UDC 621.396

WEB-APPLICATIONS VULNERABILITIES TESTING TECHNIQUE

M. ABOUKRA, N.V. NASONOVA

Belarusian State University of Informatics and Radioelectronics, Republic of Belarus

Submitted 9 November 2020

Abstract. The technique for detecting vulnerabilities in web applications is developed. The technique combines exploration, GrayBox and testing principles to detect the riskiest vulnerabilities rated by OWASP.

Keywords: Web-applications, vulnerabilities, BlackBox, WhiteBox testing.

Introduction

An unsafe program is a potential target for an attacker who can use existing vulnerabilities to unauthorized access to available information, influence the operation of programs and services (start or stop), and inject malicious code into the system [1]. Software vulnerabilities are mainly caused by logical errors and software errors.

Adversary model

A web application testing methodology is needed to protect a company's web resources from the most common, dangerous, malicious threats associated with programming and configuration errors in web applications. First an adversary model should be developed. Assume, that the attacker has no access to the application source code. It can interact with it indefinitely (it is on the same local network or the application is available via the Internet and there is no firewall). This option is common. Also, assume, that the attacker understands the architecture of web applications and is qualified enough to perform the attack. The task of the vulnerabilities testing technique is to find the most critical vulnerabilities, spending a small number of resources.

Application vulnerabilities

There are several popular classifiers of vulnerabilities [2]:

– CWE (Common Weakness Enumeration) – a database of vulnerability types. The main task of the project is to provide descriptions of common types of vulnerabilities, ways to prevent, detect and fix them;

– CVE (Common Vulnerabilities and Exposures) – a vocabulary of specific vulnerabilities of software;

– SecurityFocus BID;

– IBM ISS X-Force.

The CWE classifier contains generalized classes of vulnerabilities that are often found in software products [3]. The main threats to web application security are shown in Fig. 1.

The most common at the moment are the following web application vulnerabilities: XSS (Cross Site Scripting), various injections (PHP, SQL), RCE (Remote Code Execution), errors in the implementation of authentication and authorization, and unsafe deserialization of objects [3].

Vulnerability detection methods

Successful testing of web applications requires a systematic approach or methodology. The most popular are OWASP and WASC. They are the most complete and formalized methodologies to date.

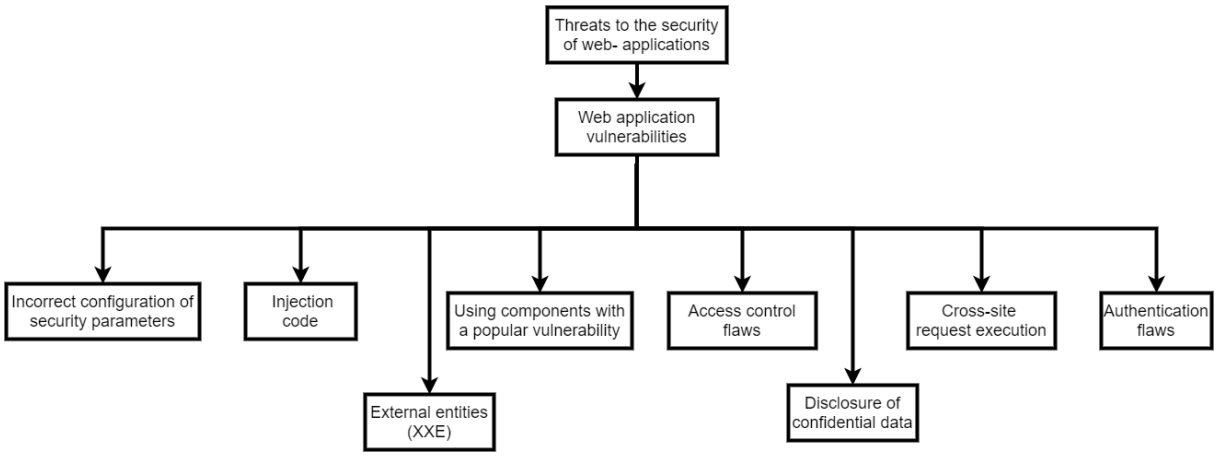

Fig. 1. The main threats to the security of web applications

There are several testing principles that we can apply to detect vulnerabilities [4]:

– DAST – dynamic (i.e. requiring execution) analysis of the application without access to the source code and the server side, the so called BlackBox testing;

– SAST – static (i.e., not requiring execution) analysis of an application with access to the source code of the web application and to the web server; in fact, it is an analysis of the source code for formal signs of vulnerabilities and a server security audit;

– IAST – dynamic analysis of the security of a web application, with full access to the source code, the web server – WhiteBox testing;

– Source code analysis – static or dynamic analysis with access to the source code without access to the server environment.

