Ensurement of web application security over the entire life cycle

Natalia PLEȘCA
State University of Moldova
natalia-plesca@yandex.ru

***Abstract* — web applications have a growing role in the work of any organization today, but there is another aspect of web application exploitation: compromising the application can lead to trust loss in the organization, or the organization will lose some of its customers or will have multiple direct or indirect financial losses (to recover and restore vandalized resources). For these reasons, the development of a secure application is of considerable importance, along with the goal of developing a functional web application.**

***Index Terms* — life cycle, security, vulnerabilities, web application.**

It is known that in the recent period of time, the activity of any organization can't be achieved without its use of web applications implemented in various business processes: presentation sites, electronic shops, banking systems for Internet banking, sales records etc. Also, multiple corporate applications or computer records, which until recently have been used as desktop applications, lately, through modernization, began the use of web technologies.

Web application security is a branch of information security, dealing in particular with the security of web sites, applications and web services [1].

At a high level, web application security is based on application security principles, but these principles are specifically applied to Computer Information Systems and Applications, accessible to users via web interfaces.

In the early 2000s, with the emergence of Web 2.0, which has as a basic feature the user's involvement in building and / or modifying web content (socializing systems), increases the exchange of information through social networks. It also increases the number of services and businesses managed through the web. For this reason, hackers are increasingly trying to compromise digital resources and corporate networks.

To improve the security of web applications, an international community called the Open Web Application Security Project (OWASP) was set up to coordinate worldwide efforts and reduce the risks associated with web applications.

The Top of the most critical vulnerabilities that threaten web applications, according to the Open Web Application Security Project, include:

• Cross Site Scripting (XSS) - this threat allows client scripts to be injected and executed in the victim's browser, which can lead to hijacking user sessions, abnormal website presentation, etc. This vulnerability is caused by inappropriate validation of the data received from the user. HackerOne, which offers rewards for finding bugs in applications, mentions that XSS in 2017 still remains one of the biggest threats in web applications.

• Injection of data and codes through SQL queries - the malicious user can force the application to run unplanned codes or modify the data needed for the application. SQL injections, as part of server queries, are the most common. According to Cenzic's report, created in 2012, XSS and SQL injecting threatens the most common use of 37% and 16% of web apps [2].

The vulnerabilities described below have a lower share of occurrence - from 4% to 1%.

• Malicious file execution - files are included remotely, in order to compromise the server activity. Attacks of this type can affect .php, .xml files, or other files containing code that accepts files from users.

• Unreliable references to objects - the vulnerability can occur when the programmer has placed a reference to an internal object directly in the code: file reference, directory, database record, URL, or a parameter of a form in the code. Routes to web resources must not be accessible and visible in codes. The malicious user could manipulate the object references without having this permission.

• Data leaks and inappropriate error handling - this type of attack is also called "phishing": illegally obtaining sensitive data, log-ins, passwords, bank card details, etc. An attacker can use this vulnerability to steal sensitive data or to make more serious attacks with stolen data, leading to colossal financial losses. Applications may also lose data about their configuration, internal or private data. In 2012, about $ 1.5 billion was lost due to a lack of proper supervision of this vulnerability.

• Successful authentication and session management - the attacker could compromise passwords, keys, or authentication elements in order to assume the identity of other users. This vulnerability is possible when account and session data are not properly protected.

• Unsecured storage of encrypted data - attackers use poorly protected data for identity theft and other offenses (bankcard frauds). This vulnerability can occur in web applications that do not properly use data encryption functions to protect them.

• Unreliable communications - Vulnerability comes from leaking sensitive information due to a poor network infrastructure. It occurs as a result of the failure of network traffic encryption when sensitive communications are made.

All of these vulnerabilities must be controlled while developing web applications. Most web resource owners do

not follow the application security endorsement recommendations at each stage of the secure software development lifecycle, and as a result they get and exploit web-based applications that contain vulnerabilities that could be avoided during the first stages of the life cycle. The presence of these vulnerabilities is a "door" for malicious access to application data - sometimes private, or other informational resources.

Therefore, the same company OWASP recommends assessing application security at each stage of the life cycle. That is, it recommends testing to be done at each stage and not just after the implementation stage, so there is the possibility of correcting the activity processes, allocating additional expenses to eliminate the risks.

Testing is the process of verifying the state of a system or applications according to a set of criteria. OWASP proposes the set of actions to be taken in the fabrication process - it can be considered a testing methodology [3, 4].

Step 1. Before starting developing

1.1. Defining the lifecycle - an appropriate lifecycle should be defined before the application development begins, in which security assurance is inevitable at every stage.

1.2. Reviewing policies and standards - the need to ensure that policies, standards and documentation are in place to be followed in the development process. Documentation is extremely important because it provides guidelines and polices to teams.

"People can only do the right thing if they know what the right thing is."

If the application has to be developed in Java, for example, it is essential to have a secure Java coding standard. If the requirement to encrypt data is formulated, it is essential to have an encryption standard. No policy or standard can cover all the situations faced by the development team. By documenting common and predictable issues, there will be fewer situations where the team will have to make decisions during the development process.

1.3. Developing metrics to ensure traceability - planning of the measurement program is required before development starts. By defining the measurement criteria, the visibility of defects will be ensured both in the processes and in the developed product. It is essential to define metrics before you start developing, as it may be necessary to modify the data collection process.

