Secure Software Development Techniques and Challenges in their Practical Application

1st Ee Sun Jun
School of Computer Science & Engineering
Taylor's University
Selangor, Malaysia,
sunjun.ee@gmail.com

2nd Tong Yi Hong
School of Computer Science & Engineering
Taylor's University
Selangor, Malaysia,
yihong16032245@gmail.com

3rd Ahmed Ifrah Ibrahim
School of Computer Science & Engineering
Taylor's University
Selangor, Malaysia,
ahmed.ifrah.ibrahim@gmail.com

4th Fatima-tuz-Zahra
School of Computer Science & Engineering
Taylor's University
Selangor, Malaysia,
fatemah.tuz.zahra@gmail.com

Abstract – The main focus of this paper is to analyze and discuss the most common secure software development practices currently being adopted in the industry along with their significance, as well as to identify the challenges faced by developers when undertaking measures and techniques in writing secure software. It is a well-known fact that software security has been the top priority of many software companies such as Google and Facebook to thwart attackers and protect user data in this world full of cybercriminals. Understanding how most software companies in the industry operate to ensure security helps developers to identify strengths and weaknesses in their current security frameworks. Hence, by researching into previous literature and papers that are relevant to the topic and by conducting an interview with a professional in the field, this paper provides insights on the most popular secure software development framework and practices in the world as well as problems faced by companies when adopting these practices. Several security practices and activities that are required to create secure software are discovered alongside the problems that arise when companies are trying to apply these practices. This paper also proposes a few solutions that can be used to resolve these problems, which can be easily understood and implemented by software companies to transition into a truly secure software development environment.

1 Introduction

Computer system has become increasingly ubiquitous nowadays with almost every field and industry relies on the use of software system to some degree to sustain business operations and to boost productivity. It is nearly impossible to think of any service or product that does not involve the use of a computerized system in some way, as computer system has been integrated deeply in most business environments to serve as the central of control and coordination for tasks such as management, automation and complex calculations [1]. For instance, financial institutions employ computer system to manage and keep track of transactions, digital currency, and commodities trading that requires little to no user intervention whereas military sectors also depend heavily on the use of software to encrypt communications, simulate military tactics and to pilot unmanned drones [2][3]. Not only that, the use of software and computers, wireless networks and smart internet of things gadgets [4][5] has also infiltrated the lives of the population worldwide in the form of smart homes devices, healthcare devices, smart personal gadgets like watches, etc. As of 2020, 45% and 59% of the global population owns a smartphone and is an active internet user respectively. [6][7] Software and computer system have truly revolutionized the way the world operates, making our lives more convenient and interconnected than ever.

Software has indeed helped in every aspect of our lives, but it is also a double-edged sword. Software has been put in charge of a massive amount of information ever since computerized system started getting popular, from less sensitive data such as username and phone number to more critical information such as credit card numbers and company trade secrets. Therefore, software system if misused or if used inappropriately could lead to devastating effects such as reputational damage, identity theft, heavy economic loss and could even compromise the military defense system of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2020 doi:10.20944/preprints202009.0478.v1

a nation. [8] That being said, there exists cyber attackers and clandestine users who are actively looking for ways to exploit a software system either for personal gains or to help achieve the goals of their organizations. For instance, just recently on 24th April 2020, the video game company Nintendo had its server under attack which resulted in the breach of account information of more than 160,000 users, including their login IDs, passwords, and credit card information. [9] On the larger scale, hackers that work for the North Korean government have been robbing banks from around the world by tampering with the log files and transaction records, as well as by holding valuable data for ransom using ransomware. [10] In fact,[11] reveals that 86% of breaches were financially motivated while 27% of the breach incidents are related Ransomware.

Software is vulnerable by nature because of their size and complexity which may contain vulnerabilities that are undiscovered or overlooked by developers. Cyber attackers exploit these vulnerabilities with the help of social engineering or backdoor viruses to gain unauthorized access to the system [12]. The earliest computer virus could be traced back to the Creeper Program designed as a security test in 1971, the Code Red worm that spreads itself through the internet in 2001, and more recently the WannaCry ransomware attack that encrypts and holds user data for ransom in 2017. [13] With the help of the internet especially, launching a cyber-attack has become very easy for attackers as they can masquerade themselves and launch the attack at a distance which prevents their identity from being exposed. As such, the security of software should be highly regarded to prevent giving any opportunities to the attackers to hack into the system.

Secure software is one that can fulfill the CIA triad, namely Confidentiality, Integrity, and Availability. Confidentiality ensures that information is accessible only by authorized individuals, Integrity ensures that information is correct and is not modified in an unauthorized manner whereas Availability ensures that services and resources are available as needed and are not denied to authorized users. Failure to comply to the CIA may result in the software being vulnerable to all kinds of cyber attacks such as eavesdropping and denial-of-service attack [14]. If such insecurities become a part of systems involving internet of things which are exponentially interconnected, they can lead to disastrous effects, such as loss of sensitive data, manipulation, identity theft, etc. Such systems are relatively more vulnerable to security and privacy issues [15].

Unfortunately, very often software security is not given enough emphasis by organizations during the development phase as it does not relate to the direct functioning of a system. Instead, organizations usually release security patches only after the development has finished when major security loopholes are found, which often introduces more flaws. It was only after Microsoft introduced the Security Development LifeCycle (Fig.1.) in 2004 that companies started to put more effort into building more secure software by embedding security-related activities into each phase of the SDLC. [16] Examples of such activities include threat modeling, security testing, and risk assessment. [17] Because of this initiative, worldwide spending on information security and cybersecurity have reached $ 114,152M in 2018, [18] which shows that companies have now realized software security is equally as important as its functionality, because “Software security is an emergent property of a complete system, not a feature” [19].

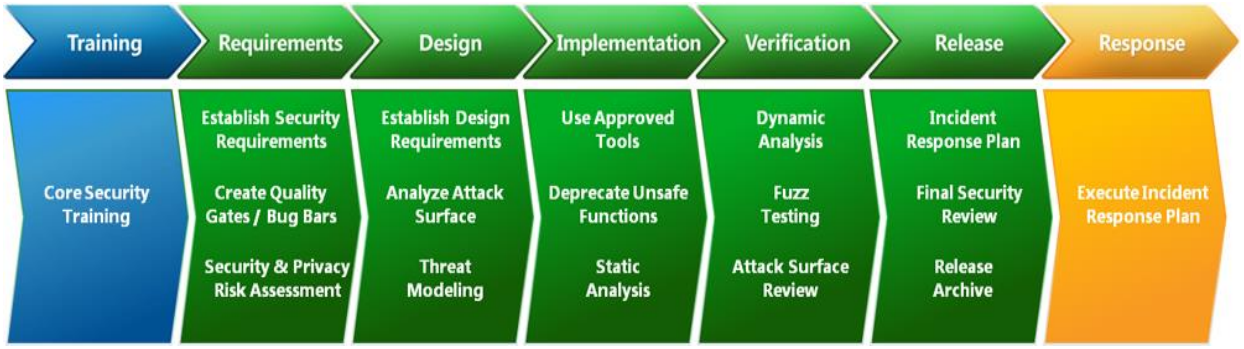

Fig.1. Security Development Lifecycle by Microsoft

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2020 doi:10.20944/preprints202009.0478.v1

Based on the above discussions, it is clear that software security is very important and should not be disregarded under any circumstances. Developers nowadays also have to comply with security policy, standards, and guidelines in each development phase depending on the security model being used to ensure that the software being developed is free of vulnerabilities and security bugs. As such, there is a need to research into each kind of the secure software development practices to see which are the most common in the industry and their significance, as well as to identify the challenges faced by developers when trying to comply with these practices. Despite the fact that there are several solutions available to combat security attacks in computer systems and internet of things [20][21], it is necessary to implement security practices in every phase of a system starting from planning.

