International Journal of Advances in Engineering and Management (IJAEM)
Volume 6, Issue 08 Aug. 2024, pp: 635-640 www.ijaem.net ISSN: 2395-5252

A Cluster Model of Collaborative Vocations for Managing Software Security Vulnerabilities at the Source Code Level

Julius N. Obidinnu, S. O. O. Duke

Department of Computer Science, University of Cross River State, Calabar, Nigeria.
Department of Computer Science, University of Cross River State, Calabar, Nigeria.

Date of Submission: 25-08-2024 Date of Acceptance: 05-09-2024

ABSTRACT: Software security mishaps have disastrous effects on businesses, including adverse publicity, which always triggers a sharp drop in the valuation of associated software development companies. These mishaps occur due largely to the exploitation of security vulnerabilities in source code. This paper posits that the path to creating secure software begins by rigorously testing the source code for vulnerabilities. The paper further posits that such source code testing requires the expertise of a range of skilled human software security experts that should span the entire software development life cycle. However, there is a lack of software security human experts in the software engineering field in relation to the abundant number of software developers specialized in various coding technologies. Consequently, the typical software development projects do not necessarily have available software security human experts to complement the conventional developer's lack of knowledge (and interest) in the domain. Furthermore, there is minimal attention towards the training of these set of skilled experts. This paper therefore presents a vulnerability patch/alarm cycle model, which clarifies that deploying software without managing source code security vulnerabilities remediation exposes it to a deluge of exploitative attacks, requiring patches, which oftentimes, introduces additional security vulnerabilities, and the cycle continues. Then, the cluster model of collaborative software security vocations is also presented to depict a methodology for linking the distinct range of skilled human software security experts and their functions in managing source code security vulnerabilities across the phases of an application's lifecycle. The model provides a vision and a process to change the focus, and enhance the development of more secure software systems.

KEYWORDS: Internet, Source Code, Vulnerability, Software Security, Career Cluster, Security Architect, Security Auditor

I. INTRODUCTION

Notwithstanding the efforts at stemming the tide, stakeholders are in agreement that software security threats remain an ongoing conundrum[1]. In fact, the highest number of information assets exposed due to security breaches in the United States of America occurred in 2021, which was the last year in the survey [2]. Associated with these breaches are the disastrous effects on businesses, including adverse publicity, which always triggers a sharp drop in the valuation of associated software development companies. It is established that the root cause of majority of the software security breaches is the exploitation of vulnerability in the source code of deployed software [3]. Meanwhile, the cheapest and most effective approach to resolving software security issues is to identify the defects early in the software development life cycle [2]. This paper posits that, vulnerability in source code go unnoticed, and gets deployed, because software security is under-theorized, hence, programmers under-emphasize it in the software development process. Besides, minimal attention is given to the training of the set of skilled experts in this domain. Consequently, there is a dearth of personnel who are sufficiently skilled in resolving source code security vulnerabilities. Looking to the future therefore, there is a need to draw attention to the training of human experts for employment in multiple areas of source code security-oriented testing skills that meets a wide variety of requirements across the entire software development life cycle (SDLC). This will lead to the availability of security conscious and skilled personnel to be coopted

International Journal of Advances in Engineering and Management (IJAEM)
Volume 6, Issue 08 Aug. 2024, pp: 635-640 www.ijaem.net ISSN: 2395-5252

across the phases of the SDLC to undertake the security-oriented tasks using standardized methodologies.

This paper presents two models. The first of the models is a vulnerability patch/alarm cycle model, which captures a significant aspect of the subsisting source code security vulnerabilities management process. This model makes it clear that deploying software without managing source code security vulnerabilities exposes it to a deluge of exploitative attacks, requiring patches, which oftentimes, introduces additional security vulnerabilities, and the cycle continues. However, applying security-oriented patches over and over again, as though systems administrators had nothing else to do, is never going to give us a secure internet. The second is a cluster model of collaborative software security vocations, which introduces a methodology for interfacingthe distinct range of skilled human software security experts and their functions necessary for managing source code security vulnerabilities across the phases of applications' lifecycle.In practice, the model seeks to achieve more secure systems, by making software security an integral part of every phase of an application's overall lifecycle. To achieve this, the model identifies the actors and functions that will promote interoperability through baseline and enhanced capabilities across the SDLC.

