2020 13th CMI Conference on Cybersecurity and Privacy (CMI) - Digital Transformation - Potentials and Challenges | 978-1-7281-9056-3/20/$31.00 ©2020 IEEE | DOI: 10.1109/CMI51275.2020.9322704

A Survey and Comparison of Secure Software Development Standards

Armando Ramirez
Computer Science Department
University of Wisconsin-Parkside
Kenosha, WI, USA
ramir022@rangers.uwp.edu

Anthony Aiello
Computer Science Department
University of Wisconsin-Parkside
Kenosha, WI, USA
aiello@cs.uwp.edu

Susan J Lincke
Computer Science Department
University of Wisconsin-Parkside
Kenosha, WI, USA
lincke@uwp.edu

***Abstract*—There are standards, guidelines, and certifications for software security to help guide software development projects into becoming more securely written to comply with any regulations that may apply to the project. These best practices and standards include Common Criteria, The Open Group Architecture Framework (TOGAF), Security Assurance Maturity Model (SAMM), Building Security In Maturity Model (BSIMM), Application Security Verification Standard (ASVS), OWASP, and SAFECode, in addition to the national or international standards groups, PCI, NIST and ISO/IEC. In this paper, we focus on secure software development by surveying and comparing these methods and standards and discover which areas of the software development life cycle SDLC, that one or more could be applied to improve the security of a software application during its development lifecycle.**

Keywords— security standards, OWASP, CC, TOGAF, SAMM, BSIMM, SAFECode, risk, risk management, risk assessment.

I. Introduction

One of the most important issues and sometimes biggest challenges for companies is the development of secure software and implementing and managing security in the software development life cycle (SDLC) in order to make software more secure and adhere to standards and regulatory requirements [19] [20] [27]. There exists a set of standards, guidelines, and certifications for software security to help guide secure software development projects. We survey these to describe the strength of each of these and to provide an overview of where a standard may best be used. This paper focuses on secure software development and ways to incorporate best practices throughout the lifecycle of the software product under development. We compare some of the most widely used standards, the best practices and guidelines, through research and literature review. We also discuss areas of the development process these standards and guidelines may apply to for a software development project to become more compliant with its relevant security requirements and/or regulations. Our goal is to provide a survey of a selection of software security standards, as well as to determine popular aspects of risk evaluation.

Risk analysis literature exists for software design (e.g., [15]), security requirements (e.g., [16]), and safety standards (e.g., [17]); however, we are curious about which secure software development processes are adopted within industry security standards. Virtually no literature exists to date, listing and comparing many of the existing industry standards for secure software development. In this paper, we introduce a few of the most popular industry standards to compare a set of standards, so that the reader may have access to a quick view and gain an understanding of which guidelines they may follow to develop more secure software.

The scope of our paper focuses on providing a survey of commonly used standards for creating secure software applications. We are not aware of other papers that offer a survey, that contrasts coverage areas of these standards. This survey is useful if one wishes to comply with standards or produce a more secure software product.

II. Literature Review

There are many reasons that security standards are ignored by developers during software development. Organizations often have conflicting goals of development speed versus security, which discourages the use of secure software standards or even software development lifecycle standards. Gandhi et al. [4] states that through assurance controls and security standards, we can manage vulnerable software. Geer suggests simplifying software security technologies to require less time and security expertise [5]. This simplification allows for quicker adoption for processes and some technologies such as SDLs or maturity models; however, security expertise still needs to be a high priority despite the rapid changes required in software and hardware [6] [7]. Aranda et al. [8] suggest that small companies rarely implement requirements techniques that address risk because it demands a radical change in their processes and increases cost. Our paper provides engineers with a road map of where to acquire both simple and complex secure software standards, according to their needs.

It is recognized that there is a need for security standards. Farhan [19] points out the necessity of integrating security across the SDLC because traditionally addressing security was part of testing, which found bugs often too late. Second, a good standard is necessary to address a myriad of attacks such as SQL injection, command injection, buffer overruns, cross-site scripting, failure to handle errors, string formatting problems, neglecting change control, integer bugs, and poor usability. Fujdiak [20] recommends a two-pronged approach: proactive, which can help prevent, and reactive, which can help discover faults. Implementation of security standards can help to raise the quality of security in a software product and address these issues.