2 Literature Review

In general, software developers disregard the concept of protection. This usually contributes to many of their programs having weak defenses [22]. Given the overwhelming amount of attacks that continually impact companies and governments, it is important to make sure that the technological systems these institutions use are secure. Secure software development is a method that means that the programming and procedures that go through application creation are as safe as practicable. With the use of safe coding practices and a security development lifecycle, programs can be made safe and secure for years to come There is an off the shelf solution that offers a systematic road map to protect applications – The secure development life cycle (SDL). It is a collection of development techniques designed to strengthen the safety of the program [23]. These procedures should be incorporated in all phases of software development and maintenance to get the best return. Most Security development lifecycles have similar approaches amongst them. A typical Security development lifecycle has 6 main phases.

- Planning phase
- Design phase
- Implementation phase
- Testing and bug fixing
- Release and Maintenance
- End of life

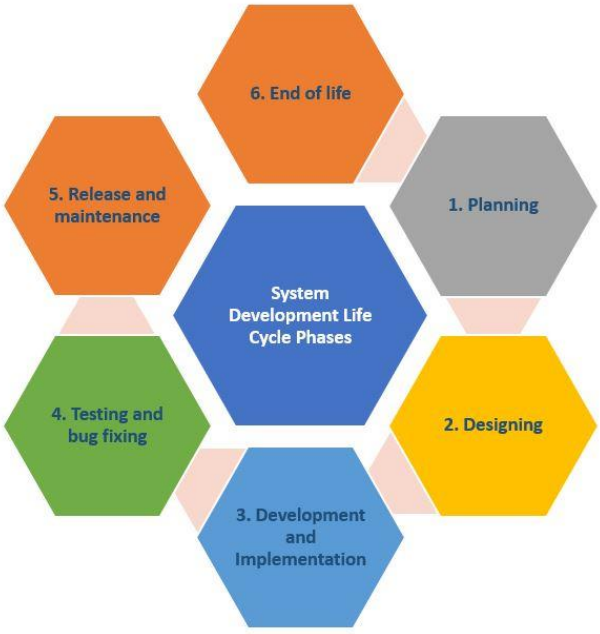

Fig. 2. Conventional SDLC phases

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2020 doi:10.20944/preprints202009.0478.v1

Most of the methods which reinforce the security of software work great at particular phases. Therefore, it is necessary to prepare ahead of time. Safe techniques for creation come in handy here.

2.1 Concept and Planning

The goal of this phase is to identify the principles of the application and to assess its viability. This includes the development of a project plan, and writing of design specifications. Security components like identification of security requirements, development of Misuse case and security training provision are also included here (Fig. 3).

First, it is important to figure out the type of SDL which will suit the software that is to be developed. It is recommended to adopt a Capability Maturity Model. Capability Maturity Models include a comparison model for a given engineering discipline of mature processes. To define possible opportunities for change a company should equate its activities with the pattern. There are generally 3 types of CMMI.

- *CMMI-ACQ* provides the acquisition organizations with development guidelines to facilitate and handle the acquisition of goods and services.
- *CMMI-SVC* provides service provider organizations with better guidelines for developing, maintaining and providing services
- *CMMI-DEV* offers the current best practices for the development, maintenance, and procurement of goods and services, including tools to help companies enhance their processes and include benchmarks for assessing process capacity and process maturity.

A. Identifying security requirements

It is a good practice to define the security requirements that would be needed for developing the application. Requirements engineering has always been crucial to project success for any large project. There are three types of security requirements that need to be identified. They are; Functional security requirements, Non-functional security requirements, and Derived security requirements. The project team will evaluate how protection can be incorporated into the development process during the necessary period. The security requirements are listed during the requirement phase, and some follow eventually from the development of threat models, industry standards such as IEEE, NIST, or ISO.

B. Developing Misuse case

Security requirements are sometimes defined as what should not be done by the system while functional requirements describe what should be done by the system. Use cases are created to describe functional requirements so they are good at clarifying what the system should be doing and not what system should do. As quoted by Guttom Sindre, Andreas L. Opdahl "A Misuse Case is the inverse of a use case i.e. A function that a system should not allow" [24]. Misuse cases are quite helpful in eliciting security requirements. They will recognize and distinguish between, various forms of attackers. Often, they describe the status of a device before and after the assault.

C. Providing security training

Everybody needs to be educated, from upper executives to customer service, developers to testers. The entire company should be aware of the significance of security. If the upper executives are not critical to safety, then they will not spend time and effort on it. Security training should be divided into two categories; General training and Specific training [25]. General training is for the overall organization. It aims to promote knowledge about safety in the environment of the organization. General training will prove effective if everyone in an institution begins to think about safety in their plans, schedules, and actions. Specific training is restricted just for the project team. Specific tasks deal with unique preparation. Each task, based on its nature, brings in certain security risks and weaknesses. Specific training is aimed at identifying these flaws; threats linked with them and identifying security steps to be taken to safeguard application from these attacks.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2020 doi:10.20944/preprints202009.0478.v1

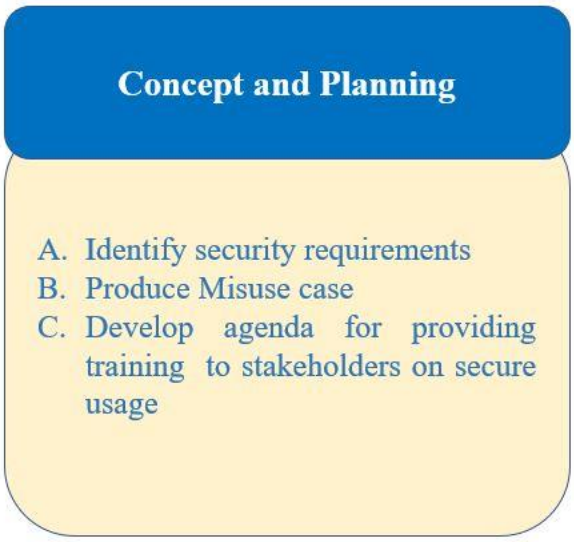

Fig. 3. Security components in Planning phase of SDLC

2.2 Design Phase

The design process works on documents created during the process of specifications. The design process emphasizes on 'how' and explains how the specified function should be enforced during the requirement phase. Throughout the design process, the project team maps out the product functionality defined in the case study for usage. It involves designing the framework layout and its implementation situations, as well as selecting components from third parties that can speed up the creation of the software. A design document is made during this stage [26].

Best practices to be used during the design phase are listed as follow and shown in Fig. 4:

A. Develop a threat model

Based on cases of abuse identified during the Requirement phase, threat models are established to classify the sources of risks, their implementation requirements, the circumstances essential for them to exist, and the effect they can have on the program. It is a risk estimation that guides security architecture, code review, and testing. Microsoft asserts that the highest priority aspect of SDL is threat modeling [27]. A great way to catch the pattern of threat is to use the attack trees. An alternate solution is the STRIDE model, which aims to classify risks by the study of multiple categories; which are: Spoofing, Tampering with data, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. Another such approach to assessing growing risk is utilizing DREAD scores. DREAD identifies five main attributes for calculating the ratings. Which are: Damage potential, Reproducibility, Exploitability, Affected users and Discoverability.

