REVIEW PAPER

SECURE SOFTWARE DEVELOPMENT: INDUSTRIAL PRACTICE - A REVIEW

By

HENRY O. NWAETE

Northcentral University, San Diego, California.

Date Received: 03/05/2022 *Date Revised: 06/07/2022* *Date Accepted: 10/07/2022*

ABSTRACT

The current state of application assets with respect to their development, functionality, scalability, user friendliness, and compatibility with legacy systems has witnessed an unprecedented degree of positive improvements. This induced increase in productivity and value has been a product of the technological innovations within and around the software development landscape. Owing to specific software development practices including software reusability, Object Oriented Programming (OOP), encapsulation, and portability, all sectors of the economy have come to embrace software products that have helped to drive business transactions. Nonetheless, the proliferation of software which has driven up the velocity, veracity and volume of data associated with transactions has become a goldmine for grabs. Hackers and adversaries alike have thus capitalized on this development to exploit the potential threats and vulnerabilities associated with software products. Insecure software is global issue, and one that impacts individuals, organizations and governments. Data loss is both a security and privacy issue, with compliance, regulatory and legal concerns, and bad actors are relentless in their efforts to steal, deface, alter/manipulate, destroy, and compromise software systems. Organizations should therefore embrace secure code principles, threat modeling, and institute a Secure Software Development Lifecycle (SSDLC) practice that will aid the embedment of security into the development phase, to contain the risks, threats, and vulnerabilities that are inherent in software development. This paper demonstrates an effort to provide and arm organizations with the necessary tools, processes, and mechanisms that can be leveraged to combat cyber-threats and enforce Information Assurance (IA) within and around enterprise application assets. Beginning with an overview of the contemporary software development practices witnessed in diverse organizations, including financial, energy, aviation, commerce, nuclear, defense, and several other Critical Infrastructure (CI) organizations, the tenets of a composite, structured and robust. SSDLC has been presented to promote a defense-in-depth security for enterprise organizations.

Keywords: Security-by-Design, Threat Modelling (TM), Architectural Review, Penetration Testing (PT), Static Code Analysis, Web Application Monitoring.

INTRODUCTION

All sectors of the economy are equal beneficiaries of the astronomical developments in software development. Innovations in software development, including encapsulation, software reuse, Application Programming Interface (API), framework and library inventories have aided the generation of functionalities that could be leveraged to virtually achieve anything (Kiswani et al., 2018; Pathak, 2018; Schoeni, 2015). Software Defined Networking (SDN), hypervisors (virtualization), ubiquitous and embedded devices, and Internet of Things (IoT) have all contributed to the unprecedented developments witnessed across the globe (Bhukya & Pabboju, 2016; Ren

et al., 2016; Van Rossem, 2018, Wang et al., 2019).

Nonetheless, the aforementioned developments have constituted an increased attack surface, summing up to a plethora of cybersecurity issues. Given the diverse range of vulnerabilities associated with application functionalities, adversaries have continued to leverage these vulnerabilities to compromise application systems (Williams et al., 2018). The impact of cyberattacks could be devastating, especially when the target is a critical infrastructure. This has been evident in the aviation, energy, financial, maritime, power, and several other critical infrastructure sectors, examples are the hack on Federal Bureau Investigation (FBI) enterprise portal, the breach on German pharma and tech firms by a group of Chinese hackers, and the Ghostwriter cybercampaign by Russia targeting EU elections (Allahar, 2019; CSIS, 2022; Kriebel et al., 2018).

Cyberattacks are pervasive and could impact any entity, including individuals, organizations and governments. The number of cyberattacks executed so far cannot be accounted for as several organizations have failed to disclose the incidents they have encountered. However, the disclosed cyber incidents have failed to persuade individuals, organizations, the governments and all stakeholders to increase their cybersecurity efforts to be robust enough to promote Information Assurance (IA) and contain cyberattacks (Banerjee et al., 2017). Despite the potential impacts of cyberattacks, organizations still lag in cybersecurity implementations, owing to the lack of cybersecurity awareness and understanding, and limited fund allocated for cybersecurity on the part of senior management (Vidas et al., 2018). Furthermore, the limited number of skilled cyber personnel with the capacity to steer the organization away from cyber pitfalls have also contributed to the deficiencies in software development.

The need for robust cybersecurity postures within organizations cannot be over emphasized, and the software development stage seems to be the best place to initiate the awareness, standards and policies. Organizations should be cognizant of the impacts of cyber breaches, ranging from losses in revenue, productivity, goodwill, disclosure of sensitive information (proprietary data), intellectual property theft, and customer dissatisfaction (IT Governance, n.d). Given the classification of data processed by critical infrastructure organizations, including internal, sensitive, confidential, High Business Impact (HBI), Medium Business Impact (MBI), payment card, healthcare, and other Personal Identifiable Information (PII), organizations should be conversant with the associated laws and frameworks in order to remain compliant with all that are applicable (Fernández-García et al., 2018; Fernandes et al., 2018).

