Security Integrated Scrum

Milind Daftari[1], Chetan Chaku[2], Archana J K[3]

[1]Software Engineer (Application Security and DevSecOps), J&K, India
[2]Software Engineer, J&K, India
[3]Technical Support Engineer, Karnataka, India

Abstract - *Agile is one of the most widely used and rapidly growing methodologies of software development. It resolves many of the limitations of previously used methods such as Waterfall and Spiral. Scrum framework, which is a part of Agile, is the most widespread among growing IT organizations due to its ability to deliver results quickly. For a long time, the idea of software security has been alien to the scrum framework. Therefore, security features are implemented after the development is complete, leading to an increase in costs. Contrary to this, implementing security principles from the start in the software development life cycle improves efficiency and reduces costs. The current research and prevalent practices used to include security within the Scrum framework modify the framework considerably and take away the simplicity, which is central to Scrum. To deal with these issues, we propose a model which is minimally invasive to the concepts of the Scrum framework and instead supplements the main framework to enable it to handle product security alongside its development responsibilities. Our model relies on automated secure code reviews and organizing the detected flaws as a part of the product security flaw backlog to include security considerations within the development pipeline. Based on a survey conducted involving a medium and a large-scaled IT company, our model received an average score of seven out of ten across six different parameters.*

Key Words: Agile; Scrum; Security; Secure Code Review; Software Development; Product Flaw Backlog

1.INTRODUCTION

Agile is a set of techniques followed by a team to administer a project or plan by dividing it into various stages with continuous collaboration with customers [1]. Agile methodology has become the most commonly used practice these days because of its capability to be adaptable, scalable, fast-paced, sustainable, and simplistic. Previously, software development models such as the Waterfall and Spiral were used, but they had many limitations. A working result is not obtained in the waterfall model until the later stages of the development life cycle, and it is not well suited for complex projects. The waterfall model involves a high amount of risk and cannot handle a change in requirements after development has started and requires a lot of documentation [2]. In the spiral model, the whole process is complex and expensive and is highly dependent on risk analysis. The Agile methodology was implemented to overcome these limitations. The most commonly used types under Agile Methodology are Kanban, Scrum, and Extreme Programming.

Scrum is a lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems [6]. It promotes greater transparency, product quality and efficiency while reducing costs. Scrum was developed by Schwaber and Sutherland and is described in the Scrum Guide [6]. A team that operates based on the Scrum Framework is known as a Scrum Team and typically has less than ten people. Less number of people ensures better communication and increased productivity. “Fig.1” illustrates the Scrum Framework

1.1 Entity Types

A Scrum Team has three entity types:

i. Product Owner (PO): The Product Owner is a person who is responsible for maximizing the value generated by the scrum team. Generally, the PO defines, organizes, maintains, and prioritizes tasks and goals that need to be completed or achieved by the Scrum Team.

ii. Developers: In the Scrum Framework, the developers are members of the team who work on tasks specified in the backlogs.

iii. Scrum Master (SM): The Scrum Master supports the Scrum Team in implementing and maintaining the Scrum Framework within the team. The SM is responsible for the team’s effectiveness by improving communication and resolving impediments.

1.2 Scrum Framework Events

The Scrum Framework consists of the following events:

i. Sprint: A sprint is an event in which the developers work on the tasks assigned to them. Its length is always fixed, limited to a maximum of 4 weeks.

ii. Sprint Planning: Sprint Planning is the event in which the Scrum Team gets together and reviews the open items. The Product Owner ensures that the items selected contribute to the final product goal. The items to be worked upon in the sprint are decided in the Sprint Planning meeting.

iii. Daily Scrum: Daily Scrum is a time-boxed event to inspect the sprint's progress and happens every day (not exceeding 15 minutes). Generally, developers answer three questions in this event –

- What did they do on the last working day?
- What are they planning to do today?
- Are there any impediments?

iv. Sprint Review: Sprint Review is an event in which the outcome of the sprint is assessed as to how it maps with product goals.

v. Sprint Retrospective: Sprint Retrospective event helps the Scrum Team to analyse the work done in the previous sprint and discuss how it can be improved.

1.3 Scrum Artifacts

The Scrum Framework involves three components known as Scrum Artifacts.

i. Product Backlog: It is a list maintained by the Product Owner in agreement with the key stakeholders and contains items that will enable the Scrum Team to contribute to the Product Goal.

ii. Sprint Backlog: Before the start of every sprint, few items from the Product Backlog are moved to the sprint backlog by the Product Owner after a review by the Developers during the Sprint Planning meeting. If an item cannot be completed in the sprint, it is moved back to the Product Backlog.

iii. Increment: An increment is a piece of working functionality that creates value and helps the Scrum Team achieve the product goals. It is also considered to be the Definition of Done.