B. Support for automated tools

Automated software engineering resources will be to support SSD. The usage of common research and configuration methods helps to define protection approaches at SDLC's product criteria specification stages. Furthermore, it is proven that static analyzers and interactive support for programmers reduce software bugs that could result in the making of unsafe software. Nonetheless, to integrate with the new production environment, automatic tool support must be functional, less technical, accessible, and customizable. The organization will ensure that developers are presented with appropriate tool-related preparation and knowledge and ensure that they implement a method that will result in market benefit [28].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2020 doi:10.20944/preprints202009.0478.v1

C. Develop an encryption strategy

Encryption is the most popular way of protecting data from unplanned confidentiality or modification, regardless of whether the information is located or sent. While encryption can be constructed retroactively into a feature, it is simpler, more convenient, and more cost-effective to recognize encryption during the design process [29]. There are several components to consider when creating an encryption strategy.

D. Define what to be protected

At the very minimum, all internet data will be encrypted when in motion, and communication inside private networks should always be encrypted. Organizations should create specific guidelines for which types of data should be encrypted and which methods are appropriate for data protection.

E. Define the type of mechanism to be used for encryption

Despite the common use of encryption, proper implementation continues to remain quite difficult. Instead of needing developers to find the right choice at the time of execution, management should create strict guidelines of encryption that provide specifics on each element of encryption. Only industry-standard encryption libraries should be used, and only solid, unbreakable algorithms, key lengths, and cipher styles should be authorized.

F. Define the key and certificate solution to be used

Data encryption is only a part of the encryption technique. The other component is the security key and credential management system. Every entity that can obtain an encryption key or certificate is an entity that can enter the encrypted information, and so creating a mechanism for handling the keys and certificates will monitor who has control and develop a simple audit log of that access.

Fig. 4. Practices to be used for secure design phase in SDLC

2.3 Implementation Phase

The final step of deployment is where security bugs will reach the program. If secured, then the possibilities of software entry flaws will be significantly reduced. This is the stage where an application is being created. This means

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2020 doi:10.20944/preprints202009.0478.v1

preparing the application code, error checking it, and creating the test-fitting stable builds. The implementation phase begins with secure code writing followed by a review of static code using automated tools and manually reviewing the codes.

A. Write Secure code

The development team needs to be informed about security vulnerabilities in the implementation level. The development team should know the effects of security flaws at the implementation level, how they take place, and what effect they may have on the application. Developers must use a threat model approach to predict serious risks while coding correlating functionalities to take extra care [30]. One aim of designing stable applications is to reduce the number of bugs involved with unintended vulnerability at the coding stage.

B. Using safe functions only

Many programming languages have functions and APIs whose safety consequences were not appreciated at the beginning but are now commonly deemed unsafe. Although C and C++ are recognized to have several hazardous strings and buffer manipulation mechanisms, several other languages at least have some functions that are hard to use safely.[31] Developers should be provided training on what functions to prevent within the coding standards and also show their safe counterparts. Similarly, third party tools should be implemented to better locate and evaluate the use of hazardous functions. Most static analyses and linting methods have a framework for defining usage during the construction period.

C. Handle errors

At a certain stage all programs deployed into errors. While typical usage errors will be recognized during the testing phase, it is nearly difficult to foresee all the ways an attacker might interact with the application. With this in mind, a core aspect of every application is to manage and respond in a managed and elegant manner to unexpected errors, and either recover or send an error code [32]. Error handling should be incorporated in the logging method and various levels of specific information must ideally be supplied to clients as error codes and to log file administrators. The specific aspects of the issue should not be disclosed when alerting the user of an error [33]. Error messages should be generic to users and should preferably guide devices to accomplish a useful action from a usability point of view. This gives almost no information to an attacker about what went wrong and instructs authorized customers to what they should do next.

D. Perform Static code analysis

Static code testing is carried using automated tools. This is a simple, inexpensive, and efficient way of detecting popular code security vulnerabilities. There are computer vulnerabilities that are found very quickly by automatic devices. Such vulnerabilities include buffer overflows, variables not initialized, integer overflow, etc. [34] It is worth noting that the use of security tools does not secure the software, it only helps the software to be secure. It looks for common issue trends simply based on the software architecture, instead of the application 's behaviour while it is running

E. Perform a code review

Code reviews are conducted to identify security vulnerabilities in the code left after an evaluation of the static code. Automated tools used for static analysis has limited capabilities. They can find only the weaknesses to which they are intended. Application teams perform manual code reviews to support automated tools [35] A review of the code will take three measures. The first step is to re-run the static analysis tool and look for created errors and alerts. Some warnings might be mistakes or mistakes that are not seen. The second step is to look for known threats. Arithmetic issues, buffer overflows, cryptographic issues, and SQL injection are the most noticeable vulnerabilities. Finding these weaknesses can remove great many potential security holes. The third stage is to conduct an in-depth study of vulnerable areas

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2020 doi:10.20944/preprints202009.0478.v1

2.4 Testing phase

Security testing is vital and plays a major role in detecting security vulnerabilities before application publication. Security tester will think like an intruder and attempt to conduct different attacks to find vulnerabilities in the operating system. When the program reaches the test process, it is complete in functional terms. We cannot now protect software by stopping flaws from reaching the system but what we can do is detect vulnerabilities that already reside in the program.

Best practices to be used during the testing phase:

A. Create a test plan

Test preparation may be considered as the most critical step of the testing phase. Tests allow the use of cases of abuse and danger models during test preparation to detect potential threats, how they can be conducted out, and their effects. Testers then ready for study texts. The test schedule is established during project preparation [36]. Study scenarios are given precedence. The pattern of execution of test cases is established. Scope research is established for multiple areas. Safety test preparation is not just about developing test cases but also about effectively coordinating the whole protection monitoring operation.

B. Dynamic Testing

Security checking for dynamic analysis operates against an operating version of a system or service, usually implementing a sequence of pre-configured attacks in a restricted yet automatic variant of what a human intruder would attempt. These tests run against the completely compiled or processed software as it runs, and thus dynamic analysis can evaluate situations that only become apparent when all the systems are integrated. DAST investigates the actions of the program when it operates, and not the syntax of the language(s) in which the program is written, DAST will verify code written in a language that does not yet have strong SAST support [37].

C. Fuzzing

Fuzzing is a specific subset of DAST. Fuzzing is the process of creating or morphing data and forwarding it on to the data parsers in a program to see if they are reacting. Although it is impossible that any particular instance of a fuzzed input would uncover insecure behavior, continuing the procedure several amounts of times, usually will. The advantage of fuzzing is that it will run endless different checks against code until the software is set up, with the only expense being CPU time. Fuzzing generates much better parsing code quality than either conventional DAST or SAST [38].

