Classification of Software Security Tools

Aleksandar Dimov and Vladimir Dimitrov [0000-0002-7441-253X]

Sofia University „St. Kliment Ohridski", Faculty of Mathematics and Informatics
1164 Sofia, 5 James Bourchier Blvd., Bulgaria

{aldi, cht}@fmi.uni-sofia.bg

Abstract. Software security is considered one of the most important quality characteristics in the modern digital society. Indeed, protection of data and software availability in cloud-oriented environment is needed in order to achieve user trust and reliably to meet their requirements. In order to better support software developers in provision of security enabled code, more studies of application of available tools that would help them is needed. In this paper, we are going to further define the sublayers of software security provision with respect to application into Software development life cycle and analyze and classify software tools with respect to their applicability to those layers. This will help professionals to better setup their security environment with respect to the particular security requirements that they have.

Keywords: Software Security, Software Development Process, Security Tools.

CEUR-WS.org/Vol-2933/paper28.pdf

1 Introduction

Cyber Security by the National Institute of Standards and Technology (NIST) [7] is "The ability to protect or defend the use of cyberspace from cyber-attacks." There are two aspects in cyber security: software and information protection. Depending on the focus, some authors talk about the software security, other about information security. Other similar aspects of cyber security are hardware and network security, but all of them are interconnected in the sense that cyber-attacks are based on some software vulnerability. For example, vulnerability in the embedded software in essence may be hardware or network vulnerability etc.

In any case, the most destructive and dangerous cyber-attacks are focused on the information and that is why the cyber security standards are focused on information security like that of ISO 27000 series of standards [8]. However, information vulnerability is achieved usually via software vulnerability.

Software security can be accomplished in two directions: secure software development, which is a NIST initiative [11], and software vulnerability protection. In both cases, software security professionals face a challenge when they need to choose and evaluate the proper tool to help them fulfil their everyday tasks.

Copyright © 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Main goal of this paper is to propose a classification framework for different software tools aimed at software and cyber security provision.

The structure of the paper is as follows: Section 2 is devoted to related works; Section 3 provides information about the background of our work; Section 4 describes the methodology of this research; Section 5 describes our classification of software security tools; and finally, Section 6 concludes the paper and states the directions for further research.

2 Related works

Software security is widely researched area both by academia and software industry. Many classifications attempt to streamline the efforts to provide security in different subdomains, like security vulnerabilities, popular attacks, development methods, etc. However not many works exist targeted specifically at software security tools. Mainly industrial surveys exist that are targeted towards commercial software tools. Some of them are listed below.

In [2] a classification of the so-called DevSecOps software (DevSecOps stands for Development Operations which includes secure development practices) categorizes tools into the following groups:

- Static Code Analysis Software.
- Container Security Software.
- Dynamic Application Security Testing (DAST) Software.
- Log Analysis Software.
- Penetration Testing Software.
- Static Application Security Testing Software.
- Vulnerability Scanner Software.
- Web Application Firewall Software.

In [4] a brief overview is provided which divides security tools into two main categories – security scanning and runtime protection tools. These categories are further classified as follows:

- Security scanning tools:
  - Static Application Security Testing (SAST) – examines the code of software, while it is not being executed as a whole.
  - Dynamic Application Security Testing (DAST) – checks whether the software system is vulnerable, by simulation of attacks towards it.
  - Interactive Application Security Testing (IAST) – similar to SAST, but makes the security scanning after build of the software.
  - Software composition analysis (SCA) tools – checks software systems for vulnerabilities in open source and third party components.
- Runtime Protection Tools (RPT):
  - Web Application Firewalls (WAF).

- Bot Management.
- Runtime Security Self-Protection (RSSP).

A comprehensive guide to classical software security tools that are also open source is provided in the book of Howlett [5].

Although not a classification, an interesting survey is presented in [6] that aims to measure the adoption of security tools among software developers. This work may be used in conjunction with ours to better study security from Software development life-cycle point of view.