When the development of an application starts, the initial design and plan stresses on the functional requirements rather than the security considerations; hence, Scrum may fail in producing software that has good security properties [3]. This leads to increased costs, the need for extra resources who specialize in application security and also creates technical debt. For a long time, security has been seen as a blocker or at least a massive speed bump that slows down a project, in some cases bringing the project to a complete halt. To avoid this, application security must be included in the design of the application from the start [2].

Web Applications are among the most vulnerable categories of applications served via the internet and have a large attack surface [3]. The number and types of flaws identified in web applications have increased in the past 11 years [4].

The first level of security which can be implemented in any application is securing the code, which can reduce the attack surface significantly. Secure Code Review [5] has become one of the most critical parts of software development. Still, it is not inherently integrated into any methodologies, including Scrum Framework. All of the methods proposed in the past few years create significant workflow changes within the framework, disturbing its core values and structure, thereby decreasing the simplicity and speed of the whole process.

This paper proposes a model that relies on automating secure code review and seamlessly integrating it with Scrum and within the continuous integration pipeline.

The remainder of this paper is divided into six sections. The second section discusses the related work of other researchers. The third section discusses the prevalent

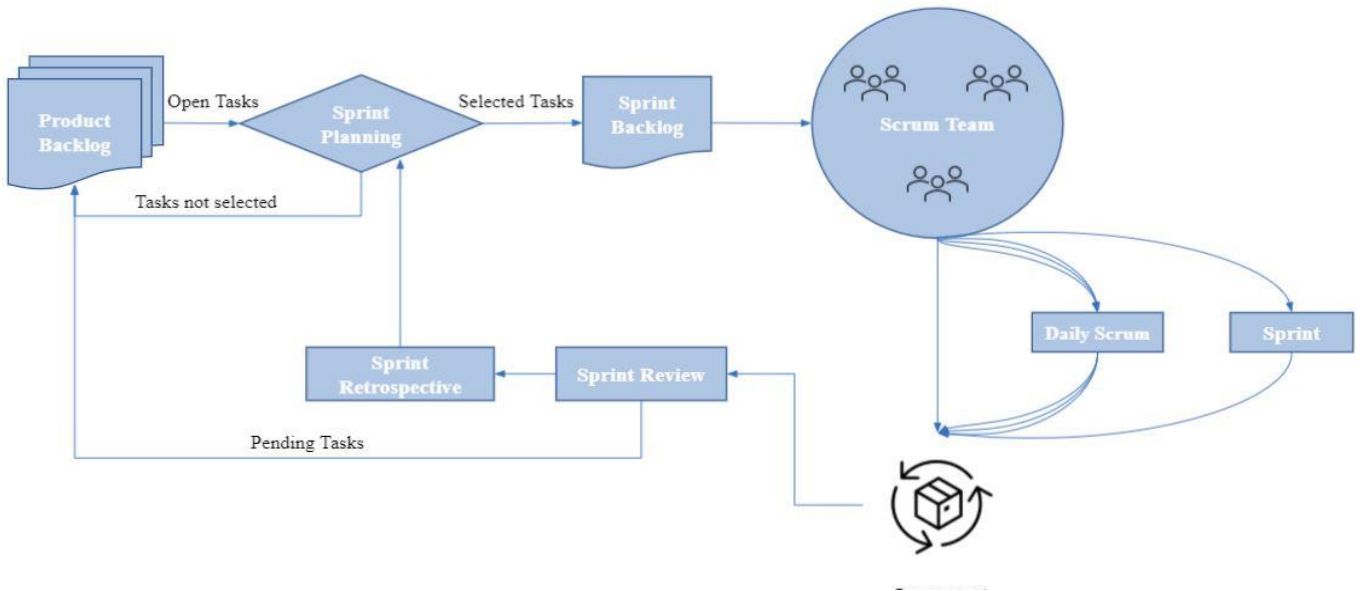

Fig. 1: SCRUM Framework

security measures. The fourth section explains our proposed model's functional and operational concepts, followed by the explanation of the technical concepts and workflows in the fifth section. The model proposed in the previous two sections is evaluated in the sixth section. The seventh section presents the conclusion of this research paper and the scope of work that can be done in the future.