Research in many cases recommends security standards, without offering a description of the standards. Fujdiak [20] recommends a secure software development lifecycle (SSDLC) where each development phase adds in security considerations. They suggest adhering to the standards set from OWASP, ISO/IEC, and NIST but do not contrast standards to use for different phases of a SSDLC and do not provide coverage to the standards they recommend. Geer introduces different security standards in achieving standardized security implementations including BSIMM or SAMM [5]. While these papers recommend security standards, none of these papers provide a survey of available secure software standards.

Shan et al. [22] reports the results of a questionnaire among industry sectors and found two standards that are most applied in industry, ISO 2700X, and and for security analysis methodologies, Common Criteria (ISO 15408). They also provide a valuable table of standards that are used for specific sectors of industry. While they provide survey results of commonly used standards, they do not contrast or compare these standards.

Some papers have used one or more standards during their development. Popić [27] recognizes that two of the biggest challenges in the software industry are software quality, and software security, and discusses the MISRA-C standard, which focuses on the safe use of the C programming language in critical systems. Jefcoat [18] created a file erasing tool that became certified by Common Criteria. Farhan [19] recommends NIST 800-64 for the SDLC and comments that OWASP – CLASP is simple and easy to implement. While these papers describe a singular or a pair of standards, they do not contrast them.

We also considered research related to software risk analysis and standards. Risk analysis literature exists for software design (e.g., [15]), security requirements (e.g., [16]), and safety standards (e.g., [17]); however, we are curious which risk analysis processes are adopted within industry security standards. In this paper, we contrast risk recommendations by these standards.

We evaluate standards related to maturity models, secure software development and international standards. We hope readers find our paper helpful in finding security standard(s) and source(s) that suit their needs.

III. Survey of Standards

In this section, we perform a survey and comparison of security standards for software development and describe a few of the different security standards listed in Table 1. Here we provide an overview of some of the most popular software standards used in industry from a general perspective.

A. Maturity Models

Maturity models help organizations introduce and evolve their software security process in a guided step-by-step manner. We examine the following maturity models, which are open community projects, and widely used in the IT industry: Building Security In Maturity Model (BSIMM) and Security Assurance Maturity Model (SAMM). These documents provide an overview by listing but not describing processes in detail.

TABLE I. SAMPLE OF STANDARDS AND THEIR APPLICATIONS

Standard	Application
ISO/IEC	62443-4-1 - Software in industrial control systems
	27034 – Application security standard – aviation, defense, financial, medical
	27034-3 Application management process
	27034-4 Application security validation
	27034-5 Protocols and application security control data structure
	61508 - Automated systems functional safety
	61508-3 - Software requirements for 61508 Standards
	26262 – Auto industry safety & security standards
	27002:2013 - Organizational security guidelines
	15026 - Assurance cases for software and systems
	41062:2019-2 - Software acquisition standards
	62304 – Medical device SDLC requirements framework/regulations
	15408 – Common Criteria computer security certification
NIST	800-37 - Risk Management Framework
	800-53 - Federal Information Security Management Act, FISMA Requirements
	SP 800-128 - Security: Configuration Management of Information Systems
MISRA - C/C++	Motor Industry Software Reliability Association C/C++ Secure software
PA-DSS	PCI Security Standards Payment Application Data Security Standard [30]
Maturity Models	SAMM – Security Assurance Maturity Model [9]
	BSIMM - Building Security In Maturity Model [12]
Design, Code, Test	TOGAF - The Open Group Architecture Framework [14]
	CC - Common Criteria [13]
	OWASP Application Security Verification Standard ASVS [10]
	SANS Institute

SAMM defines this standard as “open framework to help organizations formulate and implement a strategy for software security” [9], and BSIMM defines it as “a reflection of the current state of software security” where their primary aim is to “wider software security community plan, carry out, and measure initiatives of their own” [12]. SAMM and BSIMM both define four categories called ‘business functions’ and ‘domains’ respectively, where each category is further defined into three practices. Both models also define three maturity levels, but SAMM has an additional implicit starting point at zero where activities have not been started.

The practices and activities presented in these models differ slightly in the approach that each model takes to achieve a higher maturity level. For example, SAMM presents the activities along with the measurement of performance, associated assurance benefits, personnel involved, and costs. On the other hand, BSIMM presents the security activities along with the people involved and performance measurement.