There also exist some works aimed at scrutinizing security tools from particular category, e.g. [1], [3], [10], [13].

3 Background

As stated in the introduction, here we are going to classify according to the previously mentioned basic SDLC phases, various software tools, applications, guidelines and standards for audit, examination and assessment of information systems. This classification and analysis framework should help both cybersecurity professionals and trainees, former in their everyday tasks and the latter – to streamline better their learning path.

With respect to the classification, we are particularly interested to find out how are software tools that deals with security related to software systems lifecycle (SDLC). One of the fundamental models of SDLC is the waterfall model [9], and its main phases are as follows:

1. Requirement's definition – this is where the system context, goals and objectives should be defined by all stakeholders
2. System and software design – according to the requirements, create a plan for the development of the software system. This is done by defining the main system abstractions and relations among them. Usually, the result of software design should be a detailed software architecture of the software system.
3. Implementation and unit testing – this is when the code of the software is being elaborated into some program units.
4. Integration and system testing – all program units that complete system functionality should be integrated together into a working software system, which should be tested upon satisfaction of the requirements from phase 1.
5. Operation and maintenance – in a successful software system, this should be the longest phase in SDLC. This is when the system is deployed and delivered to its end users. Maintenance includes bug fixing, improvement, and development of new system features.

These phases may be considered enough for the purpose of our classification, as they appear in almost all SDLC models, even the agile methodologies. This way, our approach is not constrained or focused on specific development methodology and may be taken into consideration in various environments and software development efforts.

4 Methodology

In order to perform this survey, we have made a research on online portals supplying articles and information to help security professionals. This research was performed by search engines (Google and DuckDuckGo) with the following keywords:

- Cyber/Software/Information security portal
- Cyber/Software/Information Security blog
- Software vulnerability tools
- Exploits
- Vulnerability checking

The majority of the above-listed keywords led to results about cyber-security blogs and/or a plethora of commercial tools for strengthening security of both individuals and enterprises. Relevant to our research information in blogs, also leads to such tools. However, in this paper, from academicals point of view, we are going to consider only open-source tools that have little or small number of commercial alternatives.

Thus, this initial survey narrowed the results to only the information, residing into the following two leading security information portals that provide a number of resources for information and cyber security:

- Open Web Application Security Project (OWASP) [11];
- National Institute of Standards and Technology (NIST) [12].

For this reason in the rest of the paper and the description of software security tools that follow, we will consider mainly information taken from these two sources.

As already discussed in Section 3 of the paper, existing classifications of software security tools focus mainly on late phases of SDLC. Here we are going to extend them by inclusion of additional classification categories, for each SDLC phase. Security tools are going to be classified with respect to their application in these phases. Moreover, for the particular phases where already exist classifications, we consider them here. This way, the following main categories of software security tools, are considered:

- Tools applied during the requirements definition;
- Tools applied during design phase of software;
- Tools applied during implementation of software;

- Tools applied during testing (also called security-scanning tools [4]):
  - Static Application Security Testing (SAST);
  - Dynamic Application Security Testing (DAST);
  - Interactive Application Security Testing (IAST);
  - Software composition analysis (SCA) tools=
- Tools applied during maintenance and usage of the system:
  - Malware scanners;
  - Website security scanners;
  - Runtime protection tools, e.g. firewalls, application lockers, etc.
- Tools used for learning about security – this category is considered horizontal for the others, i.e. it is supposed that good education, higher levels of security will be reached into all phases of SDLC.

Some of these classification categories are subject of even more granular division of sub-categories, especially the ones that include many proprietary or commercial tools.

5 Software security tools overview

In this section, we briefly present the software security tools, as identified relevant information sources, as described into previous section. All tools have references into the OWASP projects web page, so we are not going to provide links about each of them here. Subsections here correspond to the classification shown above and tools are described accordingly. Summarization is provided at the end of the section.

5.1 Security tools applicable in Design phase

Security Pins is a platform that enables visualization of current investment in security features. This way it is applicable during design of software systems as it helps security professionals identify possible gaps, where security has not been taken into account and focus additional efforts there.