Organizations should be compliant with laws such as Health Insurance Portability and Accoutability Act (HIPAA), Federal Information Security Management Act (FISMA), Defense Federal Acquisition Regulation Supplement (DFARS), New York State Department of Financial Services (NYDFS), General Data Protection Regulation (GDPR), and regulations like Payment Card Industry Data Security Standard (PCI-DSS), and ISO/IEC 270000 series in order to secure the organization's software assets (Cybersecurity & Infrastructure Security Agency, 2020; The United States Department of Justice, 2021; Silva et al., 2018). Conversely, standards and publications including NIST SP 500-304, NIST SP 800-115, Open Web Application Security Project (OWASP), NIST SP 500-320, NIST SP 800-111, NIST SP 800-165, NIST SP 800-142, NIST SP 800-192, and NIST SP 800-166 should be embodied in the organizations Secure Software Development Lifecycle (SSDLC) (Al-Amin et al., 2018; NIST, 2010, 2012, 2016; OWASP, n.d). Organizations should also be aware that a default on any of the enumerated laws comes with stiff penalties, including fines and litigations. Thus, being compliant with all the applicable laws and regulations would amount to the implementation of SSDLC, including the execution of Threat Modelling (TM) Penetration Testing (PT), architectural reviews, static code analysis, and Web application monitoring (Dayanandan & Kalimuthu, 2018; Microfocus, n.d).

1. Current Software Development Landscape

Within the current software development arena, there is a diverse range of methodologies and frameworks from which organizations can choose from. Most of the

available frameworks, including Lean, Agile, Scrum, Kanban, Extreme Programming (XP), Feature Driven Development (FDD), Rapid Application Development (RAD), Dynamic System Development Model Methodology (DSDM), and Spiral, all have their unique attributes that could be further customized to suit the needs of individual organizations (Gartner, 2021; Musa et al., 2015).

1.1 Agile

From the aforementioned pool of frameworks and methodologies, the agile software development framework tends to attract more attention among developers. As represent in Figure 1, several organizations have deployed agile mechanisms towards their development efforts. To an extent, this has been due to the functional benefits offered by the agile development framework. Given the need to deliver software on time and on budget, drive revenues, and satisfy customers with a quick delivery and improved turnaround time, the Agile framework has been one of the frameworks that is best suited for the task (Camacho et al., 2016). Using agile, organizations can begin to plan their proposed implementations through project division and allocation of individual tasks. The projects are further subsumed into biweekly sprints that can be utilized in the delivery of a fully working software. Agile is quite a collaborative framework that prompts for daily standup meetings known as scrums. The scrums provide an avenue for developers to share their ideas, communicate updates on of their tasks, present any roadblocks, and get feedbacks from their peers with respect to any constraint they might be facing.

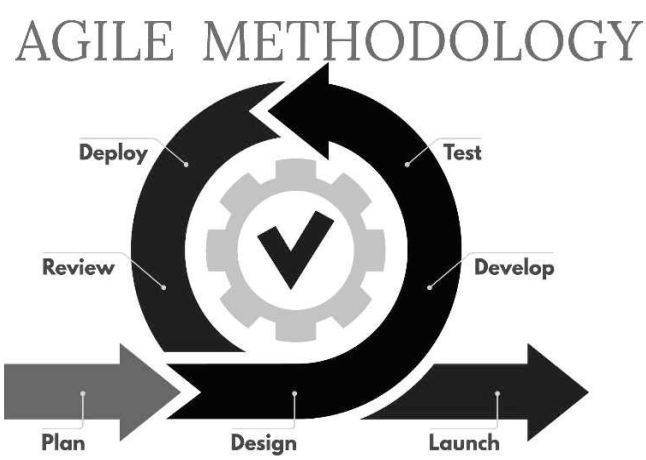

Figure 1. Agile Software Development Framework

The design phase presents the architecture of a project, workflows, potential algorithms/protocols, and user interface. This phase also serves as opportunity to elicit the necessary information from stakeholder through requirement specification. Considerations of the dependent platforms/independent for the project are also discussed during this phase. Agile code development leverages most of the tenets of object-oriented programming. Software/code reuse in agile makes it easier for developers to generate working applications within a short period of time. The use of libraries, frameworks and other software artefacts equally permit the quick turnaround time observed in agile software development methodology.

Unit tests are generally performed to validate the working capacity of codes. Using Integrated Development Environments (IDE), developers would normally run their codes, sometimes with teammates, and this provides an avenue to test if the software is running or needs to be debugged. To formally test the quality of software, Quality Acceptance Test (QAT) will check the software for flaws and bugs and ensures that the software meets the organization's standard.

User Acceptance Tests (UAT) become the next phase of tests once the QAT tests have been concluded. A small group of people, including the end users and other relevant stakeholders would normally be engaged during UATs, and the objective is to validate if the software performs as intended or specified during the requirement gathering phase.

The completion of the test exercises marks the beginning of the deployment phase. Armed with the knowledge of a Go-live date, developers would initiate the packaging of software assets for deployment. During software deployments, it is imperative to ensure that certified and good quality codes are deployed. Thus, the QAT or the UAT environment will be specified for migration. Software and database migration to production is meant to lift the authorized codes from the test environment to the

production environment.

2. Evolving Threats and Vulnerabilities

With all the prowess of the agile development framework and other related frameworks, it is important to note that the considerations for the security and privacy of software and the data associated with it are either limited or completely missing. This is not surprising given that agile and other frameworks such as scrum, XP, RAD, and FDD consider security as a Non-Functional Requirement (NFR) (Camacho et al., 2016). This sets security aside, and as an item that can only be considered after all the Functional Requirements (FR) have been treated if time and other resources permit.

Among individuals, organizations, and governments, there has been a profound utilization of technology across the globe within the past decade. This phenomenal usage of technology, powered by state-of-the-art innovation in Research and Development (R & D) has revolutionized the value and perception of data (Harvard Business Review, 2009). Innovations in products, protocols, solutions, applications and artefacts are being implemented and deployed consistently in industries and in the wild. There is equally a perceived over-reliance on technology which is evident in the number of owned and deployed Personal Devices (PD) around the globe (Gartner, 2021). There are currently an estimated 6.2 billion personal devices that are active around the world.