B. Industry Groups for Software Development: Design, Code, Test Models

Some standards emphasize these stages in detail: design, code and test, while still requiring aspects of security requirements. Coding and design standards define how security requirements should be included into the software development process. The requirements include security goals, purpose, target audience, regulations, and policies. Testing of software helps to verify and achieve effectiveness for newly integrated requirements. In this section, we introduce the following standards that are used in the private and public sectors: CC, TOGAF, ASVS, SAFECode, NIST, and ISO/IEC.

1) *Common Criteria*

Common Criteria (CC) is now an international standards framework (ISO/IEC 15408) for certifications in computer security. It originated from a combination of regulations and standards mainly from three places, TCSEC – US DOD NIST [21], CTCPEC – Canada & US, and ITSEC – European standards from the early 1990s. CC is a standard that "provides a common set of requirements for security functionality of IT products [such as hardware, software, and/or firmware] and for assurance measures applied to these IT products during a security evaluation" [13]. Moreover, CC introduces two specifications, which contribute to CC's flexibility in usage: Security Targets (STs) and Protection Profiles (PPs). ST and PP specifications allow creation of a high-level of abstraction guideline that define threats, problems, and security requirements. These specifications are not meant to have detailed description or a long description of detail operation. For this reason, CC encourages readers to use other security standards in conjunction for a more effective way to manage risks and facilitate future security evaluations by the organizations.

Vendors who certify products using Common Criteria assure that these products went through a strict standard process of specification and evaluation. These are high quality guarantees from these certified products. CC is used as a base for government certification. The Common Criteria Recognition Arrangement is an international agreement made up of Common Criteria and Common Methodology for IT Security Evaluation (CEM), respectively. Each country uses their own processes to certify, and member countries certifications are acknowledged after they have been evaluated by a collaborative Protection Panel (cPP). Goals include to improve security enhanced products and their availability, and ensure that these evaluations are done to the highest standards consistently. Each product evaluation results in an Evaluation Assurance Level (EAL) between 1 and 7. Lastly, the guidance of operation section provides examples of tests and guidance in developing tests that can be tailored for an IT product [18].

2) *The Open Group Architecture Framework (TOGAF)*

TOGAF® is a "proven Enterprise Architecture methodology and framework … to improve business efficiency". TOGAF® focuses on architecture definition for large-scale interconnected enterprise systems, addressing business, data, application, and technology domains [14]. TOGAF®'s 30-page document, "Integrating Risk and Security within a TOGAF® Enterprise Architecture" [31], lists security concerns corresponding to each of the four different domains and provides guidance for risk management.

3) *Software Assurance Forum for Excellence in Code (SAFECode)*

The practices defined in SAFECode cover a wide variety of the software industry needs with the intention to be implemented into organizations' SDL [11]. Practices that SAFECode focuses on are in the areas of design, coding, and testing. SAFECode enhances the practices by providing methods and tools for organizations to receive guidance and feedback. They also have Common Weakness Enumeration (CWE) references to strengthen security discussions, descriptions, identification, and implementation.

4) *Application Security Verification Standards (OWASP ASVS)*

OWASP Open Web Application Security Project foundation is a non-profit organization that supplies developers with current standards for secure coding practices. OWASP's ASVS is an open community security standard used to normalize security controls during software requirements, design, development, and testing [10]. For requirements, an activity checklist supports the following security verification levels: level 1 is meant for all software, level 2 is for applications that contain sensitive data, and level 3 is for the most critical applications. A quick reference guide gives users detailed actions to take during the development and coding processes of building an application. This standard also has a list of testing tools: OWASP Mobile Security Testing Guide (MSTG) for both Android and iOS is a security model listing security requirements, and a base template for setting up automated testing for mobile applications. Included is a set of insecure apps called a Hacking Playground that demonstrates the vulnerabilities they discuss [10].

C. *Standards Institutes*

Standards institutes tend to focus on specific industries that have detailed security needs. These needs impact software development in security requirements and testing.