The objective of this paper is to evoke awareness of the need to expand the focus, and the process to guide the change towards modeling, implementation, and deployment of software with security in mind across the SDLC. Career clusters are groups of occupations and industries that have in common a set of foundational knowledge and skills [4]. Clustering seeks stability and confidence by developing a set of professionals that share a wide range of skills and provide backup expertise that will ensure business continuity.

II. METHOD

The paper conducts a systematic literature review, which exposes the need to use the instrumentality of creating a software development security career cluster, to draw attention to the software security work domain, towards increasing the number of software security experts in relation to deploying more secure software. The paper then presents a vulnerability patch/alarm cycle model, which captures a significant aspect of the subsisting source code security vulnerabilities management process and the weaknesses inherent therein. Next, the paper presents a cluster model of collaborative software security vocations, which introduces a methodology that links the actors and functions necessary for managing source code security vulnerabilities across the phases of applications' lifecycle. Finally, conclusions are drawn from the presentations.

III. LITERATURE REVIEW

Hoglund and McGraw in [5] posit that the path to creating a secure application begins by rigorously testing source code for all vulnerabilities and ensuring that use of the application does not allow for the compromise of data privacy and integrity. A good starting point would be tools that can be applied directly to the source code to solve or warn about security vulnerabilities. However, the use of security testing tools by unskilled software security human experts is defective, since the scope (knowledge base) of each tool is as good/bad as the limited knowledge of its developers [6]. Besides, the tools provide false or true positive/negative alerts to the testers, which require skilled software security human experts to discern their veracity before accepting or rejecting them. Meanwhile, there are diverse areas to watch out for in software security. The testing tools can therefore only assist in tracking security issues but cannot replace the human expert, who should identify and execute the appropriate solution paths. It is therefore, inappropriate to entrust all the approaches to resolving software security issues into the hands of software security unskilled programmers that may not effectively discern between true/false positive/negative alerts from the automated tools.

Modeling, implementation, and deployment of software with security in mind requires the early involvement by business users, project managers, applications developers, as well as information security practitioners across the SDLC [7]. Early involvement of a variety of skilled human software security experts helps to catch threats across the SDLC [8]. This requires that a software security modeling expert should be coopted at the software specification phase, to proactively contribute software security architectural blueprint [9]. This enables potential security issues to be analyzed early with remediation prospects, preventing a much costlier fix down the line. The blueprint further provides the dimensions for validating the extent to which the end-product can be proven to be secured [10].

SDLC processes have been around for years, but security considerations have not always been incorporated into them [7]. Rather, security

International Journal of Advances in Engineering and Management (IJAEM)
Volume 6, Issue 08 Aug. 2024, pp: 635-640 www.ijaem.net ISSN: 2395-5252

testing tools are usually deployed towards the end of the projects, possibly by one person, to determine the extent to which a potential end product is secured. This situation prevails because there is a lack of security experts in the software engineering field, while there exists an abundance of software developers, who are experts in various coding technologies [11]. Consequently, there is a knowledge gap between security experts and software developers, since only few software developers follow the secure coding best practices, even though most of the software security vulnerabilities are caused by well-known mistakes [11]. For these reasons, the typical software development projects do not necessarily have available security expertise to complement the ordinary developer's lack of knowledge (and interest) in the area. There is need therefore, to take steps towards increasing the number of software security experts in relation to software developers.

Career clusters contain occupations in the same field of work that require similar skills. Students, parents, and educators can use career clusters to help focus education plans towards obtaining the necessary knowledge, competencies, and training for success in a particular career pathway [12]. They provide a good way to start thinking about careers. They provide an organized way to look at occupations in a domain and create a plan to move from school into a good-paying job [13]. However, software security career cluster has continued to be underemphasized in the scheme of computing and information technology. For instance, the Information & Technology domain of work, which is one of the sixteen career clusters recognized by the U.S. Department of Education did not include software security as a distinct career field, but security was mentioned sparingly in terms of Networks and Databases[14]. This trend is the same in other bibliographies [15]. There is need therefore, to use the instrumentality of creating a software development security career cluster, to draw attention to this work domain, towards increasing the number of software security experts in relation todeploying more secure software.

IV. THE VULNERABILITY PATCH/ALARM CYCLE

The vulnerability patch/alarm cycle model, which captures a significant aspect of the subsisting source code security vulnerabilities management process is presented in Figure 1. Figure 1 depicts the relationships linking the actors and their roles in the patch/alarm cycle of source code security vulnerability management.