With the persistency in enhancements and advancements of ubiquitous and embedded devices, PDs can now be seen in every home that can afford it (Krini & Laile, 2018). Tablets, Radio Frequency Identification (RFID), sensors, smart phones, health monitors, cameras, smart watches and wearables, Electronic Control Units (ECU), Personal Digital Assistants (PDA), 5G, Global Positioning System (GPS), Internet of Things (IoT), Software Defined Networking (SDN), Satellites, and other pervasive devices are now within fingertips, aiding the resolution of diverse range of issues (Anagha, 2020). Human to Machine interface/interconnectivity has also been standardized and is anticipated to scale up in the near future.

However, these interfaces and connectivity between Machine and Machine (M2M), Humans and Machines (H2M) and Machine to Human (M2H) tend to increase the vulnerability landscape (Kurachi et al., 2018). The potential vulnerabilities and threats that are associated with these developments, protocols and systems could be devasting to organizations if not checked and contained (Bertolino et al., 2018). Thus, technological innovations have been and will continue to be associated with pros and cons from a security and privacy perspective.

Owing to the increased power, functionalities, and capabilities of current and future software systems, theveracity, volume, and velocity of transactional has become very significant and invaluable (Bhukya & Pabboju, 2016; Bialas, 2016; Morrison et al., 2017). Regulatory data such as health related data and Personal Identifiable Information (PII), could be transcribed, analyzed and remapped using several technologies, thus, establishing a wide attack surface for bad actors. It is anticipated that future developments will be equipped with increased capabilities and will also be associated with bigger risks and vulnerabilities. Not to mention that adversaries will also continue to perfect their trade, leveraging data mining and database exploitation mechanisms, memory dumps and buffer overflow vulnerabilities, and threat vectors like side channel attacks, SQL Injection, inappropriate Random Number Generation (RNG), poorly designed exclusive control, and integrity checks, insufficient input validation and output encoding, cryptanalysis, broken authentication and authorization bypass (Kamtuo & Soomlek, 2016). The vulnerability landscape has been amplified by cyber criminals in their quest for digital espionage, identify theft, digital surveillance, cyber warfare. The capabilities of bad actors are further enhanced by the daily generation of exploits, threats and vulnerability research. These reinforces their will and dedication to execute devastating attacks with the intent to deface, destroy, and compromise systems, with considerable impacts that undermines the privacy, safety and security of individuals, organizations and governments.

3. Secure Software Development Lifecyle (SSDLC)

With the great efforts and sophisticated attack patterns from adversaries, it has become evident that that actions from cybercriminals have evolved over the years. The continuous enhancements and innovations in attack techniques and methodologies have significantly aided the execution of attack campaigns. Owing to the improvements in exploit generations, automated footprinting, attack vectors, and hacking techniques, there seems to be numerous ways through which systems can be compromised (Karantzas & Patsakis, 2021; The CIS Critical Security Controls for Effective Cyber Defense, n.d). There has been a perceived cyber warfare between cybersecurity professionals and cybercriminals, and given the growing number of attack vectors, zero days, and research on threats and vulnerabilities, cybersecurity professionals appear to be either at loss or playing catch-up with the adversaries.

A reactionary approach to cybersecurity will always put the bad actors in front, causing the adversary to lead and dictate the pace. Albeit the efforts of cybersecurity professionals to secure enterprise organizations cannot be undermined by this trend, their purposed should be better served with a proactive approach to identify risks, threats and vulnerabilities with enterprise assets prior to production deployment. It is important to implement software with a security mindset. Systems, solutions, software, and all related artefacts should be deployed with the least amount of vulnerability. All efforts should be geared towards the identification, remediation or mitigations of all potential vulnerabilities associated with software, and there could be no better structured and organized way of achieving this feat than to embed security into the software development lifecycle.

Irrespective of the growing concerns of cyberattacks observed at a global level, including but not limited to Log4J, CryptoLocker WannaCry, etc., developers still find it difficult to fully embrace the tenets of secure software development practices within their Development Operations (DevOps) environment (CSIS, 2021; Daley, 2017; Mohan et al., 2018). Within the current software development ecosystem, agile and scrum frameworks have been predominantly used by developers. However, these frameworks are quite limited in values from an application security perspective (Camacho et al., 2016). The prioritization of the business facing components over security to ensure the speedy delivery of products, customer satisfaction and revenue generation has been the order of the day as illustrates in Figure 2. Thus, security is practiced as an afterthought instead of being integrated into the SDLC. In order to contain this gap in security and eliminate/mitigate the associated potential risks, threats and vulnerabilities, as well as to ensure compliance with regulatory requirements, security should be included as one of the functional requirements in any given development framework.

Current development practices should be customized to accommodate security views (DHS, 2018). The SSDLC presented in this paper seeks to aid the identification, remediation, and mitigation of potential vulnerabilities prior to production deployment of assets (Banerjee et al., 2017). The proposed SSDLC engages security from the early stages of development and incorporates it all the way. Figure 3 depicts an enumeration of the different stages of the proposed SSDLC.