D. Perform penetration testing

Penetration monitoring tests applications in a manner close to the way researchers search for bugs. Penetration testing (Fig. 5) may detect the greatest range of flaws and may evaluate a software program or device in the wider sense of the system under which it operates, a behavior it conducts, objects it communicates with, and aspects in which humans and other devices communicate [39]. It makes it well adapted to finding flaws in business logic. Inviting a third-party team of technology experts to test future threats is a smart idea. Independent experts rely on their know-how and experience to replicate scenarios of assault that your team may miss. Implementing these procedures further eliminates safety issues. This offers good protection against a diverse variety of known threats, in combination with the practices from the previous phases.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2020 doi:10.20944/preprints202009.0478.v1

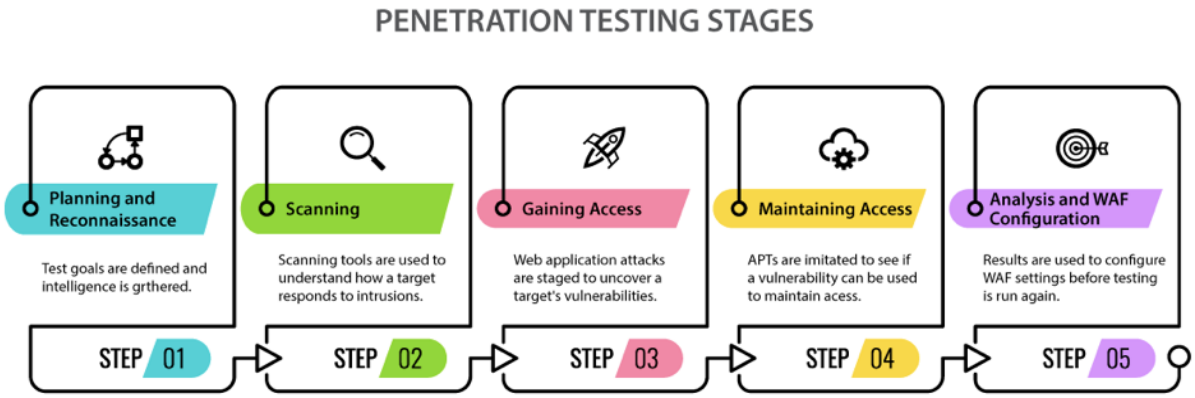

Fig. 5. Penetration testing stages [40]

2.5 Release and Deployment phase

When the program enters the release phase, it is believed that previous phases have found most security vulnerabilities, and the program is reliable and fit for deployment. But the program is subject to safety analysis and review before deployment. This security review and audit sought to assess any remaining security flaws and develop credibility over the software. The application is going live at this stage, with many instances running in a variety of environments. Ultimately new updates and patches are available, and some consumers chose to update, while some prefer to retain previous models.

A. Conduct a security review

The software is susceptible to a security analysis after the testing phase. This analysis has the purpose of identifying any surviving security vulnerabilities. That doesn't mean a whole process of safety assessment will be performed. Security assessment needs to be well prepared, so it won't make a deal of time and resources. Throughout the analysis, the program is evaluated in all its condition. Security vital components undergo thorough study and bugs identified previously are inspected. Security audit can be regarded as the last exercise in which security software is evaluated. The next individual to recognize a security vulnerability after security check should be an assailant

B. Develop an Incident response team

Larger companies frequently set up committed product safety response teams or incident response teams whose contract is to design and analyze the response and disclosure operations of the vulnerabilities. Smaller organizations will need to adequately measure the size of this attempt, based on the product's magnitude and the number of cases. Although PSIRT representatives should know all policies, guidelines, and information relating to vulnerability resolution and reporting, and therefore should be able to direct the software developer through the process, anyone involved in organizational application development and supporting functions should understand their position and duties as well [41].

C. Ensure that reporters with security vulnerabilities understand Who to approach

Security vulnerabilities reporters, customers, or other interested parties must know where, how, and to whom to submit the threat, to help start the proper answer to a possible security weakness. Because vulnerability identification can be used to target vulnerable goods, sensitive vulnerability information should be privately reported where necessary. Inside the internal and external vulnerability response policies, secure communication techniques such as encoded email addresses and public keys should be properly stated.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2020 doi:10.20944/preprints202009.0478.v1

2.6 End of life phase

"End of Life" is the stage where the developer stops supporting/releasing updates for the program. Relevant end-of-life regulations can apply to applications that hold sensitive information. Fig. 6 shows some security components that can be implemented in all phases to convert a typical SDLC into its secure version.

SDL activities recommended for this stage include:

A. Data retention

Governments identify the retention policy for certain types of data. Double-checking the retention practices of the company following the requirements of the provisions eliminates the possibility of unforeseen fines. It is advised to follow the ISO 15489-1:2001 standard [42]. It states that retention requirements and procedures of care should be structured to protect documents from unauthorized access, loss or damage, misuse, and tragedies. Besides, Organizations should provide guidelines and policies for transferring or migrating documents from one digitizing program to another

B. Create a data disposal policy

By the termination point of the program, all confidential data contained in it will be properly erased. Definitions of these data include encryption keys and private information. Appropriate end-of-life management of data keeps these details secure and avoids data breaches. Both consumer data will be disposed of until it is no longer required for commercial operation, providing the disposition does not interfere with the data protection policy, the data retention policy of consumers, a judicial order, or other legal requirements. It is advised to implement a secure way of disposing data.

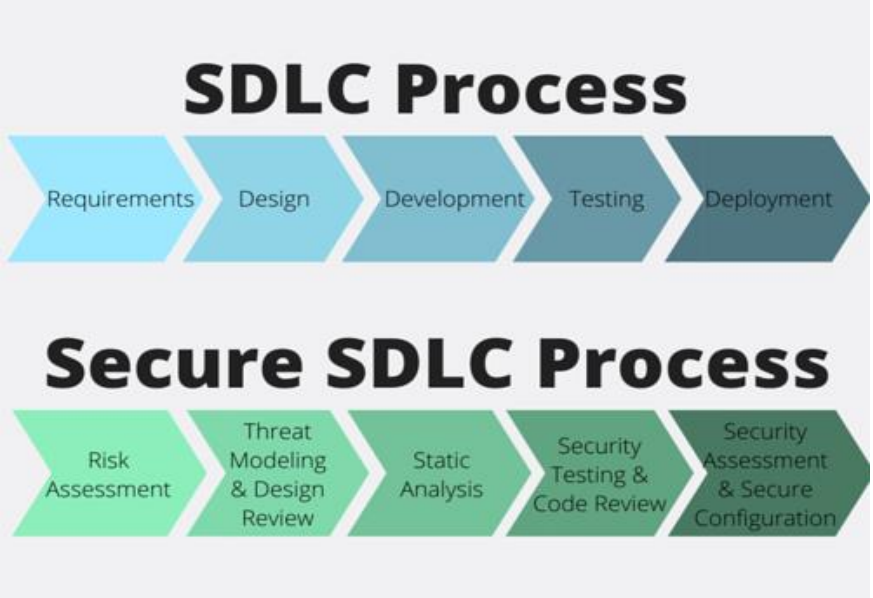

Fig. 6. Implementation of security components in all phases of SDLC [43]

3 Research Methodology & Data Collection

In the previous section, we have conducted a literature review on articles related to the most common practices in developing secure software. These findings are used as a reference for the rest of the research for comparative analysis.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2020 doi:10.20944/preprints202009.0478.v1

A. Systematic Review of Literature

This research method is done by analyzing, combining, and comparing the data from previous studies to derive a concrete conclusion about the topic. This allows the research to identify the most effective and most used security models that software engineers should adopt, as well as to provide a benchmark for software firms to assess the effectiveness of their current secure coding practices. The analysis was based on two articles obtained from credible sources like Researchgate and Semantic Scholar titled *"Surveying Security Practice Adherence in software Development"* and "*The practice of secure software development in SDLC: an investigation through existing model and a case study: The practice of secure software development in SDLC".* The authors of these two articles have done extensive study and survey to find out the most commonly adopted practices in secure software development based on BSIMM and MS SDL standards, thus comparing and analyzing the two will provide great insight into this research.