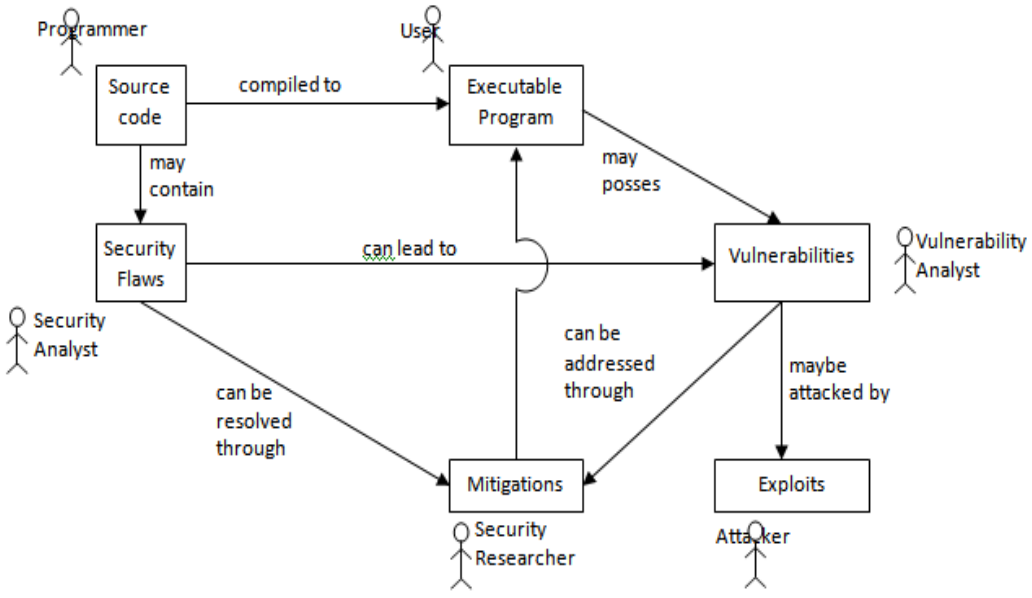

Figure 1: Actors and their Functions in the Patch/Alarm Cycle of Source Code Security Vulnerability Management

International Journal of Advances in Engineering and Management (IJAEM)
Volume 6, Issue 08 Aug. 2024, pp: 635-640 www.ijaem.net ISSN: 2395-5252

The essential features in Figure 1 are listed below:

1. Software is deployed with firewalls, but without managing source code vulnerabilities;
2. An inquisitive person uncovers and discloses a vulnerability in the deployed software;
3. Unethical hackers quickly analyze the vulnerability behaviorsand use it to launch attacks against the software;
4. Several businesses that rely on the attacked software experience interruptions;
5. The software developers at the organizations that produced the product (and the vulnerability) are deluged with phone calls and mails from the public, wanting to find out what is going on;
6. The software developers in affected organizations analyze the vulnerability, develop a fix, test the fix in a controlled environment, and release the fix to the community of users who rely on the software;
7. When a patch is published, some of the users will obtain, test, and apply it. But inevitably, many of the affected systems will never be patched during the lifetime of the vulnerability, or will receive the patch as part of major version upgrade;
8. Weeks or months go by, and a piece of malicious software is released to the internet. This software automates the exploitation of the vulnerability on unpatched systems spreading without control across a large number of sites. Although many sites have patched their systems, many have not, and the resulting panic once again causes a great deal of business interruption across the internet.
9. Using this practice, many of the patches themselves introduce additional security vulnerabilities.

Applying security-oriented patches over and over, as though systems administrators had nothing else to do, is never going to give us a secure internet-based infrastructure.

V. CLUSTER MODEL OF COLLABORATIVE SOFTWARE SECURITY VOCATIONS

We modify the model in Figure 1, with a view to including the actors and functions necessary for managing source code security vulnerabilities across the phases of an application's lifecycle. We consider that the SDLC, from inception to deployment, comprises of the initial architecture, detailed design, implementation (coding), and deployment. This consideration informs the cluster model of collaborative software security vocations, as presented in Figure 2. The peculiarities of a cluster model have been presented in earlier sections of this paper.