4. Requirement

The first phase of SSDLC presents the need for the collation of all the necessary requirements that will aid the actualization of a securely developed software (Choi &

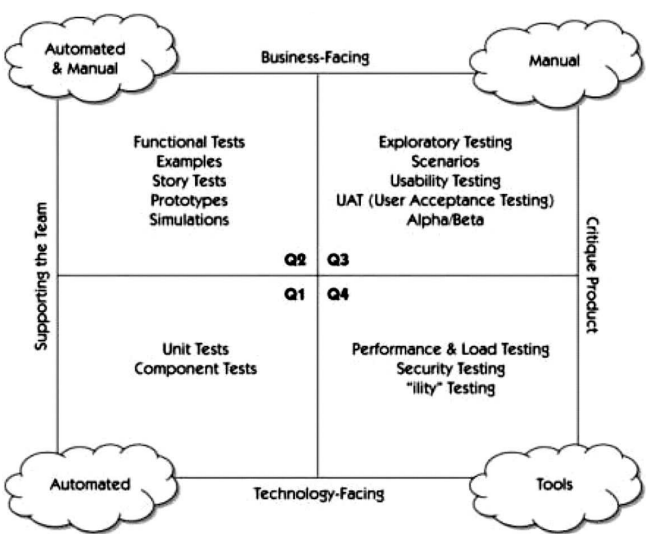

Figure 2. Agile Testing Quadrant (Crispin & Gregory, 2009)

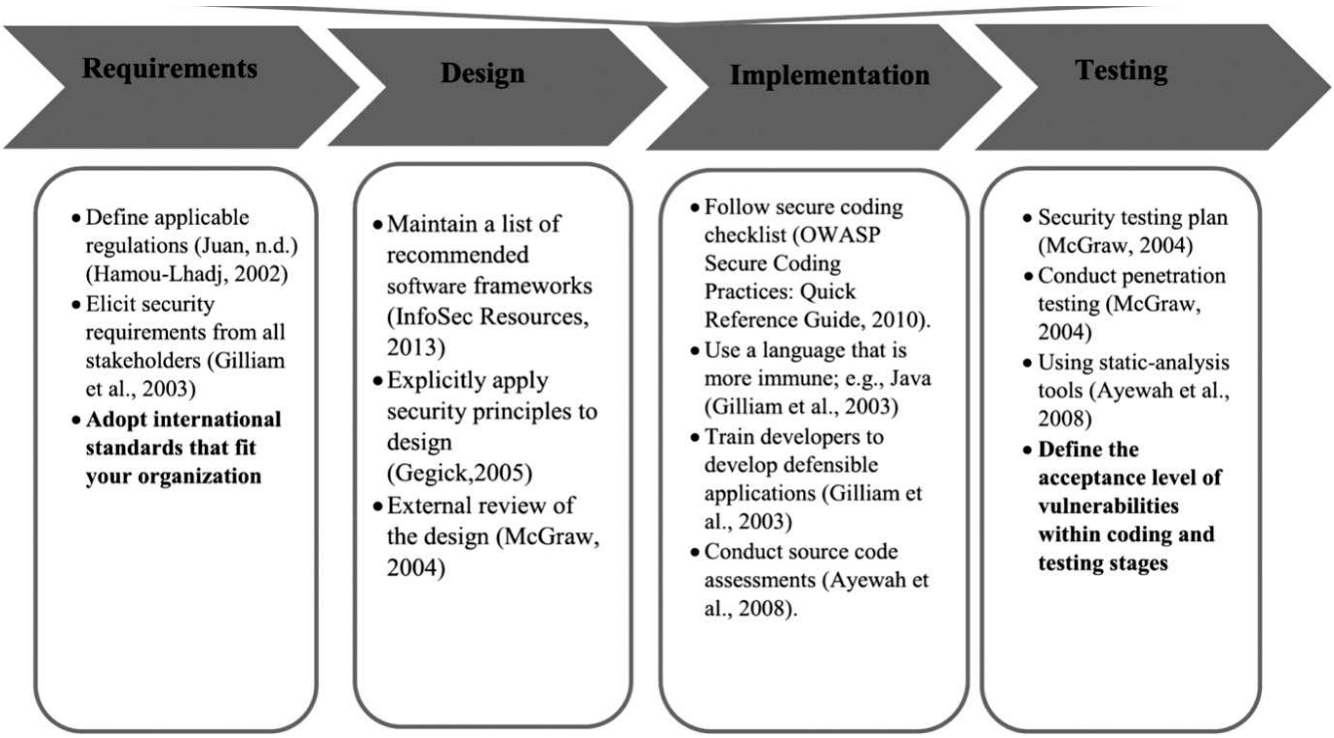

Figure 3. Secure Software Development Model (Karim et al., 2016)

Kim, 2017). Insufficient preparation and inadequate data and input solicitation has been underlying issue with most implementations in enterprise organizations. Several projects have failed due to the inability of project managers to collate and elicit critical information from all the interested parties.

All the stakeholders should be involved during the requirement specification stage of SSDLC. This provides the opportunity to identify some of the critical security issues that could otherwise pose as showstoppers or serve as major obstacles towards the deployment of software solutions to production. The requirement phase will equally serve as an avenue to enumerate the potential structure and architecture of the project (Salini & Kanmani, 2016). This will typically include data type, accessibility (geolocation), and considerations for application end users.

Requirement specification helps to understand the applicable regulations and frameworks to an application (Span et al., 2018). For example, projects pertaining to Payment Card Information (PCI) must be built in conformity with the Payment Card Industry Data Security Standard (PCI-DSS) (Cloudcheck, 2019). This will help to define how the Personal Identification Number (PIN), and the Personal Account Number (PAN) are masked, and the encryption of data at rest and in transit (Petronko et al., 2019). Applications associated with personal health information will be evaluated against HIPAA requirements (HIPAA, 2021). Also, data belonging to European Union (EU) citizens will also attract compliance to regulations such as the General Data Protection Requirement (GDPR) (GDPR, 2022). Other considerable regulations includeNYDFS, SOX, etc., (Groot, 2020). Requirement specification should generate a checklist of all regulatory frameworks that can be evaluated in order to avoid fine, and litigations (Fein et al., 2018).