Step 2. Specifying the requirements and designing the solution

2.1. Reviewing Security Requirements - Security requirements define how an application works from a secure perspective. It is essential that security requirements be tested. Testing in this case means testing assumptions made in requirements and tests to see if there are gaps in defining requirements. For example, if there is a security requirement that users have to be registered before they can access the website management section, it would mean that the user must pre-emptively register or need only to authenticate? It would be good if the requirements were clearly formulated and there were no more interpretations of them.

When looking for gaps in the formulation of requirements, it would be advisable to provide security mechanisms such as:

- User management
- User Authentication
- User authorization
- Data privacy
- Data integrity
- Session management
- Data security
- Methods of separation of subsystems
- Legislation and compliance of standards (including privacy, etc.)

2.2. Architecture Designing - applications must have well-documented architecture design. This documentation may include models, textual documents and other similar artifacts. It is essential that these artifacts be tested to ensure that the design of the architecture imposes the appropriate level of security as defined in the requirements. Identifying security flows in the design phase is not just one of the most cost-effective moments to identify defects, but can be one of the most effective moments to make the necessary changes. For example, if the project asks for authorization of the actions in several places, it may be appropriate to examine a central authorization component. If the application performs data validation in multiple locations, it might be appropriate to develop a centralized validation module (for example, implementing data entry validation in one place rather than in a dozen places is much cheaper). If vulnerabilities are discovered, they should be passed to the system design architect for alternative solutions.

2.3. Creating and Analyzing UML Models - after architectural design, UML (Unified Modeling Language) models that describe how the application works. In some cases, they may already be available. These models must be used for designers to accurately confirm how the application works. If deficiencies are discovered, they should be presented to the system designer for the development of alternative solutions.

2.4. Creating and examining models for threats to the application - with architectural design solutions, UML models - that explain exactly how the system works, it is recommended to develop a risk model that threatens the application. Realistic scenarios for avoiding threats need to be developed. Design and architecture models should be considered to ensure that these threats have been mitigated, accepted by the business or assigned to a third party, such as an insurance firm. When identified threats do not have mitigation strategies, design and architecture models must be reviewed with the system architect to modify the patterns.

Step 3. Creation of application

Theoretically, the realization is the implementation of some design models. However, in the real world, many decisions about design patterns are made during the code execution. These are often smaller decisions, either too detailed to be described in the design stage, or problems in which no standard has been used. If design and architecture models are not appropriate, the developer will have to

make many decisions independently. If there are insufficient policies and standards, the developer will be faced with the situation to make even more decisions.

3.1. Passing the code - the security team should upgrade the code, running it with developers, and in some cases with system architects. A code scroll is a high-level scroll through which developers explain the logic and flow of the implemented code. This allows the code review team to gain a general understanding of the code and allows developers to explain why certain things have been developed so and not otherwise. The goal is not to revise the code, but to understand the flow, layout and structure of the code that make up the application at a high level.

3.2. Revision of the code - after understanding the structure of the code and why certain things have been coded as they were, the tester can examine the current code for detecting security flaws. Static checks validate the code against a set of checklists, including:

• Requirements regarding availability, confidentiality and integrity;

• OWASP or Top 10 for technical exposures (depending on the depth of the review);

• Specific issues related to the language or framework used, such as the "Scarlet Study" for PHP checklists or "Microsoft Secure Coding" for ASP.NET;

• Any requirements specific to the software industry, such as Sarbanes-Oxley 404, COPPA, ISO / IEC 27002, APRA, HIPAA, Visa Merchant etc.

Regarding the return on investment (most being the time resources), white-box testing yields much higher returns than any other method of security review and is least reliant on the tester's skills. However, this test method is not ideal and the code should be tested in terms of security using other more complete testing methods.

Step 4. The implementation of the application

4.1. Testing for application penetration - after testing requirements, modeling, and reviewing the code, one could assume that all the problems have been captured. Testing the application after it has been installed provides a final check for the certainty that the application is fully functional.

4.2. Configuration management testing - the application penetration test should include checking how the application was deployed and secured. While the application can be secured, the configuration may still be vulnerable to exploitation.

Step 5. Maintenance

5.1. Elaboration of operational management reviews - a process detailing the way both the operational part of the application and the infrastructure are managed.

5.2. Making periodic robustness checks of the application - monthly or quarterly checks should be done on both the application and the infrastructure to ensure that no new security risks are introduced and that the security level is still intact.

5.3. Ensuring Change Verification - After every change has been approved and tested, ensuring quality, it is essential that the change is verified to ensure that the security level has not been affected by the change.

In conclusion, it can be argued that web application security activities in the development of web applications should be distributed throughout its life cycle, taking into account OWASP recommendations and other international and national recommendations and standards. This will provide functional and secure web applications.

REFERENCES

[1] https://en.wikipedia.org/wiki/Web_application_security, last access 25.08.17

[2] "2012 Trends Report: Application Security Risks". Cenzic, Inc. 11 March 2012. Retrieved 9 July 2012, last access 12.09.17

[3] https://www.owasp.org/index.php/The_OWASP_Testing_Framework#A_Typical_SDLC_Testing_Workflow last access 12.09.17

[4] https://www.owasp.org/index.php/Category:OWASP_Application_Security_Metrics_Project, last access 12.09.17