The following techniques are used for BlackBox testing:

– equivalent partition;

– boundary value analysis;

– analysis of cause and effect relationships;

– assumption of error.

This technique detects the following categories of errors:

– incorrectly implemented or missing features;

– interface errors;

– errors in data structures or organization of access to external databases;

– behavioral errors or insufficient system performance.

BlackBox testing is limited by its disadvantages: only a very limited number of program execution paths are tested and some tests may be redundant if they have already been conducted by the developer at the unit testing level. It is also quite difficult to create effective test cases without a clear specification. Moreover, this testing method has a number of advantages, as testing is done from the perspective of the end user and can help to detect inaccuracies and inconsistencies in the specification. The tester does not need to know programming languages and delve into the specifics of the program implementation. Testing can be performed by experts independent of the development department, which helps to avoid bias.

The opposite of the BlackBox technique is WhiteBox testing. WhiteBox testing is divided into the following levels:

– unit testing;

– integration testing;

– regression testing;

– hacking testing.

WhiteBox testing can be done at an early stage: there is no need to wait for the creation of the user interface. More thorough testing can be done, covering a large number of program execution paths. But at the same time, this method also has disadvantages, as it requires a lot of specialized knowledge to perform WhiteBox testing. When using test automation at this level, maintaining test scripts can be difficult if the program changes frequently.

The main stages of web application penetration testing include intelligence, access control, selection of parameters, checking the logic of the web application, checking the server environment. Having a test plan for an application, you can step by step examine all its components for the presence of certain vulnerabilities. Based on the specifics of each web application, certain points can be supplemented with checks specific to this application.

Vulnerability scanners are used to facilitate security testing. They allow you to avoid many routine actions and significantly reduce the time required for testing as shown in Fig. 2.

