor & Head, Department of Computer Science & Engineering, East West Institute of Technology (VTU), Bengaluru. Qualification: B.E, M.Tech, Ph.D, Board of Examiner (Member), VTU, Belagavi. His areas of research are Wireless Ad-hoc Networks, Computer Networks, Software Engineering, Genetic Algorithms, Machine Learning, IoT and Cloud Computing.