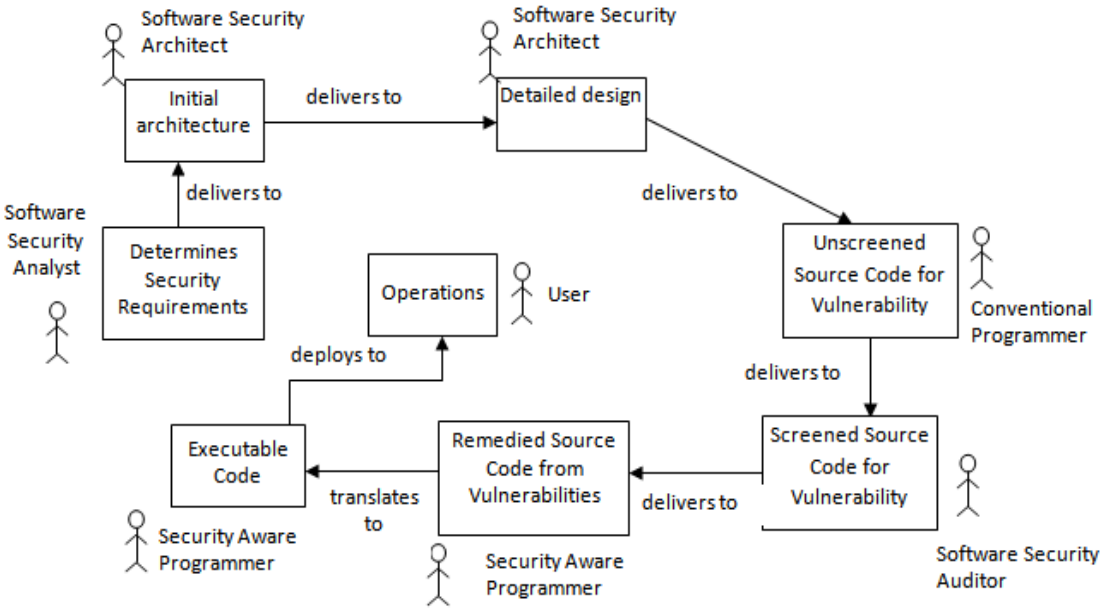

Figure 2: Cluster Model of Interfacing Software Security Vocations Showing Actors and their Functions in Managing Security Vulnerabilities across Application's Lifecycle

International Journal of Advances in Engineering and Management (IJAEM)
Volume 6, Issue 08 Aug. 2024, pp: 635-640 www.ijaem.net ISSN: 2395-5252

The cluster model in Figure 2 comprises of a set of five career specializations, including: (i) Software Security Analyst/Researcher, (ii) Software Security Architect, (iii) Conventional Programmer, (iv) Security-Aware Programmer, and (v) Software Security Auditor.

The model begins with and requires that a software security analyst/researcher, who also performs the functions of researching into hackers' attack methods and their mitigations, identifies the software security requirements of a potential system. The software security requirements of a potential system must be designed to fit into the overall design of a potential software, similar to how electrical and mechanical designs are made to fit into the design of a building plan. The security requirements are then translated into an initial high level software security architecture by a software security architect. This function of the software security architect may also be performed by the software security analyst. In other words, one actor may be skilled in performing the functions of more than one role; hence, may be authorized to conduct the functions in those roles. However, an actor should not be overburdened with too many functions, which may derail the benefits accruable from the division of labour and specialization of functions. The software security architect further translates the initial software security architecture into lower-level design details of the system, including entity and logical security requirements for the database, business processes, input/outputs, as well as the data flows.

Next, the lower-level design details are then delivered to the conventional programmer, who translates them into source code using coding technologies. The source code produced at this stage requires to be screened for vulnerabilities; hence, it is delivered to a software security auditor, who applies personal experience and relevant tools to ethically hack (attempt to break into) the software as a way of evaluating the intruder threat. This scheme is similar to having independent (eternal) auditors come into an organization to verify the book-keeping records; hence, the software security auditor applies the same tools and techniques of the malicious attackers, but will neither damage the target system nor steal information. Considering the nature of the work, a typical software security auditor should be very strong in security-oriented programming, databases, and computer networking skills and sufficiently experienced in these businesses. These qualifications and tools are then utilized to determine: (i) what an intruder can see on the target system, (ii) what the intruder can do with the information seen, (iii) whether anyone at the target system can notice an intruder's attempts or successes, (iv) what information assets that are adequately protected, and (v) what the assets are being protected against. After screening the software, the software security auditor provides a report of the vulnerabilities found and instructions on how to remedy them, and then delivers it to the security aware programmer, who remedies the vulnerabilities as reportedly found in the source code. Finally, security aware programmer translates the remedied source code into its equivalent executable code, and deploys it for use.