1) *National Institute of Standards and Technology (NIST)*

The National Institute of Standards and Technology (NIST) is a department of the US government that has a very large set of regulations, guidelines, and rules about software development and security, that are continually updated and published. There are vast amounts of free documents that are guidelines of best practices that are used for integrated control systems (ICS), which include supervisory control and data acquisition (SCADA), distributed control systems (DCS), as well as many other areas of industry. These systems are found in such industries as oil, gas, chemical, electric, water, transportation, wastewater, pulp & paper, food and beverage, and manufacturing. These document best practices to securing these systems. They also identify known threats, review system topologies & architectures, and give countermeasures to address these situations. There are regulations (e.g., FISMA) and specific guidelines that companies, vendors, and anyone doing business with the US government must comply and adhere to [21].

2) *ISO/IEC*

The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) are committees responsible for developing, promoting, and maintaining standards worldwide for industry, particularly communications and information technologies as well as sets and govern the standards for electric and electronics. While many of these standards documents cost a fee, some may be downloaded for free at [26].

a) ISO/IEC 15026-2:2011: The standard ISO/IEC 15026-2:2011 is part of the 15026 family of standards, and defines the methodology to develop an assurance case for systems and software, which is one or more claim(s) about that product supported by evidence, reasoning and justification [23]. Assurance cases are used to support the claims in areas such as its software security, reliability, operability, and safety.

They may also be referred to a specific named case e.g., reliability case, maintainability case. This standard is reviewed every five years and provides an overview about the elements and requirements needed to develop an assurance case ant its components such as claim, coverage, argument characteristics, evidence, associated assumptions, and justifications that are required. Assurance cases are an important part of the software and systems engineering. An assurance case can also be applied to safety or security of a software product.

b) ISO/IEC 41062: This standard helps those that are acquiring software for their organization. It is a recommended practice and comprehensive standard for software acquisition [24], that describes current best practices for acquiring software systems that are available e.g., software as a service (SaaS), off-the-shelf (OTS) software, and custom-developed software. Eight steps and twelve checklists are outlined with detailed actions for acquiring quality software including a planning and acquisition strategy, preparing contract requirements, selecting software vendors, and providing assurances such as software safety and software information security assurances.

3) PCI PA-DSS

The Payment Card Industry (PCI) Security Standards Council, in addition to standardizing how payment cards shall be used within business through its Data Security Standard (PCI DSS), includes a standard for Payment Application Data Security Standard (PA-DSS). This standard certifies applications that are used to accept and process card payments, particularly for third party use. [30]

The standard consists of a set of requirements and tests, which can be used to certify the software by a Payment Application Qualified Security Assessor (QSA). Assessors perform tests outright and through reviewing required instructional and training documentation, in areas including encryption, authentication, logging, information storage, wireless, network and remote access, and process verification.

IV. Comparative Analysis

Table II and III contrast the stages covered by each standard, as well as the risk management support for each.

A. Developing Processes

The two evaluated maturity models are similar in some of the activities they present. SAMM and BSIMM models have a very effective way to measure progress. As shown in Table 2, both models have a heavy focus on risk assessment through the activities that are required [9] [12]. For instance, abuse cases are activities recommended in both models that are easy to develop, demonstrate how attacks affect the software, and aid to create a more well-rounded risk assessment. Moreover, both models require activities to create threat models for the specific application and technology. Lastly, the multi-step process in these standards allow organizations to start at low level and reach higher maturity levels using a step-by-step approach with a focus on process improvement.

B. Mature Processes

The NIST and ISO standards provide extensive documentation for mature organizations or severe applications. NIST is often compared to ISO 27001, the framework for information security management systems, because of their similarities. Both NIST and ISO publications follow a strict structure. ISOs numbering system refers to a base number as a "family" of documents containing sub documents related to specific areas these policies should be integrated into. Each publication number and sub-number contains extensive documentation and procedural steps to take for each stage of the project's development lifecycle. ISO is more risk focused while NIST focuses more on security.