2. RELATED WORK

In the whitepaper presented by Veracode [8], the authors state that the adoption of application security analysis and assessment in the Scrum Framework usually occurs in two ways. The first one involves conducting periodic sprints focusing exclusively on security. The second one consists of adding security requirements and user stories to the product backlog in every sprint. It further discusses how developers can use application security tools in Scrum teams.

N. R. Darwish and I. M. Abdelwahab [16] propose a framework to enhance the security of the software product, reduce the cost and minimize threats. The solution proposed here involves modifying the scrum framework by adding two phases before the increment is produced. The first includes 12 application penetration testing methodologies such as validation Testing and cryptography and the second includes network-level penetration tests run manually or using automated utilities. The main concerns with this approach are that it includes two extensive processes within the scrum framework that impact the delivery speed, make it complex, and require external experts.

The paper by P. Maier, Z. Ma, and R. Bloem [3] starts by discussing the current industry practices and the issues with these current practices. The paper proposes a Secure Scrum process for Web Application Development which includes an agile risk analysis method that balances the agility and effectiveness of security analysis in agile development.

S. Harrison et al. [2] focus on the transition from the traditional waterfall methodology to the modern approach, which is the Scrum framework. The paper starts by discussing the waterfall development model and its issues. It then discusses the Scrum framework, defining the framework and its essential components and security limitations. As a solution, the authors propose the usage of the OWASP Application Security Verification Standard.

A. Jøsang, M. Ødegaard, and E. Oftedal [17] highlight the importance of lack of security education in IT training programs since it is a significant contributor to the introduction of security vulnerabilities in the code written by the developers of the Scrum Teams. Thus, to understand the different software development process models, introducing cybersecurity training as part of the curriculum is an important step.

C. Pohl and H.-J. Hof [10] address the current issues in security concerning scrum and discusses the need for a secure scrum. The authors then propose and explain a solution for implementing Secure Scrum. They evaluate their approach based on specific parameters using a questionnaire and conclude by stating that their approach is easy to understand and implement. However, we can ascertain that it modifies the main scrum framework by introducing many other components, thereby increasing the Scrum framework's overall complexity.

The work of D. S. Cruzes, M. Felderer, T. D. Oyetoyan, M. Gander, and I. Pekaric [7] investigates how security testing is done in Agile Teams using a cross-case analysis approach of four teams, two teams from two different locations. They observed that the lack of knowledge on security by agile teams in general, the large dependency on incidental penetration testers and the ignorance in static testing for security are indicators that security testing is under-addressed, and that more efforts should be directed to security testing in agile teams.

Z. Azham, I. Ghani, and N. Ithnin [9] bring to light the fact that to cope with the requirement change phenomenon and deliver the product faster, the developers are applying new software development methodologies, moving away from the use of the conventional software development cycle to adopting the agile development method. The authors then propose integrating security principles in development phases using scrum and suggest the element of security backlog that can be used for security features analysis and implementation in scrum phases and how a security expert known as the Security Master supports the process.

3. PREVALENT SECURITY MEASURES

Based on existing research and professional experience, we can divide the prevalent methods of inculcating security in scrum into three most common types –

3.1. *Separate Team to handle Application Security*: In this type [7], a company creates a separate team of application security engineers to assess and monitor the state of Application Security of the whole product, spanning across all Scrum Teams. The team detects and triages issues, finds the solutions, provides fixes, and tests the secured increment. All this is done after all the different Scrum teams have completed development and integrated their increments with the whole product, and the package has passed later stages of the development and deployment pipeline.

Although acting as an aggregator of solutions, this approach goes against the teachings of the scrum guide.

The Scrum Guide [6] states the Scrum Team is responsible for all product-related activities from stakeholder collaboration, verification, maintenance, operation, experimentation, research and development, and anything else that might be required. They are structured and empowered by the organization to manage their work.

Another major issue with this approach is that the Application Security team is not always aware of each component's in-depth functionality [7]. Therefore, while doing their work, they can induce functional defects, which can be harmful to the organization as this step is completed in the later stages of the pipeline.

3.2. *Separate Security Developer/Expert within each SCRUM Team*: In this type, each scrum team consists of developers and a separate individual, known as security developer or security expert, who assesses, triages, fixes, and monitors all aspects associated with the application security and quality of the increment produced by the scrum team at the end of each sprint [7][8][9]. It is similar to an approach suggested by Veracode called the Every-Sprint Approach [8], where the security user stories and fixes are included in every sprint.