Pytm is framework that may be used to define the design of a system via predefined elements. Then, based on this, it may generate specific UML diagrams (Data Flow, Sequence) and this way show potential treats to the system. This tool uses a Python syntax, which makes it applicable also during development phase.

5.2 Tools applied during Development phase of SDLC

CRSF Guard and **CSRFProtector** are programming libraries, which help developers to defend their software systems against Cross-Site Request Forgery (CSRF) attacks.

CodePulse is a tool that supports testing of software by providing real time visualization of code coverage during black box testing.

Attack Surface Detector – as its name suggests this tool is used to quantify attack surface of web applications. This tool finds web applications' endpoints, their parameters and types, especially unused and unlinked endpoints that a potential attacker could exploit. This tool is applicable during testing phase of software development lifecycle.

Cornucopia is a card game, intended to help software engineers had better define security requirements. It does not depend on particular software development process. This way, this tool is applicable during requirements definition phase in SDLC.

Enterprise Security API (ESAPI) is another program library that helps software developers better to implement security both into new and existing applications. According to this ESAPI may be classified as a tool, applicable either during implementation or maintenance phase of software systems.

HTML Sanitizer (also called **Java HTML Sanitizer**) is a tool written in Java programming language that protects web applications against Cross-site scripting, included into third party HTMLs that are integrated into this application.

FindSecBugs (**OWASP Java Find Security Bugs**) is a tool for code audits of applications written in Java. It supports continuous integration as well as a number of frameworks and libraries and has many built-in security bug patterns. It is also possible to integrate it with popular Integrated Development Environments (IDEs) like Eclipse, InteliJ, Android Studio, etc.

Application Gateway is a reverse HTTP proxy that stands between the web application and the client and facilitates developers by allowing them to focus on coding the business logic and relieving them from the responsibility to implement authorization and session management. By mediating the communication with client, this tool also promotes architectural reasoning about the application and this way it may be considered applicable during design phase of SDLC.

5.3 Security testing tools

Amass is aimed at securing the existing and legacy infrastructure of an enterprise software system. It is able to identify potentially prone to attack organizational assets and this way helps mainly software security professionals, working into the maintenance and usage phases of SDLC.

Defectdojo is a tool aimed at vulnerability management and is applicable in different SDLC phases. It offers functionality that has a large amount of commercial and open-source representatives, which are outside the scope of our classification. However, by this reason, they are going to will be included as categories rather than specific software tools and applications.

Offensive Web Testing Framework (OWTF) is a platform that aims to increase efficiency or penetration testing. This is achieved via tools that help penetration testers to improve their performance and to raise the level of test coverage. OWTF should be applicable in the testing phase of SDLC.

Zed Attack Proxy (ZAP) is another software tool, aimed at penetration testing. It is however focused particularly on web applications. It is based mainly on testing of Man-in-the-Middle vulnerabilities.

Dependability-check – is a Software Composition Analysis (SCA) tool that check software dependencies against publicly available vulnerability records.

Dependency-track is a similar tool to Dependability-check, also classified as SCA, however with bigger capabilities. It checks security issues that may arise due to third party software components, either commercial or open source. It may be useful also for developer and security QA engineers. It also has features that raise notifications in case of system components that are not updated to their last version, which makes it applicable during maintenance phase by system administrators.

Android Security Inspector is a toolset that is used to search security vulnerabilities of existing Android applications. It has a good graphical interface, provides means for customization and is applicable for both DAST and SAST.

APICheck may be also considered as a DevOps tool as it provides environment for integration of tools for checking APIs. It also makes testing chain executions of such tools. It is designed as "a universal toolset for testing REST APIs, allowing you to mix and match the tools it provides, while enabling interoperability with third party tools". As it is designed as a DevOps tool, it is considered applicable also during development and deployment and maintenance phases of SDLC.

