Correlating security vulnerabilities with software faults

José Fonseca[1], Marco Vieira[2], Henrique Madeira[2]
[1]*CISUC, Polytechnic Institute of Guarda, Portugal*
[2]*CISUC, University of Coimbra, Portugal*
josefonseca@ipg.pt, {mvieira,henrique}@dei.uc.pt

Abstract

Web applications are typically developed with hard time constraints and are often deployed with software bugs. One important type of bug is the one that creates security vulnerabilities. Knowing what kind of programming mistakes usually lead to security vulnerabilities can be an important tool to help in the detection and prevention of security flaws. In this paper we describe ongoing work on correlating software faults and security vulnerabilities like Cross Site Scripting (XSS) and SQL injection.

1. Introduction

Security is a major concern in web applications. In fact, web applications developers are normally focused on the application functionalities and on satisfying the user's requirements and easily neglect security aspects. Even assuming that key infrastructure components such as web servers and database management systems (DBMS) are fully secure (which is far from being guaranteed), poor web application code represents a major risk, as can be confirmed by innumerous vulnerabilities daily reported in specialized sites (e.g., www.securityfocus.com, http://www.ntbugtraq.com, www.cve.mitre.org, http://www.kb.cert.org/vuls, http://www.microsoft.com/technet/security).

For example, according to Acunetix audit results [1] "on average 70% of websites are at serious and immediate risk of being hacked... 91% of these websites, contained some form of website vulnerability, ranging from the more serious such as SQL Injection and XSS to more minor ones such as local path disclosure or directory listing".

These security vulnerabilities give attackers an easy gateway to valuable assets like the access to unauthorized data, execution of database stored procedures, or even gain access to privileged database accounts, impersonate another user, mimicry web applications, or even get access to the web server.

Our idea is to analyze how software faults correlate with security vulnerabilities they may generate. The goal is to observe the kind of security vulnerabilities that are generated by each type of faults in order to understand what types of faults contribute more to vulnerabilities such as XSS and SQL injection.

The injection of software faults to study security breaches is a new research topic that can be used for a variety of studies in the security area. For example, it can be used to guide code reviewers or to compare the detection coverage of automatic vulnerability scanners. Injection of realistic software faults can also be used to build attack injection tools, which could be the equivalent of a classical fault injection tool for security assessment (e.g., to evaluate Intrusion Detection Systems or Firewalls).

2. Mapping software faults with vulnerabilities

In our study, we inject the most frequent types of software programming errors (according to the field study presented in [2]) to check what kind of vulnerabilities they may produce when applied to web applications. We should clarify that we are talking about vulnerabilities in applications, and not vulnerabilities in components such as web servers, that are dependent on the configuration.

There are several automatic Web Vulnerability Scanners available to detect vulnerabilities in web applications after deployment. Most of them are commercial tools such as Acunetix Web Vulnerability Scanner [3], Spidynamics Webinspect [4], Whatchfire AppScan [5], Buyservers Falcove [6], and N-Stalker Web Application Security Scanner [7]. There are also some freeware application scanners like Gamja [8], but they lack most of the functionalities of their commercial counterparts and are not so accurate.

The XSS and the SQL injection are the most frequent types of vulnerabilities in websites and are considered as critical vulnerabilities by the vulnerability scanners.

Our study is based on two steps: 1) we inject software faults in a web application (one fault at each time) and 2) we use Web vulnerability scanners to check if the injected fault has created any security vulnerability in the target application. As we injected different types of faults we can obtain a map that relates types of faults with the vulnerabilities they may create.

We use the G-SWFIT (Generic Software Fault Injection Technique) [2] technique to inject the software faults. G-SWFIT is based on a set of fault injection operators that reproduce most common types of software faults directly in the target executable code. These fault injection operators resulted from a field study that analyzed and classified more than 500 real software faults discovered in several programs, identifying the most common (the "top-N") types of software faults [2]. Once the target application in the current study are PHP web applications, which are different form the C and C++ application used in [2], we had to introduce some necessary adaptations to the fault models (but without changing the type of fault represented by each fault model). As the faults injected are based on the most frequent ones observed in the field [2], we believe that the faults injected in our study do represent the faults that are likely to exist in web application code.

In the current stage of the study, we have used only 3 types of faults, corresponding to 26% of all the software faults observed in [2]. A comprehensive fault injection campaign is planned covering all the types of faults identified in [2].

For the evaluation experiments we used a web application developed in LAMP (Linux operating system, Apache web server, Mysql database and PHP programming language). This LAMP topology represents one of the most used combinations of technologies to build web sites. It is also in this kind of setup that the largest number of security vulnerabilities is found mainly due to SQL injection and XSS.

After the injection of each fault, the detection of possible vulnerabilities is carried out by two web application vulnerability scanners: Acunetix Web Vulnerability Scanner [3] and Whatchfire AppScan [5]. The scanners perform a comprehensive search for vulnerabilities as they execute hundreds or even thousands of tests for each vulnerability type (XSS, SQL injection, path traversal, buffer overflow, denial of service, brute force, etc.). Although the scanner may report other vulnerabilities, for the time being we are only interested in the XSS and SQL injection types. Before each experiment the web system is reset to the initial state and only one software fault is injected in each test.

3. Preliminary results and discussion

The web application code under test has around 1.500 lines of code. This application has 168 fault locations for the 3 fault types used and the results of the preliminary experiments are in Table 1.

Table 1 – Experiment results

Fault Types	# Faults	XSS	SQL Inject.
Missing "If (*cond*) {statement(s)}"	23	1	0
Missing function call	98	4	20
Missing "AND EXPR" in expression used as branch condition	48	4	0

The faults injected leaded to software bugs and application malfunctioning, but they also produced a considerable amount of security vulnerabilities (around 17%). As we can see, SQL injection vulnerabilities appears only for the fault type "Missing function call", while XSS is generated by all the fault types used. In the next steps of our work we will use more fault types and other web applications to check the possibility of mapping how software fault types generate security vulnerabilities.

4. References

[1] Acunetix Ltd, February 12, 2007, http://www.acunetix.com/news/security-audit-results.htm

[2] J. Durães and H. Madeira, "Emulation of Software Faults: a Field Data Study and a Practical Approach", IEEE Trans. on Software Engineering Vol. 32, No. 11, Nov. 2006.

[3] Acunetix Ltd, Web Vulnerability Scanner, 2007, http://www.acunetix.com/vulnerability-scanner/

[4] SPI Dynamics, WebInspect, 2007, http://www.spidynamics.com/products/webinspect/

[5] Watchfire Corporation, AppScan, 2007, http://www.watchfire.com/

[6] BuyServers Ltd., Falcove, 2007, http://www.buyservers.net/falcove.htm

[7] N-Stalker, 2007, http://www.nstalker.com/

[8] Gamja, 2007, http://lastlog.com/p4ssion/

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

[10] R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, D. Moebus, B. Ray, M. Wong, "Orthogonal Defect Classification – A Concept for In-Process Measurement", IEEE Transactions on Software Engineering, vol. 18, no. 11, pp. 943-956, November 1992.