5. Design

All the applicable secure software frameworks and libraries within an enterprise organization should be enumerated as part of the SSDLC. The maintenance of frameworks and libraries will promote consistency with respect to the security of software in the SSDLC. Such uniformity will promote software and library reuse amongst the developers and help to avoid potential

reinvention of the wheel (Wendzel, 2016). Some of the stored inventories will include industry standard security frameworks and certified code libraries.

The design phase of the SSDLC will permit the execution of Threat Modelling (TM) against the proposed implementation. Leveraging frameworks such as Spoofing identity, Tampering with data, Repudiation threats, Information disclosure, Denial of service and Elevation of privileges (STRIDE), security analysts can initiate the evaluation of all potential vulnerabilities including spoofing, tampering, repudiation, information disclosure, Denial of Service (DoS), and Encryption as they relate to the proposed implementation (Lord et al., 2016; Microsoft, n.d). The design phase will serve to assess the entire architecture of the proposed implementation. Individual standards and policies of organizations should equally be validated against the application during this phase. Security architects should assess the software components against the Minimum Baseline Security Standard (MBSS) established within the enterprise organization. Some of the notable items to be reviewed during this phase should include – authentication and encryption protocols, maintenance, database backup and restoration, logging and archival processes applicable to the proposed implementation (Hu et al., 2017). The aforementioned items should be assessed against the organization's standards and policies. Industry best practices should be observed to maximize the security posture of applications. Considerations for scalability and upgrade, and Legacy system support should also be made during the SSDLC design phase (Karim et al., 2016). It is also important for organizations to consider if external stakeholders can be deployed to review the proposed implementation. This approach helps to usher in fresh cybersecurity perspectives.

6. Implementation

Software development is typically a fun exercise for seasoned developers. An average developer focuses on good quality code generation and struggles to produce the required application functionalities. The software developer's achievement is however short-lived and does not suffice judging by the current level of successful cyber-attacks observe around the globe. Developers would unconsciously introduce vulnerabilities into their products during software programming, this is equally observed during code amendments and remediation efforts (Yoshizawa et al., 2016). The trend constitutes a fundamental issue with software development and warrants a technical and strategic approach for containment.

To mitigate these implementation issues, secure code practices should be deployed as part of the SSDLC. With the variety of secure coding practices available for developers, including the Open Web Application Security Project (OWASP), SANS Secure Coding practices, and secure coding practices from, there is a significant list of standards and frameworks that developers can leverage to secure their products (OWASP, n.d; SANS, 2021; Veracode, n.d). A majority of the vulnerabilities associated with applications are inherent to the structure of the frameworks, programming language, or other application artefacts that used for development. The need to ensure software is programmed with structured and secure programming languages is thus underscored. For example, Java Application Programming Interface (API) is filled with secure codes, and should be leveraged by developers (53 Using the Java EE Security API, 2017). Java provides memory management, with the ability to mitigate memory related attacks like DoS, and buffer overflows.

Adequate developer training is a must for SSDLC. It is pertinent for organizations to mandate regular trainings for developers, with continuous updates on innovations and technological advancements within the coding community. Developers should be provided with API plugins like Synopsis' "Secure Code Assist", and Checkmarx to help identify potential vulnerabilities in their lines of codes (Checkmarx, n.d; Synopsis, 2022). This practice should be embraced by the developers in order for them to generate better secure codes that meet industry standards, regulations and frameworks. This effort will require a strong support from senior leadership.

7. Testing

It is not uncommon for organizations to deployed

applications without undergoing any form of assessments (Daley, 2017). This practice is still observed in several organizations till date, and several of them have been compromised due to this insecure practice. The value of test plans cannot be over emphasized (McGinnis et al., 2015). Test plans should be published and utilized as an integral part software development. Test plans can be generated to capture the processes, domain, protocols, scenario, attack vectors, and logical structures that must be tested (Scarfone et al., 2008). Test plans should be populated for secure code analysis and penetration testing.

Static Application Security Testing (SAST) will facilitate the identification of potential vulnerabilities that are associated with applications at code-time. This is a key step in the SSDLC. During this phase of SSDLC, the security analyst would have complete access to the source code, giving room for a better analysis and evaluation of the software (Theisen et al., 2017). This is a typical white box approach which presents a unique and useful means to uncover findings which might have otherwise been difficult to identify.

However, Penetration Testing (PT) should be performed to identify run-time vulnerabilities once the application has been fully developed. PT will include the utilization of automated and manual efforts. The manual efforts are very significant due to their ability to validate the findings from automated scans and help to identify certain findings that are logical that will be missed by the automated scans (Semenov et al., 2021). PT should be performed on both external and internal applications to account for external attacks and insider attacks, as well as the promotion of Zero trust. Application vendors will normally not permit external entities to perform PT on their solutions given the multi-tenancy structure of certain applications. The scenario should prompt for the demand for an independent 3rd party PT client facing report from the vendor.

Vulnerability classification, remediation Service Level Agreement (SLA), and the organization's software security policies should be used to address all identified vulnerabilities. The organization's software security standard can be leveraged to determine if an application has passed a security assessment or not (Choi & Kim, 2017). This will equally help to determine if an application would be authorized for onward deployment to production. Using SSDLC, applications that do not "pass" the organization's software security policy would not be deployed to production. Assuming an enterprise policy states that all "Medium" or higher severity rated findings should be remediated before the associated applications can be deployed to production, any applications that defaults to this rule will be restricted from getting deployed.