B. Interview

The interview was conducted with a professional in the field of software development to understand the context of each secure development practices, such that how they are implemented and embedded in the SDLC as well as how each practice relate to or differ from each other. This interview is also important to discover the issues or challenges faced by the development team when developing a secure software system, hence it would be optimal to obtain the information directly from an experienced individual. Below is the transcript of the information exchanged during the interview:

1. What is your occupation and which company do you work for?

Answer 1:
"I work as a software engineer in a software firm based in Selangor that specializes in developing POS and online ordering system. We also do custom business software for clients."

2. Do you follow any specific SDLC models when developing software?

Answer 2:
"We don't have a fixed approach to SDLC because we have to deal with quite a lot of impromptu production issues especially when handling financial data and integrating the different types of payment gateways into our system. However, I would say that our company culture revolves loosely around the Agile methodology because we would often have a daily sprint or short meeting where everyone just gathers around to report on their daily progress and discuss the issues that they're facing. Of course, there's also the regular requirement engineering phase, design phase, etc., but they are not followed exactly, sometimes the phases overlap and sometimes we do things as needed."

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2020 doi:10.20944/preprints202009.0478.v1

3. Do you follow any security guidelines when implementing the software?

Answer 3:
"We do have an Application Security Team that deals with security-related stuff like creating secure libraries and make security-related changes in the codebase. They also work with the project owners, project managers, and system analysts to come up with security requirements, security modeling, and risk assessment. Often time they dictate the security portion of the code like what data to encrypt and what encryption methods and security tools to use so the rest of the development team just follows accordingly. As for us regular software developers, we are required to classify data based on their sensitivity like financial info and customer data so that the data are handled in the most secure and most appropriate way possible. Also, we need to follow the coding standard strictly, especially for those who are in charge of the web application backend to minimize the possibility of introducing vulnerability in the system. We use a software called Fortify that acts like a spell checker for the code to ensure that there's no careless vulnerability, and the AppSec team would also periodically review the code to ensure that they are in compliance to the standards and so that they can notify the development team if there are any amendments to be made. Then before we deploy the software, the AppSec team will perform all kinds of security testing like pen testing and such to identify security bugs or vulnerabilities in the system, which the QA team will then take over to perform unit and integration testing to ensure that the software passes the quality gate that was defined early on."

4. Do you face any challenges particularly when following the security guidelines?

Answer 4:
"Certainly, one of the biggest issues that we face is miscommunications. Very often the security requirements made by the AppSec team would clash with the software requirements. There was once when we were developing a time-sensitive software where a particular operation needs to be completed within a short duration. However, that duration was exceeded when we implemented the security requirements as demanded by the AppSec team. We ended up having to rework that particular function which resulted in schedule slippage. Also, the AppSec team either doesn't do the documentation or they don't do it properly. There were a few cases where the AppSec team did not document some of the security changes that they've made which caused quite a lot of confusion among the development team. In the end, we had to roll back to the previous version because the security changes had caused some functional errors. Then there's also software engineers who don't follow the security policy, especially the interns and the juniors. Often time their code is so buggy that a senior had to step in to help rectify the errors. I think this is because they lack the necessary training in writing secure software."

4 Results & Discussion

4.1 Systematic Review of Literature

It was discovered that most software companies in the US employ security practices as documented in Microsoft SDL, followed by BSIMM, SAFECode, and lastly CLASP. Out of all the security practices, Apply Secure Coding Standards came out on top with 45% reported daily use, Perform Security Review came after and is followed by Perform Security Testing. [44] There are also several other practices with roughly the same usage, including Document Technical Stack, Apply Security Tooling, Apply Security Requirements, Track Vulnerabilities, Apply Threat Modeling, and Provide Security Training. [44] Interestingly, respondents are suggesting that Provide Security Training be removed as training

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2020 doi:10.20944/preprints202009.0478.v1

delivered via lecture is useless. Instead, mentorship and practical experience is the best way to learn. [44] The result is consistent with the findings of the other paper, whereby Following secure coding checklist is strongly agreed by every respondent, and that some organizations do not possess enough security understanding to define security requirements due to the lack of experience. [45] Based on the findings, Apply Secure Coding Standards is the most adopted security practice because it is more of a good habit for general software development. It is the least that developers can do to keep their software secure, hence the popularity. Another strategy is to develop and integrate robust authentication schemes [46] into the systems, use smart technologies to develop efficient models for secure and lightweight systems [47].

Many organizations have released secure coding guidelines to help developers in writing secure code, such as SEI CERT C and C++ as well as Oracle Java Coding Standards. [48] These standards can be enforced by using SAST tools which helps developers in keeping compliance to the guidelines, [48] such as to sanitize and validate input. [49] Besides, there are still developers who do not have the necessary knowledge to apply security requirements. It could be due to the lack of relevant training, or that they are not exposed to common security practices. According to NIST, 64% of software vulnerabilities are caused by programming errors and bad coding practices which, as a result, the quality of the software deteriorates, and testers are more burdened during testing. [50] Hence, this issue must be curbed. Secure software practices usage frequency by source are shown in Table 1.

Table 1. Secure Software Practices Usage Frequency By Source [43]

SPEFP Practice	Source				
	BSIMM	CLASP	MS SDL	SAFECode	Total
Apply Secure Coding Standards	10	2	68	9	89
Perform Security Review	23	0	21	0	44
Perform Security Testing	10	3	20	4	37
Document Technical Stack	14	6	4	7	31
Apply Security Tooling	11	1	12	2	26
Apply Security Requirements	7	11	7	0	25
Track Vulnerabilities	16	0	8	0	24
Apply Threat Modeling	9	4	5	1	19
Provide Security Training	13	2	3	1	19
Improve Development Process	14	0	0	0	14
Perform Penetration Testing	9	0	2	1	12
Apply Data Claasification Scheme	11	1	0	0	12
Publish Operations Guide	4	4	2	0	10
Apply Security Principles	0	1	4	3	8
Monitor Security Metrics	1	4	0	0	5
Public Disclosure Policy	0	2	0	0	2
Excluded	68	0	14	3	85

4.2 Interview

Based on the interview, it was pleasant to know that secure software development is being practiced everywhere else in the world, including Malaysia. The company of the interviewee did adopt secure software development practices, many of which are consistent with the practices found in the systematic review, including apply secure coding standards, data classifications threat modeling, security testing, and apply security requirements. Most security-related tasks, however, are undertaken by a dedicated security team. The developers also rely on the use of SAST tools to scan for vulnerabilities as they code, which certainly helps boost productivity. Similar to the systematic review, the interviewee is also facing many challenges in these practices. One of which is the miscommunication that happened

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 September 2020 doi:10.20944/preprints202009.0478.v1

between the application security team and software development team which resulted in the conflicts between security requirements and software requirements. Miscommunication is one of the most common issues in software development which can lead to poorly made software and missed deadlines, [51] as is the case with the interviewee. The next problem is the lack of documentation which has resulted in a rework from scratch, as no documentation equates to no knowledge transfer and other developers won't know what has been done or changed. [52] The last problem is that some developers are not following the security policy, especially interns and junior developers, again due to the lack of training.