Mobile Audit is a SAST tool, for android applications; however, it is also augmented with malware detection, which makes it also suitable for the maintenance phase of SDLC.

Nettacker is tool that enables to facilitate the work of penetration testers. It automates collection of vulnerability information about a software system and generates appropriate scan reports into a set of common file formats (like HTML, JSON, CSV, etc.).

Purpleteam is a framework able to find security faults residing into a running web application and/or APIs. Upon finding a security issue it sends notifications about where and what it is. Purpleteam is a DAST tool and therefore lies into the testing phase of SDLC.

secureCodeBox is platform for execution of toolchains of security testing instruments. It is based on Kubernetes and this way is applicable as a SecDevOps tool.

5.4 Tools utilized during usage of software systems

AntiSamy is an API that ensures that user uploaded HTML and CSS files does not contain malicious issues. It ensures that users do not supply malicious HTML when for example they make forum post, manage their profile, etc. This way it provides security mechanisms for the software usage or maintenance phase of SDLC.

Bug Logging Tool (BLT) facilitates maintenance by providing means for issue tracking and management.

O-Saft (SSL advanced forensic tool) is an instrument aiming at testing of SSL properties and connection with a website. As its page states, "it's designed to be used by penetration testers, security auditors or server administrators".

5.5 Tools for education on security

Juice Shop is a vulnerable web application, full of different exploits. This way it is well suited for education of information and cyber security professionals. This way, this tool may be classified as applicable at all phases of SDLC, however its main benefit is in early phases, where project costs may be minimized if appropriate requirements are set and the corresponding design decisions lead architecture of the system that provides the necessary security level.

IoTGoat is an insecure firmware based on the embedded operating system OpenWrt and aims to facilitate learning of software developers for IoT application to find and avoid most common vulnerabilities of IoT devices.

Another similar project is **Node.js Goat**, which enables education on how to develop web applications that avoid the Top10 security risks, enlisted by OWASP.

Pygoat is a platform that have two purposes – (1) to educate software developers on how to write secure code and (2) to educate software testers on how to properly test for security issues.

Secure Coding Dojo is a security educational platform aimed at training software developers to find and avoid security bugs in software code. It offers lessons based on challenges in the form of attack/defense pairs. This tool is applicable in development phase of SDLC.

Security Shepherd is tool, which is aimed solely at education in the domain of information and cyber security. It is aimed at improving skills for penetration testing of software. It is based on Top 10 list of security threats; explains the main concept; and offers relevant security case studies. This tool is applicable at design and testing phase of SDLC. This classification considers the positive effect of education, when applied at early SDLC phases.

Snakes and Ladders is an educational game that promotes security challenges and risks and means to tackle them. It is also specifically targeted towards OWASP as it promotes also other project of this organization.

Cyber Scavenger Hunt is a website that aims to educate penetration testers.

DevSlop is a set of tools that aims to educate people on the practices of Secure DevOps (DevSecOps).

Honeypot is a repository aimed at identification of emerging attacks. Although not specifically targeted at education on security but also on attack prevention, there is no specific other category from our classification to put Honeypot in. It is actually a horizontal tool, applicable in all SDLC phases and for this reason, it is best to put it in the education category.

Table 1. Open source software security tools, with respect to their application in SDLC.

Requirements	Design	Implementation	Testing	Maintenance and usage
	Security Pins	CRSF Guard	Amass	AntiSamy
	Pytm	CodePulse	Defectdojo	BLT
		Attack Surface Detector	OWTF	O-Saft
		Cornucopia	ZAP	
		ESAPI	Dependability-check	
		HTML Sanitizer	Dependability-track	
		FindSecBugs	Android Security Inspector	
		Application Gateway	APICheck	
			Mobile Audit	
			Nettacker	
			Purpleteam	
			secureCodeBox	
Education				
Juice Shop, IoTGoat, Node.js Goat, Pygoat, Secure Coding Dojo, Security Shepherd, Snakes and Ladders, Cyber Scavenger Hunt, DevSlop, Honeypot, SamuraiWTF, Sectudo				

SamuraiWTF is a virtual machine, able to run on VMWare VirtualBox, which is configured to execute as a web penetration-testing engine. However, according to its creators, its main goal is to serve as a platform for training. For this purpose, it encapsulates a set of vulnerable applications and the tools necessary to learn how to assess various security risks associated with them.