Consequently, the enterprise vulnerability remediation SLA can be used to define the number of days a vulnerability is permitted to remain open without remediation. The National Institute of Standards and Technology (NIST) Common Vulnerability Scoring System (CVSS), an industry standard vulnerability classification metrics can be used for vulnerability rating/classification (NIST, 2016). CVSS leverages several factors, including the value of the associated asset, ease of exploitation, required technical skill, and the impact on asset/ organization's Confidentiality, Integrity, and Availability (CIA) while calculating the vulnerability severity rating.

Conclusion

Secure software development should not be optional or a non-functional requirement. Enterprise organizations should integrate SSDLC into their software development practices. Secure software development should be enshrined in the organizational culture. To succeed, SSDLC must be policy driven. There should be publication and enforcement of policy standard documentation, fully supported by senior leadership. Training, security tips and awareness should be mandated for developers and inculcated into the annual performance ratings. All the secure software libraries and frameworks should be living documents that has to be frequently reviewed and updated accordingly.

Enterprise applications should be reviewed against all the applicable regulations and frameworks during the design phase, and security testing should be iteratively

performed in all the stages of SSDLC. The software release stage must be extended to include the creation of incident response plans, with provisions for potential rollover, disaster recovery and business continuity. All efforts should be made to reduce the Recovery Point Objective (RPO), and the Recovery Time Objective (RTO). Different Security Architectures (SA) including Singapore Exchange (SGX), International Organization for Standardization (ISO) 25010 can be leveraged towards the actualization of secure software development. The implementation of this guideline in enterprise organizations will not only create the required cybersecurity awareness, but also help to identify, mitigate, and remediate threats and vulnerabilities. It will serve as precursor for the determination of enterprise security posture and equip organizations with the capacity to fend off cybercriminals and other potential cyber-adversaries.

References

[1]. 53 Using the Java EE Security API. (2017). *Retrieved from* https://javaee.github.io/tutorial/security-api.html

[2]. Al-Amin, S., Ajmeri, N., Du, H., Berglund, E. Z., & Singh, M. P. (2018). Toward effective adoption of secure software development practices. *Simulation Modelling Practice and Theory*, 85, 33-46. https://doi.org/10.1016/j.simpat.2018.03.006

[3]. Allahar, H. (2019). Innovation management and value chain design: Case of a small professional services firm. *International Journal of Innovation*, 7(2), 192-209. https://doi.org/10.5585/iji.v7i2.380

[4]. Banerjee, C., Banerjee, A., & Sharma, S. K. (2017). Estimating influence of threat using Misuse Case Oriented Quality Requirements (MCOQR) Metrics: Security Requirements Engineering. *Journal of Information Assurance & Security*, 12(3), 104–113.

[5]. Bhukya, S., & Pabboju, S. (2016, March). Software architecture techniques and emergence of problem domain in E-Governance. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, (pp. 1097-1109), IEEE. https://doi.org/10.1109/ICEEOT.2016.7754856

[6]. Bialas, A. (2016). Computer-aided sensor development focused on security issues. *Sensors*, 16(6), 759. https://doi.org/10.3390/s16060759

[7]. Camacho, C. R., Marczak, S., & Cruzes, D. S. (2016, August). Agile team members perceptions on non-functional testing: influencing factors from an empirical study. In *2016 11th international Conference on Availability, Reliability and Security (ARES)*, (pp. 582-589), IEEE. https://doi.org/10.1109/ARES.2016.98

[8]. Checkmarx. (n.d.). *The World Runs on Code.* We secure it. Retrieved from https://checkmarx.com/

[9]. Choi, K. H., & Kim, K. J. (2017). A study on acceptance procedure improvement of web application by outsourcing for mobile service. *Wireless Personal Communications*, 94(1), 5-16. https://doi.org/10.1007/s11277-015-3153-0

[10]. Cloudcheck. (2019). *How the Payment Card Industry Data Security Standard (PCI DSS) works*. Retrieved from https://cloudcheckr.com/cloud-compliance/how-the-payment-card-industry-data-security-standard-pci-dss-works/?gclid=EAIaIQobChMIor7ogay39QIVkcmUCR2IBwYfEAAYASAAEgJ63_D_BwE

[11]. CSIS. (2022). *A Shared Responsibility: Public-Private Cooperation for Cybersecurity*. Retrieved from https://www.csis.org/analysis/shared-responsibility-public-private-cooperation-cybersecurity

[12]. Cybersecurity & Infrastructure Security Agency. (2020). *Federal Information Security Modernization Act*. Retrieved from https://www.cisa.gov/federal-information-security-modernization-act

[13]. Daley, J. (2017). Insecure software is eating the world: Promoting cybersecurity in an age of ubiquitous software-embedded systems. *Stanford Technology Law Review*, 19(3), 533–546.

[14]. Dayanandan, U., & Kalimuthu, V. (2018). Software architectural quality assessment model for security analysis using Fuzzy Analytical Hierarchy Process (FAHP) method. *3D Research*, 9(3), 1-14. https://doi.org/10.1007/s13319-018-0183-x

[15]. DHS. (2018, May 15). *U.S. Department of Homeland Security: Cybersecurity Strategy*. Department of

Homeland Security.

[16]. Fein, A., Skeath, C., & Brewer, L. (2018). Key information security pointers from the FTC's stick with security guidance. *Intellectual Property & Technology Law Journal*, 30(3), 19-22.