5 Solution

One of the biggest issues when it comes to secure software development is the lack of security training for software developers. According to The Forrester Report, secure software design and secure coding is not part of the mandatory course requirements in the top five international schools for computer science. [53] In fact, persuading developers to consider security in their projects has been a difficult task for security experts, let alone reaching out to an estimated 23 million developers in the world. [54] That said, there are many ways that the issue can be solved:

A. Integrate Security Review and Testing into Implementation Phase

Companies should implement periodic security test or review on the developer's code as they write so that developers are able to receive instant feedback regarding the quality of their code, thus reinforcing their concepts on security. [55] This test can be implemented by using an automated spell-checker tool, have a security expert scrutinize the code upon turning in or by doing pair programming that is popular with agile methodology, with two programmers working on the same code and periodically switching the roles of driver and observer. [56] Pair programming is especially effective as studies have shown that collaborative pairs have 15% fewer defects in their code and that 95% of the programmers are able to code more confidently with pair programming. [57]

B. Security Mentorship

Since most developers do not learn enough security techniques in school, companies should prepare a team of security experts or mentors to help new developers in getting started with security concepts and procedures, especially interns and junior developers who do not have much hands-on experience. [55] Mentors can teach by incorporating lessons when mentees are coding, frequently review, and inspect mentee's code as well as provide feedback whenever possible. [55] Mentorship is very effective in helping developers to build a strong foundation in terms of software security because mentors have enough experience to stress test mentee's abilities and point them in the right direction which prevents them from receiving the wrong info and/or using bad coding practice, which is often the case with developers learn through self-study [58].

The other issue regarding practicing secure software development arises from improper implementation of SDLC methodologies, especially agile. From the interview conducted with the professional in the field, it is discovered that his company does not have a strict approach to SDLC (though they lean more towards agile), which has resulted in miscommunication between the security team and the software team, as well as a lack of documentation that has ultimately caused missed deadlines [59] reveals that narrow communication between departments, the unwillingness of the team to show their work, and backloading documentation are some of the most common problems that companies face when adopting agile methodology, as is the case with the interviewee's company. Agile methodology is often teased as not having enough documentation because agile emphasizes flexibility and that changes occur so often that it would be impractical to document everything in black and white. [60] As such, any security practices during secure software development must be adaptive to the agile methodology so that the development process is not obstructed. [61] This research proposes a few methods that help to integrate security approach better into the agile methodology:

C. Utilize Automated and Pro-active Tools and Software for Security

Automated security tools such as SAST/DAST and vulnerability scanner can automate the process of securing software by providing constant monitoring, identifying vulnerabilities and security loopholes as well as generating an automated report with little to no user intervention. [62] This can effectively reduce the workload of both security engineers and software developers since any doubts or confusions are reduced and everything is already documented using automation. [61] These tools also allow the security team to enforce security policies which automatically applies to the development environment when the software engineers are coding, thus allowing the security team to locate and fix any vulnerabilities or bugs as soon as possible [61]. Hence, the problems of miscommunication and lack of documentation are effectively mitigated. Another pro-active strategy that can be used is to implement pre-encryption attack detection algorithm within the system to secure it from attacks like ransomware as proposed in [63].

D. Educate Developers on Software Security and Secure Software Design

The fundamental cause of security not being integrated well in an agile software project which often results in the clash of software requirements and security requirements is still the lack of security training for developers. Most developers often fail to see the software that they are developing from an attacker's perspective because they lack exposure to how software can be exploited and broken into. [61] This can be especially dangerous for agile development that emphasizes flexibility and speed which developers may often overlook the security aspect of the software to prevent missing the deadlines. [64] As a result, many potential threats are unintentionally introduced into the software [61]. Thus, agile developers must be trained to be able to think from the perspective of a hacker.

6 Conclusion

Cyber attacks are getting increasingly sophisticated and securing software using “security through obscurity” is no longer sufficient to safeguard software from being targeted by cyber attackers. Nowadays software companies have widely adopted the use of secure software development frameworks and practices to produce software that can protect customer privacy while also fulfilling data confidentiality, integrity, and availability. Many security practices have proven to be very effective in reinforcing software security and has since become the standards for the industry, including secure coding checklists, threat modeling, security testing, and many more. A plethora of security frameworks and benchmarks such as Microsoft SDL and BSIMM have also been introduced to help software companies in deciding what security model should be implemented and to assess how far their current security practices can keep their software safe and secure. Although there is still quite a bit of challenge when it comes to practicing secure software development such as the lack of security training and exposure among developers as well as the improper implementation of SDLC methods, these challenges can be overcome easily nevertheless if companies are willing to put more effort into making secure development a part of the company’s culture. All in all, there is not a permanent solution for secure software development as software technology is evolving day by day and so do techniques used by cyber attackers to break into the software. Instead, developers should always be ready to adapt to the latest trends and changes, so that they always have the best available security tools and techniques to produce quality software with robust security components.

References

[1]. Cetin and M. Ozden, "Development of computer programming attitude scale for university students", *Computer Applications in Engineering Education*, vol. 23, no. 5, pp. 667-672, 2015. Available: 10.1002/cae.21639.

[2]. "How are computers used?", *Computerhope*, 2019. [Online]. Available: https://www.computerhope.com/issues/ch001732.htm. [Accessed: 20- Jun- 2020].

[3]. Khan N.A., Brohi S.N., Jhanjhi N. (2020) UAV’s Applications, Architecture, Security Issues and Attack Scenarios: A Survey. In: Peng SL., Son L., Suseendran G., Balaganesh D. (eds) Intelligent Computing and Innovation on Data Science. Lecture Notes in Networks and Systems, vol 118. Springer, Singapore. https://doi.org/10.1007/978-981-15-3284-9_86

[4]. Z.A. Almusaylim and N. Zaman, “A review on smart home present state and challenges: linked to context-awareness internet of things (IoT) Wireless Networks”, 25 (6), 3193-3204. https://doi.org/10.1007/s11276-018-1712-5

[5]. M. Humayun, N. Jhanjhi, B. Hamid and G. Ahmed, "Emerging Smart Logistics and Transportation Using IoT and Blockchain," in IEEE Internet of Things Magazine, vol. 3, no. 2, pp. 58-62, June 2020, https://doi.org/10.1109/IOTM.0001.1900097.

[6]. C. J, "Global digital population 2020 | Statista", *Statista*, 2020. [Online]. Available: https://www.statista.com/statistics/617136/digital-population-worldwide/#:~:text=How%20many%20people%20use%20the,in%20terms%20of%20internet%20users. [Accessed: 20- Jun- 2020].

[7]. "HOW MANY SMARTPHONES ARE IN THE WORLD?", *BankMyCell*, 2020. [Online]. Available: https://www.bankmycell.com/blog/how-many-phones-are-in-the-world. [Accessed: 20- Jun- 2020].

[8]. T. Rid and B. Buchanan, "Attributing Cyber Attacks", *Journal of Strategic Studies*, vol. 38, no. 1-2, pp. 4-37, 2015. Available: 10.1080/01402390.2014.977382.

[9]. "All Data Breaches in 2019 & 2020 - An Alarming Timeline - SelfKey", *SelfKey*, 2020. [Online]. Available: https://selfkey.org/data-breaches-in-2019/. [Accessed: 20- Jun- 2020].

[10]. B. Buchanan, "How North Korean Hackers Rob Banks Around the World", *Wired*, 2020. [Online]. Available: https://www.wired.com/story/how-north-korea-robs-banks-around-world/. [Accessed: 20- Jun- 2020].