This approach induces the cost of an additional security expert to each SCRUM team. In larger organizations, the number of scrum teams can be high, so adding a security expert to each team would increase the project's overall cost and create redundancy in the processes. The time required to implement security user stories and fixes would increase as one person has to handle multiple items resulting from the work of the developers of the scrum team.

3.3. *Developers within the SCRUM Team*: In this approach, the developers within the scrum team handle security [7][10]. This approach is the least disruptive to scrum. Even so, there are few issues identified in this approach. The developers will not have the expertise to implement security user stories or fix the issues found during secure code review. The cost of training each developer in application security is high as external consultants and trainers are expensive. Every developer will have his own way of implementing or fixing flaws, which will make it challenging to establish Standard Operating Procedures and best practices. In case of security flaws in the application's design or the code, sometimes the feedback is very late to the developer, thereby causing a delay in detection and response. There is no overall governance and monitoring when working with multiple Scrum teams merging their increments to the main product at the end of each sprint.

4. PROPOSED MODEL

To address the issues affecting these approaches and to increase efficiency and quality of the software development process, we introduce a methodology, Security Integrated Scrum (SIS), where the developers in the Scrum Team are solely responsible for the security of the part of the application they handle, and all such Scrum Teams in the organization share an Application Security, Monitoring and Governance Team (ASMG Team). ASMG Team consists of developers with expertise in application security and is managed by a security program manager. It can follow either the scrum framework or Kanban. "Fig. 2" identified the role of the ASMG team in the organization.

1. ASMG Responsibilities: We recommend that the ASMG Team should take up the following responsibilities.

a) *Develop standard operating procedures:* The ASMG team should develop standard operating procedures, guidelines, and best practices, which all developers in the Scrum Teams can follow while implementing security user stories or resolving existing security flaws. This will establish standards throughout the product and help in creating easily implementable security designs and concepts. Although the developers of the Scrum Teams will fix issues related to the component of the product handled by them, they can request support from the ASMG Team whenever they need help in triaging & fixing identified flaws, implementing new stories, or internal training. This will allow the developers to keep control within the Scrum Team and be responsible for all product-related activities.

b) *Accountability for the product's security*: The ASMG team is accountable for the security of the entire product where the individual scrum teams merge their increments. They continuously monitor the state of application security and notify the relevant scrum team whenever there are any issues related to the component handled by the respective scrum team. The ASMG Team can hold the Scrum Team accountable if it observes that their component compromises the product's security and recommends improvements to the Product Owner of the Scrum Team.

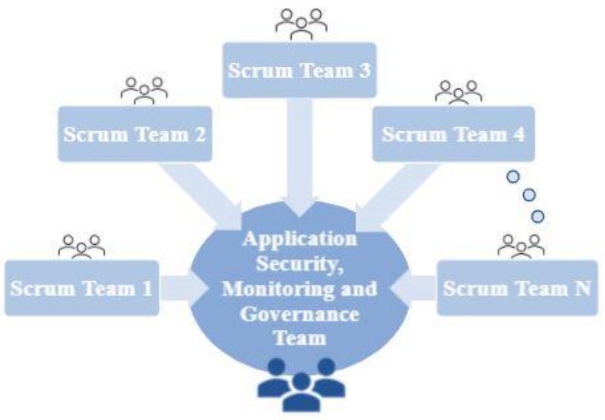

Fig. 2: Position of ASMG Team in the Organization

2. Benefits of ASMG: The benefits of having an Application Security, Monitoring and Governance Team (ASMG) are as follows:

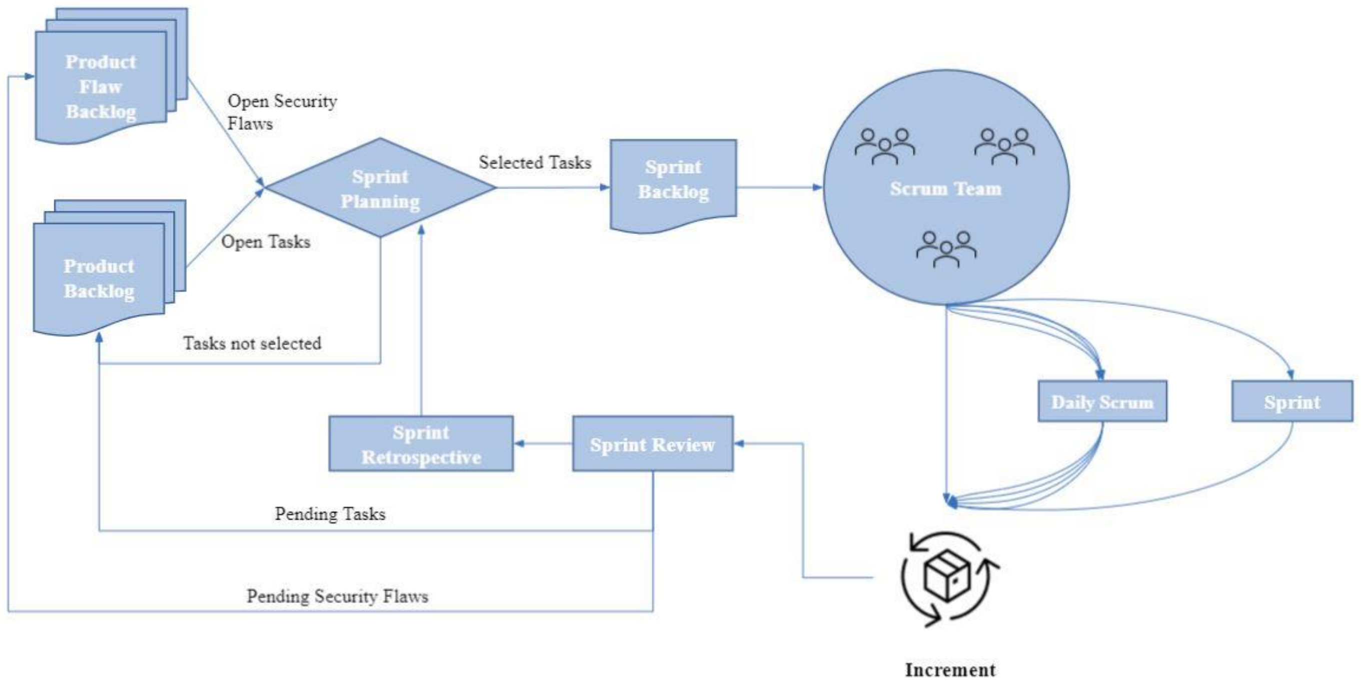

Fig. 3: The Security Integrated SCRUM (SIS) Model

- Developers in the Scrum Team always have support available whenever they face any issues related to implementing stories or security fixes.
- The overall cost is reduced as ASMG acts as a shared resource among all the Scrum Teams.
- Continuous monitoring and governance are in place.
- They provide a single source of information for the developers to implement security user stories and fix security flaws, thereby maintaining consistency throughout the codebase.
- They set up the security standards and guidelines which are to be followed by all the scrum teams.

5. SYSTEM DESCRIPTION AND IMPLEMENTATION

We begin by introducing a Product Flaw Backlog [9] which is similar to the Product Backlog of the Scrum Framework "Fig. 3". The only difference between the two is that the Product Flaw Backlog contains Security User Stories and all existing security flaws. In contrast, the Product Backlog mainly contains product enhancements, new functional stories and bug fixes.

It is minimally invasive to the Scrum Framework and holds the Product Owner responsible for prioritizing tasks from the Product Backlog and Product Flaw Backlog to reach the product goal. We recommend that prioritization for each sprint should be a healthy mix of items from both the backlogs, i.e., around 75% from product backlog and 25% from product flaw backlog.

To include Secure Code Review [5] in a Scrum environment and integrate it into the pipeline, we can use automated tools such as Veracode [11], SonarQube [12], or Synopsys [13] and many other tools available in the market. We propose a workflow to integrate Secure Code Review

The workflow mainly consists of two cycles:

A. Commit Code Analysis Cycle (CCAC)

This cycle works on a commit-by-commit basis. A developer can exit the cycle only if no legitimate open flaws are found in the commits or a manual override is done due to False Positives. CCAC is to be completed within the sprint and the team overseeing it is the Scrum team itself. Following are the steps involved in CCAC.

1) Developer commits the code in the repository: This is the first step in CCAC. A developer commits the code in the repository and the code is forwarded to the next step for analysis.

2) Secure Code Review of the Commit: In this step, the commit is passed through an automated scanning utility such as Veracode , Sonarqube or Synopsys and the results are sent to the developer who committed the piece of code. If the committed code has any security flaw, the developer moves to the next step in the cycle, else if the code has no flaws, the developer comes out of the cycle and the committed code moves to the next phase of the pipeline.