[17]. Fernandes, A. M., Pai, A., & Colaco, L. M. M. (2018, March). Secure SDLC for IoT based health monitor. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)* (pp. 1236-1241). IEEE. https://doi.org/10.1109/ICECA.2018.8474668

[18]. Fernández-García, A. J., Iribarne, L., Corral, A., Criado, J., & Wang, J. Z. (2018). A flexible data acquisition system for storing the interactions on mashup user interfaces. *Computer Standards & Interfaces*, 59, 10-34. https://doi.org/10.1016/j.csi.2018.02.002

[19]. Gartner. (2021). *Gartner Forecasts Global Devices Installed Base to Reach 6.2 Billion Units in 2021: Remote and Hybrid Work is Increasing the Number of Devices Per Person*. Retrieved from https://www.gartner.com/en/newsroom/press-releases/2021-04-01-gartner-forecasts-global-devices-installed-base-to-reach-6-2-billion-units-in-2021

[20]. GDPR. (2022). *What are the GDPR fines?* Retrieved from https://gdpr.eu/fines/

[21]. Groot, J. D. (2020). *What Is The NYDFS Cybersecurity Regulation? A Cybersecurity Compliance Requirement for Financial Institutions*. Retrieved from https://digitalguardian.com/blog/what-nydfs-cybersecurity-regulation-new-cybersecurity-compliance-requirement-financial

[22]. Harvard Business Review. (2009). *Creating a Culture of Innovation*. Retrieved from https://ncuone.ncu.edu/d2l/le/content/91264/viewContent/569088/View?ou=91264

[23]. HIPAA. (2021). *What are the Penalties for HIPAA Violations?*. Retrieved from https://www.hipaajournal.com/what-are-the-penalties-for-hipaa-violations-7096/

[24]. Hu, V. C., Kuhn, R., & Yaga, D. (2017). Verification and test methods for access control policies/models. *NIST Special Publication*, 800, 192. https://doi.org/10.6028/NIST.SP.800-192

[25]. IT Governance. (n.d.). Cybersecurity Governance and Frameworks. Retrieved from https://www.itgovernanceusa.com/cybersecurity-standards

[26]. Karantzas, G., & Patsakis, C. (2021). An empirical assessment of endpoint detection and response systems against advanced persistent threats attack vectors. *Journal of Cybersecurity and Privacy*, 1(3), 387-421. https://doi.org/10.3390/jcp1030021

[27]. Karim, N. S. A., Albuolayan, A., Saba, T., & Rehman, A. (2016). The practice of secure software development in SDLC: an investigation through existing model and a case study. *Security and Communication Networks*, 9(18), 5333-5345. https://doi.org/10.1002/sec.1700

[28]. Karim, N. S. A., Albuolayan, A., Saba, T., & Rehman, A. (2016). The practice of secure software development in SDLC: an investigation through existing model and a case study. *Security and Communication Networks*, 9(18), 5333-5345. https://doi.org/10.1109/ICMCS.2018.8525494.

[29]. Kriebel, F., Rehman, S., Hanif, M. A., Khalid, F., & Shafique, M. (2018, July). Robustness for smart cyber physical systems and internet-of-things: From adaptive robustness methods to reliability and security for machine learning. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* (pp. 581-586). IEEE. https://doi.org/10.1109/ISVLSI.2018.00111

[30]. Krini, O., & Laile, E. (2018). Unambiguous and Reliable Positioning in the vehicle in terms of Functional Safety and Cyber Security. In MATEC Web of Conferences (Vol. 210, p. 03013). EDP Sciences. https://doi.org/10.1051/matecconf/201821003013

[31]. Kuhn, D. R., Kacker, R. N., & Lei, Y. (2010). Practical combinatorial testing. *NIST Special Publication*, 800(142), 142.

[32]. Lord, S., Helfgott, A., & Vervoort, J. M. (2016). Choosing diverse sets of plausible scenarios in multidimensional exploratory futures techniques. *Futures*, 77, 11-27. https://doi.org/10.1016/j.futures.2015.12.003

[33]. McGinnis, C., Yaga, D., Podio, F. (2015). *Conformance Testing Methodology Framework for*

ANSI/NIST-ITL 1-2011 Update: 2013, Data Format for the Interchange of Fingerprint, Facial & Other Biometric Information. NIST Special Publication, 500, 304.

[34]. Microfocus. (n.d.). *Retrieved from* https://www.microfocus.com/en-us/cyberres/application-security

[35]. Microsoft. (n.d.). *Microsoft Security Development Lifecycle (SDL)*. Retrieved from https://www.microsoft.com/en-us/securityengineering/sdl/

[36]. Mohan, V., ben Othmane, L., & Kres, A. (2018, September). BP: Security concerns and best practices for automation of software deployment processes: An industrial case study. In *2018 IEEE Cybersecurity Development (SecDev)* (pp. 21-28). IEEE. https://doi.org/10.1109/SecDev.2018.00011

[37]. Morrison, P., Smith, B. H., & Williams, L. (2017, May). Measuring security practice use: A case study at IBM. In *2017 IEEE/ACM 5th International Workshop on Conducting Empirical Studies in Industry (CESI)* (pp. 16-22). IEEE. 10.1109/CESI.2017.4

[38]. Musa, S. B., Md Norwawi, N., Selamat, M. H., & Al-Alwani, A. (2015). Systematic review of web application security development model. *Artificial Intelligence Review*, 43(2), 259-276. http://doi.org/10.1007/s10462-012-9375-6

[39]. NIST. (2012). *NIST Special Publication 800-165*. Computer Security Division. Retrieved from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-165.pdf

[40]. NIST. (2016). *Special Publication 800-166, Derived PIV Application and Data Model Test Guidelines*. Retrieved from https://csrc.nist.gov/News/2016/Special-Publication-800-166

[41]. OWASP. (n.d). *Project Spotlight: Top 10*. Retrieved from https://owasp.org/projects/spotlight/#:~:text=The%20OWASP%20Top%2010%20is,organisations%20and%20is%20then%20analysed

[42]. OWASP. (n.d.). *OWASP Top Ten*. Retrieved from https://owasp.org/www-project-top-ten/

[43]. Pathak, N. (2018). UML 2.0 based round trip engineering framework for the development of SPF based secure application. In *Journal of Engineering Science and Technology* (Vol. 13, No. 9, pp. 2734-2749). School of Engineering, Taylor's University, Malaysia.