Sectudo is a tool that should help to learn the most common security defects that are typical for mobile applications and their Server-side APIs. It includes a security-learning guide and two versions of the same web application – the one equipped with vulnerabilities and the other one – safe.

As seen from the Table 1, majority of the tools is targeted towards testing and education on security, closely followed by security tools, applicable during development. Relatively few tools are available during development phase of SDLC and very little or no tools are targeted at security in early software development phases. While it is not a big surprise, that most efforts were focused on ensuring security by testing, it is somehow upsetting that so small amount of tools is available during early phases of SDLC, which are considered crucial for development success and minimization of costs.

6 Conclusion

Software security is one of the most discussed topics by both academia and industry in the area of software development. This is the reason why there is wide variety of software tools aimed at provision of security at different levels. This fact makes it difficult for software development professionals to streamline and foster their efforts in ensuring security of developed applications.

In this paper, a classification of tools against main phases of software development is made, together with short description of these tools. We have considered only open source, non-commercial tools, which is the reason that all of the considered tools are available within the Open Web Application Security Project (OWASP) framework.

Future work is needed in order to augment the taxonomy proposed here in several directions, including, but not limited to:

- More tools should be included into the classification by study of alternative sources.
- Additional and possibly orthogonal classification categories – for example how is educational category related to others, what is its importance, etc. Are there other orthogonal categories of tools, applicable in all SDLC phases.

7 Acknowledgements

This research is supported by the National Scientific Program "Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES)", financed by the Ministry of Education and Science and the Sofia University "St. Kliment Ohridski" Research Science Fund project No. 80-10-74/25.03.2021 ("Data intensive software architectures").

References

1. Kumar Ch. 12 Online Free Tools to Scan Website Security Vulnerabilities & Malware, 2021, https://geekflare.com/online-scan-website-security-vulnerabilities, last accessed 24/04/2021.

2. Best DevSecOps Software. https://www.g2.com/categories/devsecops, last accessed 24/04/2021.
3. SourceForge, Software Composition Analysis (SCA) Tools. https://sourceforge.net/software/software-composition-analysis-sca, last accessed 24/04/2021.
4. Peterson J. Application Security Testing: Security Scanning Vs. Runtime Protection, 2020, https://resources.whitesourcesoftware.com/security/ast-application-security-testing, last accessed 24/04/2021.
5. Howlett, T. (2004). Open source security tools. Prentice Hall.
6. Witschey, J., Zielinska, O., Welk, A., Murphy-Hill, E., Mayhorn, C., & Zimmermann, T. (2015, August). Quantifying developers' adoption of security tools. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (pp. 260-271).
7. NIST, Information Technology Laboratory, Computer Security Resource Center, Glossary, last accessed 2021/04/16.
8. ISO, ISO/IEC 27001, Information Security Management, https://www.iso.org/isoiec-27001-information-security.html, last accessed 2021/04/16.
9. Sommerville, I. (2016). Software Engineering. 10th edition. Pearson Education.
10. Rahman, A. A. U., & Williams, L. (2016, May). Software security in devops: synthesizing practitioners' perceptions and practices. In 2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery (CSED) (pp. 70-76). IEEE.
11. OWASP, https://owasp.org, last accessed 24/04/2021.
12. NIST, Information Technology, Cybersecurity, https://www.nist.gov/cybersecurity, last accessed 24/04/2021.
13. Okun, V., Guthrie, W. F., Gaucher, R., & Black, P. E. (2007). Effect of static analysis tools on software security: preliminary investigation. In Proceedings of the 2007 ACM workshop on Quality of protection (pp. 1-5).