[11]. "2020 Data Breach Investigations Report", *Verizon Enterprise*, 2020. [Online]. Available: https://enterprise.verizon.com/en-gb/resources/reports/dbir/. [Accessed: 20- Jun- 2020].

[12]. D. Votipka, R. Stevens, E. Redmiles, J. Hu and M. Mazurek, "Hackers vs. Testers: A Comparison of Software Vulnerability Discovery Processes," *2018 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, 2018, pp. 374-391. doi: 10.1109/SP.2018.00003

[13]. "A Brief History of Computer Viruses & What the Future Holds", *Kapersky*. [Online]. Available: https://www.kaspersky.com/resource-center/threats/a-brief-history-of-computer-viruses-and-what-the-future-holds. [Accessed: 20- Jun- 2020].

[14]. S. Qadir and S. Quadri, "Information Availability: An Insight into the Most Important Attribute of Information Security", *Journal of Information Security*, vol. 07, no. 03, pp. 185-194, 2016. Available: 10.4236/jis.2016.73014.

[15]. Dhuha Khalid Alferidah, NZ Jhanjhi, A Review on Security and Privacy Issues and Challenges in Internet of Things, in International Journal of Computer Science and Network Security IJCSNS, 2020, vol 20, issue 4, pp.263-286

[16]. C. Romeo, "Secure Development Lifecycle: The essential guide to safe software pipelines | TechBeacon", *TechBeacon*. [Online]. Available: https://techbeacon.com/security/secure-development-lifecycle-essential-guide-safe-software-pipelines. [Accessed: 20- Jun- 2020].

[17]. "How to approach secure software development", *Ptsecurity*, 2020. [Online]. Available: https://www.ptsecurity.com/ww-en/analytics/knowledge-base/how-to-approach-secure-software-development/. [Accessed: 20- Jun- 2020].

[18]. "Gartner Forecasts Worldwide Information Security Spending to Exceed $124 Billion in 2019", *Gartner*, 2018. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2018-08-15-gartner-forecasts-worldwide-information-security-spending-to-exceed-124-billion-in-2019. [Accessed: 20- Jun- 2020].

[19]. J. Allen, *Software security engineering*. Upper Saddle River, N.J.: Addison-Wesley, 2008.

[20]. M. Almulhim, and N. Zaman, "Proposing secure and lightweight authentication scheme for IoT based E-health applications", 2018 20th International Conference on Advanced Communication Technology (ICACT), 481-487. https://doi.org/10.23919/ICACT.2018.8323802

[21]. Almulhim, M., Islam, N., & Zaman, N. (2019). A Lightweight and Secure Authentication Scheme for IoT Based E-Health Applications. International Journal of Computer Science and Network Security, 19(1), 107-120.

[22] Core.ac.uk. 2020. *Reusable Knowledge In Security Requirements Engineering: A Systematic Mapping Study*. [online] Available at: <https://core.ac.uk/reader/52813164> [Accessed 25 June 2020]. [2] Guttom Sindre, Andreas L. Opdahl, "Eliciting Security Requirements by Misuse Cases", IEEE Explore

[23]. Usenix.org. 2020. [online] Available at: <https://www.usenix.org/system/files/conference/soups2018/soups2018-assal.pdf> [Accessed 25 June 2020].

[24]. Sindre, G. and Opdahl, A., 2020. *Eliciting Security Requirements By Misuse Cases*. [online] Semanticscholar.org. Available at: <https://www.semanticscholar.org/paper/Eliciting-security-requirements-by-misuse-cases-Sindre-Opdahl/575879fcaba9d1ff770d71d1ad807a5cae1ca5a0> [Accessed 25 June 2020].

[25]. 2018. [online] Available at: <https://www.researchgate.net/publication/310471907_The_practice_of_secure_software_development_in_SDLC_an_investigation_through_existing_model_and_a_case_study_The_practice_of_secure_software_development_in_SDLC> [Accessed 25 June 2020].

[26] Gary McGraw 2015. [online] Available at: <https://resources.sei.cmu.edu/asset_files/WhitePaper/2015_019_001_434546.pdf> [Accessed 25 June 2020].

[27]. Docs.microsoft.com. 2020. *Threat Modeling For Drivers - Windows Drivers*. [online] Available at: <https://docs.microsoft.com/en-us/windows-hardware/drivers/driversecurity/threat-modeling-for-drivers> [Accessed 25 June 2020].

[28]. Morrison, P., Smith, B. and Williams, L., 2017. *Surveying Security Practice Adherence In Software Development*. [online] Semanticscholar.org. Available at: <https://www.semanticscholar.org/paper/Surveying-Security-Practice-Adherence-in-Software-Morrison-Smith/98e63067249fb4c13076f2481818890c7f39ff89> [Accessed 25 June 2020].

[29]. Safecode.org. 2018. *Fundamental Practices For Secure Software Development*. [online] Available at: <https://safecode.org/wp-content/uploads/2018/03/SAFECode_Fundamental_Practices_for_Secure_Software_Development_March_2018.pdf> [Accessed 25 June 2020].

[30]. Owasp.org. 2017. *OWASP – Secure Coding Practices, Quick Reference Guide*. [online] Available at: <https://owasp.org/images/0/08/OWASP_SCP_Quick_Reference_Guide_v2.pdf> [Accessed 25 June 2020].

[31]. Open-std.org. 2016. *Programming Languages — C*. [online] Available at: <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf> [Accessed 25 June 2020].

[32] Cwe.mitre.org. 2020. *CWE - CWE-388: 7PK - Errors (4.0)*. [online] Available at: <https://cwe.mitre.org/data/definitions/388.html> [Accessed 25 June 2020].

[33]. Cwe.mitre.org. 2020. *CWE - CWE-544: Missing Standardized Error Handling Mechanism (4.0)*. [online] Available at: <https://cwe.mitre.org/data/definitions/544.html> [Accessed 25 June 2020].

[34]. Zhu, J., 2020. *Mitigating Access Control Vulnerabilities Through Interactive Static Analysis | Proceedings Of The 20Th ACM Symposium On Access Control Models And Technologies*. [online] Dl.acm.org. Available at: <https://dl.acm.org/doi/10.1145/2752952.2752976> [Accessed 25 June 2020].

[35]. Morrison, P., Smith, B. and Williams, L., 2017. *Surveying Security Practice Adherence In Software Development*. [online] Semanticscholar.org. Available at: <https://www.semanticscholar.org/paper/Surveying-Security-Practice-Adherence-in-Software-Morrison-Smith/98e63067249fb4c13076f2481818890c7f39ff89> [Accessed 25 June 2020].

[36]. Pdfs.semanticscholar.org. 2013. *Secure Software Development*. [online] Available at: <https://pdfs.semanticscholar.org/d0be/605180eb14926e91003715cb3f535503aecb.pdf> [Accessed 25 June 2020].

[37]. Martina Lindorfer, M., 2017. *Efficient And Comprehensive Mobile App Classification Through Static And Dynamic Analysis*. [online] Available at: <https://iseclab.org/files/publications/Lindorfer2015Marvin_Efficient.pdf> [Accessed 25 June 2020].

[38]. Takanen,, A., 2015. *Fuzzing For Software Security Testing & Quality Assurance*. [online] Conference.eurostarsoftwaretesting.com. Available at: <https://conference.eurostarsoftwaretesting.com/wp-content/uploads/th13.pdf> [Accessed 25 June 2020].