TABLE II. STANDARDS COVERAGE BY DEVELOPMENT STAGE

Stage	**BSIMM**	**SAMM**	**ISO/IEC**	**NIST**	**MISRA**	**ASVS**	**CC**	**TOGAF**	**SANS**	**PA-DSS**
Process	X	X	X	X		X	X	X	X	X
Training	X	X	X	X	X	X	X	X	X	X
Requirements			X	X	X	X	X	X		X
Design			X	X	X	X	X	X		
Develop Code					X	X	X		X	
Test			X	X	X	X	X	X		X
Deploy			X	X	X	X	X	X		
Review			X	X	X	X	X	X		
Certify*			Organization				Product	Person	Person	Product

*Certification exists for a person (TOGAF, SANS), product (CC, PA-DSS), or organization (ISO 20000)

TABLE III. RISK MANAGEMENT IN SECURITY STANDARDS

Risk Requirements	**SAMM**	**BSIMM**	**TOGAF**	**CC**	**ASVS**	**SAFE-Code**	**NIST**	**ISO**
Risk Training	X		X				X	X
Threat Identification	X	X	X	X	X	X	X	X
Classification of Data and Software Assets	X	X	X				X	X
Specified and Detailed Coding Threats					X	X		
Risk Prioritization	X	X	X				X	X
Baseline Mitigation for Known Risks	X		X				X	X
Risk Monitoring			X				X	X

C. Coding

SANS, OWASP, and TOGAF provides specific coding examples and known code vulnerabilities and pitfalls to avoid in a few languages and are excellent tools for the developer in understanding how to write secure software applications. They are popular tools used for managing common code vulnerabilities and threats that every developer should have in their knowledge base. ASVS and SAFECode describe detailed software-oriented threat lists, as opposed to emphasizing the risk process [10][11]. Both standards specify a list of coding practices and requirements that should be implanted in software (e.g., string and buffer functions, input/output validation, cross site scripting defenses, cryptography). The approach that these standards take in handling risks are somewhat different. ASVS lists a series of checkpoints aiding the user to fulfil the requirements that revolve around the application. On the other hand, SAFECode recommends coding practices and a general approach for implementation, but also includes verifications and references that allow the users to fully understand and mitigate threats.

D. Testing

Secure development requires control implementation and verification. SAFECode and ASVS recommend the following testing techniques to ensure effectiveness of requirements and coding practices: attack surface, fuzz and robustness testing, penetration testing, static analysis tools, among others. BSIMM focuses on penetration testing, aiding developers to find problems using testing tools, perform deep-dive analysis, and customize testing tools.

E. Certification

CC offers certification for products that need to meet strict regulations and compliance, while PA-DSS is a product certification specific to payment card processing. TOGAF and SANS certifies individuals and developers for their knowledge of secure software development processes according to standards.

F. Focus on Risk Processes

Some standards provide guidance on the risk process; others provide specific threat models applicable to code; yet others point to external standards for threat models. TOGAF provides a general process without a specific model, whereas CC provides a threat model without specifying a risk process. Both refer to external methodologies and standards to focus on specific risks in the IT product while keeping a broad scope for the whole project.

All security standards specify threat identification, which is one of the first steps for risk assessment requirements. Unlike CC or TOGAF, security standards such as SAFECode, SAMM, and BSIMM recommend the use of misuse and abuse cases to tailor the risk assessment for specific applications.

Table 4 summarizes how these standards focus on risk. The first aspect considered for these security standards is risk training; SAMM and TOGAF have the more complete spectrum of requirements for risk management training with SAMM providing a software-oriented perspective and TOGAF is more focused on enterprise architecture.

The OCTAVE process for risk assessment includes risk identification (defining assets, threats, vulnerabilities); risk analysis and prioritization; and risk treatment [29]. The risk process is useful because it can be applied to any IT project; however, when standards (e.g., CC and TOGAF) do not provide general risk guidance OCTAVE may be helpful for inexperienced users.

Classification of data and software assets are complementary to risk prioritizations because it adds more relevance and weight to the risks. Moreover, asset classification in software development aids in the threat identification because organizations can identify more clearly the risks revolving assets and how the organization, users, and customers are affected.

An important risk requirement not heavily enforced in these security standards is baseline mitigation for known risks. Establishing a baseline mitigation for known risks enables the users to have their work checked against some of the common risks involved during development but also facilitates the process in identifying risks in future projects. Only TOGAF, SAMM, and NIST introduce the concept of mitigations for known risks.