VI. CONCLUSION

Removing all software security threats and vulnerabilities may not be practicable, but through sustained efforts they can be brought to their safest limits. The notion of treating software security as prophylaxis, whereby it is considered necessary only at the point preceding deployment of an application or the installation of a firewall that guards its environment should be discarded. Deploying software without managing source code security vulnerabilities exposes it to a deluge of exploitative attacks, requiring patches, which oftentimes, introduces additional security vulnerabilities, and the cycle continues. Applying security-oriented patches over and over again, as though systems administrators had nothing else to do, is never going to give us a secure internet.

To achieve more secure systems, security considerations need to be an integral part of every phase of an application's lifecycle rather than being only an add-on feature of software. Making software security an integral part of an application's overall lifecycle helps to detect exploitable vulnerabilities early on in the process. Minimizing the presence of these vulnerabilities in software reduces the risks of their associated adverse effects, and improves the system's ability to react promptly in detecting malicious signals, deterring malicious exploitation, and protecting the information assets .Making a potential software to be security-aware at each of the SDLC phases should be a collaborative process, whereby human software security experts at each phase have the opportunity to develop and contribute a shared understanding of security requirements and options. Such increased awareness strengthens security, as it makes more people compatible with user needs.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 6, Issue 08 Aug. 2024, pp: 635-640 www.ijaem.net ISSN: 2395-5252

REFERENCES

[1]. D. E. Bambauer, 2021,"Conundrum. Minnesota Law Review": 394. Retrieved March 18th, 2021 from https://scholarship.law.umn.edu/mlr/394

[2]. G.Huang, 2023, "Designing Security into Software Systems using Threat Modeling", Advanced Information Security Sandia National Laboratories.

[3]. J. N. Obidinnu andO.S. Duke, 2013, "Deploying Security-Aware Software Systems Using Source Code Vulnerabilities Analysis", IEEE: African Journal of Computing & ICT, Vol. 6. No. 5, December 2013, pp. 171-180.

[4]. N. A. Jankowski,C. L. Kirby, D. D. Bragg, J. L. Taylor, andK. M. Oertle,2009, "Illinois' career cluster model", Urbana-Champaign, IL: OCCRL, University of Illinois at Urbana-Champaign.

[5]. G. Hoglund andG. McGraw, 2004, "Exploiting Software: How to Break Code", Boston: Addison-Wesley Professional.

[6]. J. S. Valacich andJ. F. George, 2020, "Modern Systems Analysis and Design (Eight Edition)", Boston: Pearson Education, Inc.

[7]. GCN: Government Advisory Council Executive Writers Bureau,2024, "How a process model can help bring security into software development", Retrieved June 25th, 2024 from: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwji38GZkIzyAhUIxIUKHb4fAbk4FBAWegQIChAD&url=https%3A%2F%2Fgcn.com%2Farticles%2F2010%2F03%2F03%2Fics2-process-model-for-software-security.aspx&usg=AOvVaw13bxvmbJyeX3ZxlhY8uJmJ

[8]. A. Shostack,2014, "Threat Modeling: Designing for Security", Indianapolis: John Wiley & Sons.

[9]. N. Shevchenko, T. A. Chick, P. O'Riordan, T. P. Scanlon, andC. Woody, 2021, "Threat Modeling: A Summary of Available Methods", Software Engineering Institute, Carnegie Mellon University.

[10]. W. J. Rapaport, 2023, "Philosophy of Computer Science", DRAFT Version of 26 January 2023.

[11]. D. G. Spampinato, E. Hagen, E. T. Baadshaug, K. Krister, andK. S. Velle, 2021, "SeaMonster: Providing tool support for security modeling". NISK-2021 conference.

[12]. R. Pico-Saltos, P. Carrión-Mero, N. MontalvánBurbano, J. Garzás, &A. Redchuk,2024, "Research Trends in Career Success: A Bibliometric Review", Sustainability 2024, 13, 4625.

[13]. Career Exploration, "Career Clusters", Retrieved July 15th, 2024 from http://www.breitlinks.com/careers/career_clusters.htm

[14]. U.S. Department of Education,"Career Cluster-info", Retrieved July 15th, 2024 from http://www.breitlinks.com/careers/career_pdfs/CareerCluster-info.pdf

[15]. O*NET OnLine, "Browse by Career Cluster", Retrieved July 12th, 2024 from https://www.onetonline.org/find/career?c=11&g=Go