3) Triage and Solution Approach: In this step, the developer triages the flaws found in the previous step and

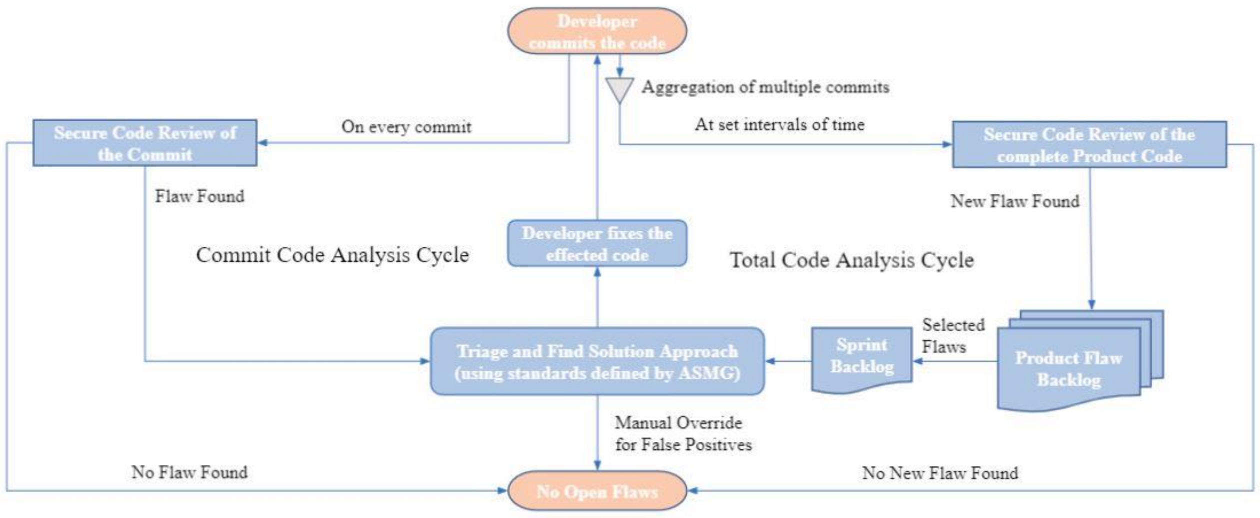
Developer commits the code
Aggregation of multiple commits
On every commit
At set intervals of time
Secure Code Review of the Commit
Secure Code Review of the complete Product Code
Flaw Found
New Flaw Found
Commit Code Analysis Cycle
Developer fixes the effected code
Total Code Analysis Cycle
Selected Flaws
Sprint Backlog
Product Flaw Backlog
Triage and Find Solution Approach (using standards defined by ASMG)
Manual Override for False Positives
No Flaw Found
No New Flaw Found
No Open Flaws

Fig. 4: SIS Model Workflow

identifies the solution approach. We recommend that the developer uses solution approaches set up by the Application Security, Monitoring and Governance (ASMG) Team to ensure that standardized fixes are implemented, and quality and consistency are maintained. If the flaws identified are false positives, the developer can break the cycle and move to the next phase of the pipeline. If the flaw is legitimate, the committed code moves to the next step of fixing the code.

4) Fixing the Legitimate Flaws: In this step, the developer fixes the flaws and commits the corrected code. As soon as the code is committed, it enters the cycle again and stays in the cycle until no legitimate open flaws are present or there is a manual override due to false positives.

B. Total Code Analysis cycle (TCAC)

This cycle scans the complete product code, committed by all Scrum Teams. All flaws identified are sent to the Product Flaw Backlog. TCAC can span multiple sprints and the team overseeing it is the ASMG Team. This is a continuous cycle and will always contain some flaws in the backlog because security is a continuous process [14] and cannot be considered complete. Following are the steps involved in TCAC.

1) Developers commit the code in the repository: Multiple developers commit the code in the repository and merge it into the codebase. Multiple such commits are aggregated, and the code is forwarded to the next step for analysis.

2) Secure Code Review of the complete Product Code: In this step, the scan of the complete product code will be triggered at set intervals of time through an automated scanning utility such as Veracode, SonarQube, or Synopsys. The ASMG team will define the scan interval after discussion with management and other stakeholders. If new flaws are identified, they are automatically moved to the Product Flaw Backlog of the Scrum Team responsible for the area where the flaw was detected; else, if the code has no new flaws, the control comes out of the cycle.

3) Product Flaw Backlog: In this step, the product owner maintains a prioritized list of all open security flaws assigned by the automation utilities related to the component handled by their Scrum team.