TABLE IV. RISK METHODOLOGY BY STANDARD

Standard	**Risk Methodology Characteristics**
SAMM	Multiple-step risk process. Build and maintain application-specific threat models. Develop attacker profile from software architecture. Develop and maintain abuse-case models per project. Evaluate risk from third-party components. Elaborate threats models with compensating controls.
BSIMM	Multiple-step process. Classify and inventory data according to security goals Identify potential attackers. Build attack patterns and abuse-cases. Create technology-specific attack patterns. Build and maintain top N possible attacks list. Create and use automation to mimic attackers. Implement testing techniques/tools (penetration test focus) Includes active team to develop new attack models.
CC	Generalized risk for IT products. Define protective profiles and security targets. Identify threats to IT products & operation environment Enforce security rules, procedures, or guidelines.
TOGAF	Generalized risk for projects. Define threats based around data, services, unauthorized use. Assess risk based on the risks' effect and frequency. Classify risks based on impact to the organization. Mitigate risk and determine residual risk. Monitor risk.
ASVS	Discusses code-related risks. Provide a lists of security requirements applications should implement. Identify all application components, e.g., libraries, external systems. Verify a high-level architecture for the application is defined. Verify that all application components are defined in terms of the business and security functions provided. Verify centralized implementation of security controls. Verify that components are segregated via security controls. Verify that all applications components, libraries, modules, frameworks, platforms, and operating systems are free from known vulnerabilities. Implement testing techniques and tools.
SAFE Code	Standard Focuses on Threat Model (i.e., risk identification). Implement secure coding practices. Identify threats using techniques/methodologies: STRIDE, misuse cases, brainstorming, and/or threat library. Evaluate and prioritize risks using a risk rating system; mitigate according to priority. Validate using testing techniques and tools.

V. Conclusion

We have examined and compared common business model examples and frameworks for building security in and improving software security in the business processes. We conclude that many standards might not cover all the security requirements for secure software development when used individually. Instead, a process for creating secure software relies on implementing more than one of the standards, especially to conform to regulation or for certification of a secure software application.

References

[1] F. H. Shezan, S. F. Afroze and A. Iqbal, "Vulnerability detection in recent Android apps: An empirical study," 2017 International Conf. on Networking, Systems and Security (NSysS), IEEE, 2017, pp. 55-63. doi: 10.1109/NSysS.2017.7885802.

[2] Health and Human Services, "Health Information Privacy - HIPAA," HHS.gov, 18-Dec-2015. [Online]. Available: https://www.hhs.gov/hipaa/index.html.

[3] European Union, "EU GDPR Information Portal," EU GDPR Portal. [Online]. Available: https://www.eugdpr.org/.

[4] R. A. Gandhi, K. Crosby, H. Siy and S. Mandal, "Gauging the Impact of FISMA on Software Security," in Computer, vol. 47, no. 9, IEEE, pp. 103-107, Sept. 2014. doi: 10.1109/MC.2014.248.

[5] D. Geer, "Are Companies Actually Using Secure Development Life Cycles?," in Computer, vol. 43, no. 6, IEEE, pp. 12-16, June 2010. doi: 10.1109/MC.2010.159.

[6] J. F. Dhem and N. Feyt, "Hardware and software symbiosis helps smart card evolution," in IEEE Micro, vol. 21, no. 6, pp. 14-25, Nov/Dec 2001. doi: 10.1109/40.977754.

[7] J. P. Bowen, M. Hinchey, H. Janicke, M. Ward and H. Zedan, "Formality, Agility, Security, and Evolution in Software Development," in Computer, vol. 47, no. 10, pp. 86-89, Oct. 2014. IEEE. doi: 10.1109/MC.2014.284.

[8] J. Aranda, S. Easterbrook and G. Wilson, "Requirements in the wild: How small companies do it," 15th IEEE International Requirements Engineering Conf. (RE 2007), IEEE, 2007, pp. 39-48. doi: 10.1109/RE.2007.54

[9] OWASP, "OWASP SAMM Project," [Online]. Available: https://www.owasp.org/index.php/OWASP_SAMM_Project

[10] OWASP, "OWASP: Application Security Verification Standard Project," OWASP. [Online]. Available:https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project

[11] SAFECode. [Online]. Available: https://safecode.org/publication/SAFECode_Dev_Practices0211.pdf.

[12] BSIMM, "Building Security In Maturity Model | BSIMM," Building Security In Maturity Model|. [Online]. Available: http://www.bsimm.com

[13] Common Criteria, "Common Criteria," Publications : New CC Portal. [Online]. Available: https://www.commoncriteriaportal.org.