[44]. Petrenko, K., Mashatan, A., & Shirazi, F. (2019). Assessing the quantum-resistant cryptographic agility of routing and switching IT network infrastructure in a large-size financial organization. *Journal of Information Security and Applications*, 46, 151-163. https://doi.org/10.1016/j.jisa.2019.03.007

[45]. Ren, Y., Liu, L., Zhang, Q., Wu, Q., Guan, J., Kong, J., & Shao, L. (2016). Shared-memory optimizations for inter-virtual-machine communication. *ACM Computing Surveys (CSUR)*, 48(4), 1-42. https://doi.org/10.1145/2847562

[46]. Salini, P., & Kanmani, S. (2016). Effectiveness and performance analysis of model-oriented security requirements engineering to elicit security requirements: a systematic solution for developing secure software systems. *International Journal of Information Security*, 15(3), 319-334. https://doi.org/10.1007/s10207-015-0305-x

[47]. SANS. (2021). *Web Application Security Awareness*. Retrieved from https://www.sans.org/security-awareness-training/products/specialized-training/developer/

[48]. Scarfone, K. A., Souppaya, M. P., Cody, A., & Orebaugh, A. D. (2008). *Sp 800-115. Technical Guide to Information Security Testing and Assessment*. Retrieved from https://csrc.nist.gov/publications/detail/sp/800-115/final

[49]. Schoeni, D. E. (2015). Long on rhetoric, short on results: Agile methods and cyber acquisitions in the Department of Defense. *Santa Clara Computer & High Tech. LJ*, 31, 385.

[50]. Semenov, S. S., Weilin, C., Liqiang, Z., & Bulba, S. S. (2021). Automated penetration testing method using deep machine learning technology. *Advanced Information Systems*, 5(3), 119-127. https://doi.org/10.20998/2522-9052.2021.3.16

[51]. Silva, L. V., Barbosa, P., Marinho, R., & Brito, A. (2018). Security and privacy aware data aggregation on cloud computing. *Journal of Internet Services and Applications*,

9(1), 1-13. https://doi.org/10.1186/s13174-018-0078-3

[52]. Span, M. T., Mailloux, L. O., Grimaila, M. R., & Young, W. B. (2018, June). A Systems Security Approach for Requirements Analysis of Complex Cyber-Physical Systems. In *2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)* (pp. 1-8). IEEE.

[53]. Synopsis. (2022). *Secure Code Assist Overview*. Retrieved from https://community.synopsys.com/s/article/SecureAssist-Overview

[54]. The CIS Critical Security Controls for Effective Cyber Defense. (n.d.). In *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/The_CIS_Critical_Security_Controls_for_Effective_Cyber_Defense

[55]. The United States Department of Justice. (2021). *Privacy Act of 1974*. Retrieved from https://www.justice.gov/opcl/privacy-act-1974#:~:text=The%20Privacy%20Act%20of%201974,of%20records%20by%20federal%20agencies

[56]. Theisen, C., Herzig, K., Murphy, B., & Williams, L. (2017, May). Risk-based attack surface approximation: how much data is enough?. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)* (pp. 273-282). IEEE. https://doi.org/10.1109/ICSE-SEIP.2017.9

[57]. Van Rossem, S., Tavernier, W., Colle, D., Pickavet, M., & Demeester, P. (2018). Introducing development features for virtualized network services. *IEEE Communications Magazine*, 56(8), 184-192. https://doi.org/10.1109/MCOM.2018.1600104

[58]. Veracode. (n.d.). *Software Code Security & Secure Code Analysis Software Code Security Protects the Enterprise*. Retrieved from https://www.veracode.com/security/code-security

[59]. Vidas, T., Larsen, P., Okhravi, H., & Sadeghi, A. R. (2018). Changing the game of software security. *IEEE Security & Privacy*, 16(2), 10-11. https://doi.org/10.1109/MSP.2018.1870863

[60]. Wang, W., Zhang, X., Hao, Q., Zhang, Z., Xu, B., Dong, H., & Wang, X. (2019). Hardware-enhanced protection for the runtime data security in embedded systems. *Electronics*, 8(1), 52. https://doi.org/10.3390/electronics8010052

[61]. Wendzel, S. (2016). How to increase the security of smart buildings?. *Communications of the ACM*, 59(5), 47-49. https://doi.org/10.1145/2828636

[62]. Williams, M. A., Dey, S., Barranco, R. C., Naim, S. M., Hossain, M. S., & Akbar, M. (2018, December). Analyzing evolving trends of vulnerabilities in national vulnerability database. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 3011-3020). IEEE. https://doi.org/10.1109/BigData.2018.8622299

[63]. Yoshizawa, M., Washizaki, H., Fukazawa, Y., Okubo, T., Kaiya, H., & Yoshioka, N. (2016). Implementation support of security design patterns using test templates. *Information*, 7(2), 34. https://doi.org/10.3390/info7020034

ABOUT THE AUTHOR

Henry O. Nwaete, Northcentral University, San Diego, California.