4) Sprint Backlog: In this step, the product owner selects items from both the Product Backlog and the Product Security Backlog and adds the final list to the Sprint Backlog after discussing with the Developers of the Scrum Team. This generally happens during the Sprint Planning meeting.

5) Triage and Solution Approach: In this step, the developer triages the flaws found in the previous step and identifies the solution approach. We recommend that the developer uses solution approaches set up by the Application Security, Monitoring and Governance (ASMG) Team to ensure that standardized fixes are given, and quality and consistency are maintained. If the flaws identified are false positives, the developer can break the cycle and move to the next phase of the pipeline. If the flaw is legitimate, the product code moves to the next step of fixing the code.

6) Fixing the Legitimate Flaws: In this step, the developer fixes the flaws and commits the corrected code. As soon as the code is committed, it again enters CCAC and TCAC cycles. In the TCAC cycle, if the flaws are resolved, it exits the cycle; else, if not resolved, the flaws are added back to the product backlog for future resolution.

6. EVALUATION

To evaluate our model, we conducted a survey involving two software development companies with five employees from each company as representatives of a scrum team with different roles. The companies comprised of a large-scale company with more than 5000 employees and a medium-scale company with around 500 employees.

Following are the types of employees from each company-

- 2 Developers
- 1 Scrum Master
- 1 Technical Manager
- 1 Application Security Engineer

We presented our detailed model and the implementation approach to them and asked them to rate it on specific parameters on a scale of 1 to 10 (1 being the lowest, 10 being the highest). The parameters were:

- Scalability
- Adaptability
- Sustainability
- Speed
- Cost-Effectiveness
- Efficiency

The table contains the average of the ratings given by the five employees from each company for each parameter.

Table -1: Model Evaluation

Evaluation Parameter	**Rating**[a]	
	Company A (Medium-scale)	**Company B (Large-scale)**
Scalability	7	6
Adaptability	8	7
Sustainability	7	8
Speed	6	7
Cost-Effectiveness	8	6
Efficiency	7	7
Average Overall Rating	*7.17*	*6.84*
Aggregated Total Rating		7

[a] Ratings assigned out of 10 for each parameter

From the evaluation of the model by experienced professionals, we can gain an insight into the possible performance benefits of implementing a security integrated scrum approach. The evident one being a formalized implementation of security integration with minimal areas of conflict and clear demarcation of responsibilities within the organization. This would also lead to a variety of less evident but equally important benefits in terms of increasing the scalability, sustainability, and cost effectiveness of the development.

The results clearly depict that the employees at all levels in the development team feel that this approach would be a viable alternative to their current security integration approach or in many cases a great addition to their development methods that lack an organized approach to security integration.

We can also decipher the variations in the degree of benefits one can hope to achieve by implementing the security integrated scrum approach in different sizes of the organization by looking at the curated results.

Scalability for a medium-scale organization is much easier to achieve as compared to a large-scale organization because of the inherent disparity in demand in the two scenarios, however, our approach facilitates scalability by decoupling the actual implementation of security measures by the developers in scrum teams from the ASGM team's responsibility of coordinating and managing standard operating procedures for security flaws across the organization.

Adaptability across a medium-scale organization will always be easier as compared to a large-scale organization due to the inertia in the members of the organization to stick with their comfortable working procedures. This can however be improved by providing clear and effective roles and responsibilities to every member involved with the security integrated scrum approach which we have done in this paper.

In terms of sustainability, we can see according to the results that there is much more scope of benefit in large-scale organizations than medium-scale ones. This can be attributed to the fact that, in medium-scale organizations the development process can be managed using ad-hoc methods and run-time implementations comparatively easily as compared to the large-scale organizations due to overall lesser scale of the project and fewer people involved. However, large-scale organizations cannot afford this kind of ad-hoc approach due to the sheer scale of the project and this is where a well-organized method like the security integrated scrum, with clear procedures of operations shines.

Speed of implementation is an area of concern among the employees as seen from the results, which is a genuine concern given the coordination required between the scrum teams and the ASMG team but we have to realize that this would be one time setup requirement since once the ASMG team is setup and functioning, the developers need only use the standard operating procedures defined by the ASMG team to complete the development process and any roadblock would only occur in one-off cases mismatch in the current security situation and no corresponding SOP for that.

Cost-Effectiveness is a major area where the security integrated scrum is very promising. We can see from the results that for small and medium-scale organizations, the implementation of an ASMG team can reduce the requirement for security experts manifold which would lead to a major boost in cost-effectiveness. For large-scale organizations, even though due to the size of ASMG team there would still be significant costs but overall due to the streamlining of the process, there would be significant improvement in development costs.