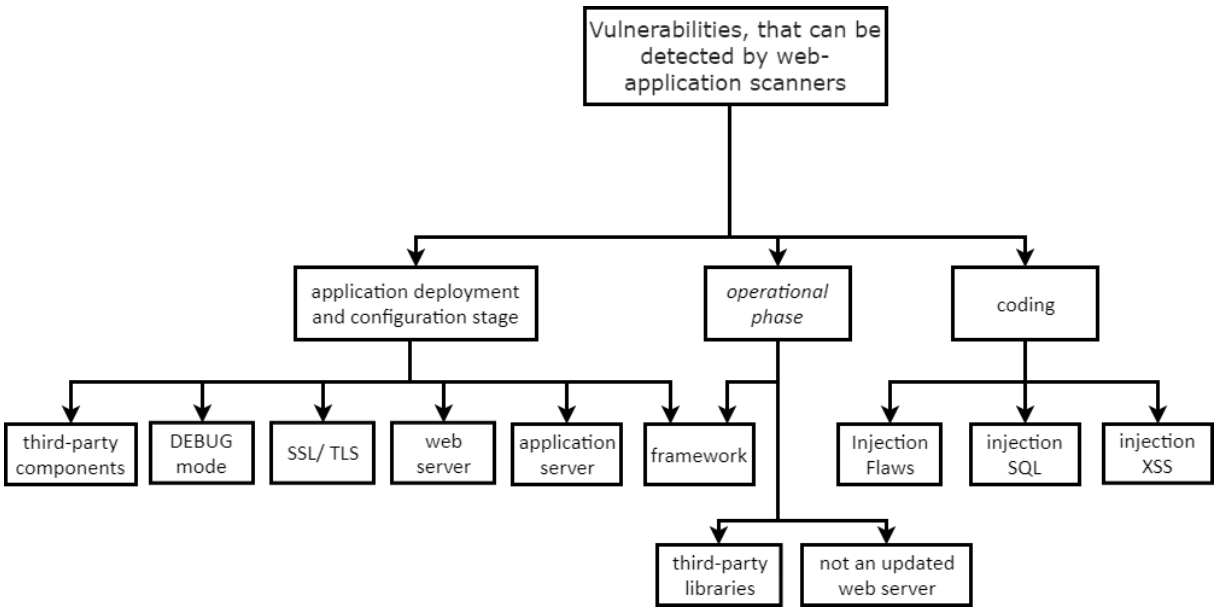

Fig. 2. Vulnerabilities detection using web application scanners

The task of searching for vulnerabilities in the development phase is much more successful when using the scanner in manual mode, rather than automatically. The task of searching for vulnerabilities of the second and third types is automated quite well (especially if the scanner is configured for application technologies and its environment). In manual mode, the operator works with the analyzed application through a web browser, which is connected to the application through the scanner as through an intermediate proxy server. In this case, the operator observes all HTTP requests and HTTP responses in the scanner and, at his choice, can compose and send the necessary test requests based on them. When working with a scanner in this mode, there are the following features: firstly, increased requirements are imposed on the qualifications of the operator, and secondly, it is important for the operator to get the maximum of convenient tools from the scanner to automate the creation and sending of test requests and analyze application responses to them. Burp Suite and ZAP Proxy are typical representatives of this class of scanners. In the automatic mode of operation, the operator is only required to configure the parameters for launching the scanner: specify the application URL and username/password (when testing the closed part). Typical representatives of this class are HP WebInspect, Acunetix, etc.

Development of a technique for detecting vulnerabilities in web applications

The developed technique stages are given in Fig. 3. Vulnerability search starts with exploration. For these purposes, we use the nmap network scanner. The next stage includes the so-called GrayBox testing. It includes both BlackBox and WhiteBox testing elements. First, we conduct an automatic scan for vulnerabilities using ZAP. It also helps you to discover entry points to the application and facilitate further exploration. To bypass authorization, we use dictionary search. As a dictionary we take the top 10,000 most frequently encountered passwords, which allows us to sort out most weak passwords in a relatively short time. Then, after the automated search, we manually test the detected entry points to the application (web pages that accept user input and API endpoints). For manual testing, the application source code should be accessed. These steps will help you to find most of the basic vulnerabilities that an attacker can detect. Then it is necessary to conduct WhiteBox testing, which will make it possible to detect more complex vulnerabilities. The algorithm of the proposed technique is shown in Fig. 3.

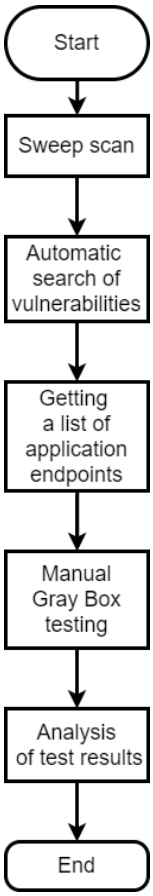

Fig. 3. Algorithm of the developed methodology for searching for vulnerabilities

Thus, this technique allows you to combine the existing methods of searching for vulnerabilities, which makes it possible to detect the most critical vulnerabilities inherent in web applications, the criticality of which is rated by OWASP [5].

Conclusion

A technique for detecting vulnerabilities in web applications is developed. The technique combines exploration, GrayBox and testing principles to detect the riskiest vulnerabilities rated by OWASP. It is suggested to be applied by the companies for their inside testing of the vulnerabilities of the proprietary applications.

References

1. Introduction to Secure Coding Guide [Electronic source]. URL: https://developer.apple.com/library/archive/documentation/Security/Conceptual/SecureCodingGuide.
2. Top 25 Software Errors [Electronic source]. URL: https://www.sans.org/top25-software-errors.
3. BlackBox and WhiteBox testing [Electronic source]. URL: https://quality-lab.ru/blog/key-principles-of-black-box-testing/.
4. Damn Vulnerable Web Application (DVWA) [Electronic source]. URL: http://www.dvwa.co.uk/.
5. OWASP Broken Web Applications [Electronic source]. URL: https://owasp.org/www-project-broken-web-applications/.