[39]. BALOCH, R., 2020. *ETHICAL HACKING AND PENETRATION TESTING GUIDE*. [online] Tsoungui.fr. Available at: <https://tsoungui.fr/ebooks/Ethickal-haking-postexploitation.pdf> [Accessed 25 June 2020].

[40]. ThreatModeler (n.d.). 'Threat Model for Security Penetration Testing. [image]. Available at: https://threatmodeler.com/threat-modeling-for-security-penetration-testing/

[41]. 2.0, F., 1.1.1, F., Teams, F., Teams, F. and 1.1, F., 2020. *FIRST Services Framework*. [online] FIRST — Forum of Incident Response and Security Teams. Available at: <https://www.first.org/standards/frameworks/> [Accessed 25 June 2020].

[42]. Mountain, I., 2014. *Document Retention Guide United Kingdom*. [online] Iapp.org. Available at: <https://iapp.org/media/pdf/resource_center/Document_Retention_Guide_UK.pdf> [Accessed 25 June 2020].

[43]. Tal, Checkmarx (2014). Secure SDLC. [online]. Available at: https://www.checkmarx.com/glossary/a-secure-sdlc-with-static-source-code-analysis-tools/

[44]. P. Morrison, B. Smith and L. Williams, "Surveying Security Practice Adherence in Software Development", *Proceedings of the Hot Topics in Science of Security: Symposium and Bootcamp on - HoTSoS*, 2017. Available: 10.1145/3055305.3055312 [Accessed 21 June 2020].

[45]N. Karim, A. Albuolayan, T. Saba and A. Rehman, "The practice of secure software development in SDLC: an investigation through existing model and a case study", *Security and Communication Networks*, vol. 9, no. 18, pp. 5333-5345, 2016. Available: 10.1002/sec.1700.

[46]. Teoh Joo Fong, Azween Abdullah, NZ Jhanjhi, Mahadevan Supramaniam, "The Coin Passcode – A Shoulder-Surfing Proof Graphical Password Authentication Model for Mobile Devices", in International Journal of Advanced Computer Science and Applications (IJACSA), Vol 10, No, 1, pp. 302-308, 2019

[47]. M Humayun, NZ Jhanjhi, M. Z Alamri, "Smart Secure and Energy Efficient Scheme for E-Health Applications using IoT: A Review", International Journal of Computer Science and Network Security 20 (4), 55-74.

[48]. "Coverity Support for SEI CERT C, C++, and Java Coding Standards", *Synopsys*, 2016. [Online]. Available: https://www.synopsys.com/content/dam/synopsys/sig-assets/datasheets/sei-cert-c-coding-standard.pdf. [Accessed: 21- Jun- 2020].

[49]. S. Shrum, "Rec. 01. Declarations and Initialization (DCL) - SEI CERT Oracle Coding Standard for Java - Confluence", *Carneige Mellon University | Software Enginerring Institute*, 2017. [Online]. Available: https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487329. [Accessed: 21- Jun- 2020].

[50]. P. Humphreys and C. Tapp, "Coding standards to secure code in embedded systems", *Techdesignforums*, 2015. [Online]. Available: https://www.techdesignforums.com/practice/technique/coding-standards-for-secure-embedded-systems/#:~:text=According%20to%20research%20by%20the,vulnerabilities%20stem%20from%20programming%20errors.&text=The%20US%20federally%20funded%20organization,using%20C%2C%20C%2B%2B%20and%20Java. [Accessed: 21- Jun- 2020].

[51]. W. MILLER, "3 Common Issues with the Software Development Process", *Software Testing and Quality Assurance by iBeta*, 2019. [Online]. Available: https://www.ibeta.com/3-common-issues-with-the-software-development-process/. [Accessed: 21- Jun- 2020].

[52]. A. Trica, "The Importance of Documentation in Software Development", *Filtered*, 2018. [Online]. Available: https://filtered.com/blog/post/project-management/the-importance-of-documentation-in-software-development#:~:text=The%20presence%20of%20documentation%20helps,knowledge%20transfer%20to%20other%20developers. [Accessed: 21- Jun- 2020].

[53]. C. Klein, "How to give developers secure coding training | Synopsys", *Software Integrity Blog*, 2019. [Online]. Available: https://www.synopsys.com/blogs/software-security/secure-coding-training/. [Accessed: 22- Jun- 2020].

[54]. "Evans Data Corporation | Worldwide Developer Population and Demographic Study 2019 Volume 2", *Evansdata*, 2019. [Online]. Available: https://evansdata.com/reports/viewRelease.php?reportID=9. [Accessed: 22- Jun- 2020].

[55]. R. Lemos, "5 ways to better educate developers on application security | TechBeacon", *TechBeacon*, 2015. [Online]. Available: https://techbeacon.com/security/5-ways-better-educate-developers-application-security. [Accessed: 22- Jun- 2020].

[56]. "What is Agile Software Development?", *Agile Alliance*, 2020. [Online]. Available: https://www.agilealliance.org/agile101/. [Accessed: 22- Jun- 2020].

[57]. W. Laurie and R. R. Kessler, "THE COLLABORATIVE SOFTWARE PROCESS", *International Conference on Software Engineering*, 2015. [Accessed 22 June 2020].

[58]. V. Cassone, "How to find a mentor and accelerate your learning: a beginner's guide.", *freeCodeCamp.org*, 2018. [Online]. Available: https://www.freecodecamp.org/news/how-to-find-a-mentor-and-accelerate-your-learning-a-beginners-guide-1a0a41ca65e3/. [Accessed: 22- Jun- 2020].

[59]. S. Dhir, D. Kumar and V. Singh, "Success and Failure Factors that Impact on Project Implementation Using Agile Software Development Methodology", *Advances in Intelligent Systems and Computing*, pp. 647-654, 2018. Available: 10.1007/978-981-10-8848-3_62 [Accessed 22 June 2020].

[60]. S. Lowe, "15 signs you're doing agile wrong", *InfoWorld*, 2016. [Online]. Available: https://www.infoworld.com/article/3074332/15-signs-youre-doing-agile-wrong.html. [Accessed: 22- Jun- 2020].

[61]. R. Sass, "How to Balance Between Security and Agile Development the Right Way", *WhiteSource*, 2016. [Online]. Available: https://resources.whitesourcesoftware.com/blog-whitesource/how-to-balance-between-security-and-agile-development-the-right-

way#:~:text=The%20reason%20is%20that%20agile,secure%20software%20before%20it%20ships. [Accessed: 22-Jun- 2020].

[62]. S. Koussa, "What do SAST, DAST, IAST and RASP mean to developers?", *SoftwareSecured*, 2018. [Online]. Available: https://www.softwaresecured.com/what-do-sast-dast-iast-and-rasp-mean-to-developers/. [Accessed: 22-Jun- 2020].

[63]. Kok, S.H.; Abdullah, A.; Jhanjhi, N.; Supramaniam, M. Prevention of Crypto-Ransomware Using a Pre-Encryption Detection Algorithm. *Computers* 2019, *8*, 79. DOI: https://doi.org/10.3390/computers8040079.

[64]. V. N., D. Asirvatham and M. G., "Challenges to Practice Agile Methods in Global Software Development – A Review of Literature", *International Journal of Computer Applications*, vol. 179, no. 40, pp. 28-33, 2018. Available: 10.5120/ijca2018916947.