In terms of efficiency, we can clearly see the consensus across the board, be it medium or large-scale organizations, that the presented implementation of security integrated scrum would help in improving the efficiency of the inclusion of application security as part of the development process using scrum methodology.

7. CONCLUSION AND FUTURE WORK

The importance of security in the software development life cycle is increasing every day, and the threats are becoming increasingly advanced technically. As more people are becoming aware of the risks of ignoring software security, considerable research is being done towards inculcating security in the commonly used development frameworks such as the Scrum Framework. This paper identifies some of the major concerns with the current practices used to consider security during the development process. We propose a model to deal with these issues and use the product flaw backlog and automated secure code review in the pipeline to provide an organized and standardized workflow. This workflow helps implement fixes for security flaws across the organization while maintaining the sanctity of the base scrum framework as much as possible.

To further strengthen the evaluation of our model, we could organize a more wide-scaled survey including practical implementation of the model to compare and contrast the model's working against the currently prevalent models. This would lead to a high level of trust in the system's practicability, which would help organizations implement the model on a large scale.

Apart from this, we are also working to increase the model's scope to include other aspects of security like network security, threat analysis, vulnerability management, and penetration testing. This would lead to a more robust and complete security model while maintaining the basic principles of the scrum framework.

REFERENCES

[1] "5 Important Types of Agile Methodology (2021)," Jigsawacademy.com.[Online].Available:https://www.jigsawacademy.com/blogs/product-management/types-of-agile-methodology/.

[2] S. Harrison, A. Tzounis, L. Maglaras, F. Siewe, R. Smith, and H. Janicke, "A security evaluation framework for U.k. e-government services agile software development," Int. j. netw. secur. appl., vol. 8, no. 2, pp. 51–69, 2016.

[3] P. Maier, Z. Ma, and R. Bloem, "Towards a secure SCRUM process for agile web application development," in Proceedings of the 12th International Conference on Availability, Reliability and Security - ARES '17, 2017.

[4] "State of Software Security v11," Veracode.com. [Online]. Available: https://www.veracode.com/state-of-software-security-report.

[5] The MITRE Corporation, "Secure Code Review," in Systems Engineering Guide, The MITRE Corporation, Ed. MITRE Corporate Communications and Public Affairs, 2014, pp. 192–196.

[6] K. Schwaber and J. Sutherland, "The Scrum Guide," Nov. 2020.

[7] D. S. Cruzes, M. Felderer, T. D. Oyetoyan, M. Gander, and I. Pekaric, "How is security testing done in agile teams? A cross-case analysis of four software teams," in Lecture Notes in Business Information Processing, Cham: Springer International Publishing, 2017, pp. 201–216.

[8] S. Simplified, "Successful application security testing for agile development," Veracode.com. [Online]. Available: https://www.veracode.com/sites/default/files/Resources/Whitepapers/whitepaper-agilesecurity.pdf.

[9] Z. Azham, I. Ghani, and N. Ithnin, "Security backlog in Scrum security practices," in 2011 Malaysian Conference in Software Engineering, 2011, pp. 414–417.

[10] C. Pohl and H.-J. Hof, "Secure Scrum: Development of secure software with Scrum," arXiv:1507.02992 [cs.CR], 2015.

[11] "Static Analysis (SAST)," Veracode.com. [Online]. Available: https://www.veracode.com/products/binary-static-analysis-sast.

[12] "SAST Testing," Sonarqube.org. [Online]. Available: https://www.sonarqube.org/features/security/.

[13] "Coverity SAST Software," Synopsys.com. [Online]. Available: https://www.synopsys.com/software-integrity/security-testing/static-analysis-sast.html.

[14] N. Hyvärinen, "Cyber security is a continuous process," F-secure.com, 29-May-2017. [Online]. Available: https://blog.f-secure.com/cyber-security-is-a-continuous-process/.

[15] The OWASP Foundation, "Application Security Verification Standard 4.0.2," Oct. 2020.

[16] N. R. Darwish and I. M. Abdelwahab, "A security testing framework for scrum based projects," International Journal of Computer Applications, vol. 138, no. 7, pp. 12–17, 2016.

[17] A. Jøsang, M. Ødegaard, and E. Oftedal, "Cybersecurity through secure software development," in Information Security Education Across the Curriculum, Cham: Springer International Publishing, 2015, pp. 53–63.