[14] Open Group, The TOGAF® Standard, V. 9.2, 2018. Available: www.opengroup.org.

[15] D. Verdon and G. McGraw, "Risk analysis in software design," in IEEE Security & Privacy, vol. 2, no. 4, pp. 79-84, July-Aug. 2004. doi: 10.1109/MSP.2004.55.

[16] N. Laoufi, "From Risk Analysis to the Expression of Security Requirements for Systems Information," 2015 Fourth International Conf. on Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec), IEEE, 2015, pp. 84-89. doi: 10.1109/CyberSec.2015.25.

[17] D. R. Wallace, D. R. Kuhn and L. M. Ippolito, "An analysis of selected software safety standards," in IEEE Aerospace and Electronic Systems Magazine, vol. 7, no. 8, pp. 3-14, Aug. 1992. doi: 10.1109/62.151140.

[18] K. Jefcoat, "What is Common Criteria Certification, and Why Is It Important? - Blancco", Blancco.com, 2018. [Online]. Available: https://www.blancco.com/blog-what-is-common-criteria-certification-why-is-it-important/

[19] A. R. Shehab Farhan and G. M. Mostafa Mostafa, "A Methodology for Enhancing Software Security During Development Processes," 2018 21st Saudi Computer Society National Computer Conf. (NCC), 2018, pp. 1-6, doi: 10.1109/NCG.2018.8593135.

[20] R. Fujdiak et al., "Managing the Secure Software Development," 2019 10th IFIP International Conf. on New Technologies, Mobility and Security (NTMS), pp. 1-4, doi: 10.1109/NTMS.2019.8763845.

[21] NIST, "National Institute of Standards and Technology," NIST, 17-Jun-2020. [Online]. Available: https://www.nist.gov/. [Accessed: 20-Jun-2020].

[22] L. Shan, B. Sangchoolie, P. Folkesson, J. Vinter, E. Schoitsch and C. Loiseaux, "A Survey on the Application of Safety, Security, and Privacy Standards for Dependable Systems," 2019 15th European Dependable Computing Conf. (EDCC), Naples, Italy, 2019, pp. 71-72, doi: 10.1109/EDCC.2019.00023.

[23] IEEE Standard--Adoption of ISO/IEC 15026-2:2011 Systems and Software Engineering--Systems and Software Assurance--Part 2: Assurance Case," in IEEE Std 15026-2-2011 , vol., no., pp.1-28, 11 Oct. 2011, doi: 10.1109/IEEESTD.2011.6045293.

[24] ISO/IEC/IEEE International Standard - Software engineering - Recommended practice for software acquisition," in ISO/IEC/IEEE 41062:2019(E) , vol., no., pp.1-56, 20 Feb. 2019, doi: 10.1109/IEEESTD.2019.8645777.

[25] A. Hazeyama, "Survey on Body of Knowledge Regarding Software Security," 2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2012, pp. 536-541, doi: 10.1109/SNPD.2012.64.

[26] ISO (International Organization for Standardization), Publicly Available Standards. [Online]. Available: https://standards.iso.org/ittf/PubliclyAvailableStandards/index.html. [Accessed: 28-Jul-2020]

[27] S. Popić, G. Velikić, H. Jaroslav, Z. Spasić and M. Vulić, "The Benefits of the Coding Standards Enforcement and it's Influence on the Developers' Coding Behaviour: A Case Study on Two Small Projects," 2018 26th Telecommunications Forum (TELFOR), 2018, pp. 420-425, doi: 10.1109/TELFOR.2018.8612149

[28] ISO, "International Organization for Standardization," ISO, 09-Jan-2020. [Online]. Available: https://www.iso.org/. [Accessed: 04-Aug-2020].

[29] C. Woody and C. Alberts, "Considering Operational Security Risk during System Development," in IEEE Security & Privacy, vol. 5, no. 1, pp. 30-35, Jan.-Feb. 2007. doi: 10.1109/MSP.2007.3

[30] [30] PCI Security Standards Council, Payment Card Industry (PCI) Payment Application Data Security Standard, v 2.0, Oct 2010, URL: https://www.pcisecuritystandards.org/minisite/en/docs/pci_pa_dss_v2-0.pdf.

[31] Open Group, Integrating Risk and Security within a TOGAF® Enterprise Architecture, 2016, Available: www.opengroup.org.