International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 06 | June 2022 www.irjet.net p-ISSN: 2395-0072

Comparitive Analysis of Secure SDLC Models

Sarthak Tirpude[1]

[1]*Student, Department of Software Engineering, Delhi Technological University, Delhi, India*

Abstract - *Most firms have a well-oiled machine in place for developing, delivering, and maintaining functional software. However, when it comes to safeguarding that programme, not so much. Many development teams still see security as an impediment, something that creates roadblocks and compels them to redo work, preventing them from bringing innovative new things to market. However, unsecure software puts firms at risk. Cool new features will not protect you or your consumers if your product contains exploitable flaws that hackers can exploit. Instead, your team must incorporate security across the software development life cycle (SDLC) so that it facilitates, rather than hinders, the delivery of high-quality, highly secure products to market. The aim of this paper is to evaluate these three security approaches by discussing their common process, strength, and limitations. Moreover, we discuss why these approaches do not effectively increase software security, and then we set the stage for a future framework to enhance and simplify the application of various security activities within the SDLC. The structure of this paper is as follows. First, briefly describing the Touchpoints, CLASP, and SDL approaches respectively, then the comparison between the three approaches based on a set of criteria. Followed by critical evaluation of the three approaches, I discuss activities that are shared among the approaches, then I illustrate the strength and the weakness for each one, comparing more models at the end.*

Key Words: Secure SDLC

1.INTRODUCTION

What is the secure SDLC ?

A software development life cycle (SDLC) is a framework for the development of an application from conception to decommissioning. Various SDLC models have arisen over the years, ranging from waterfall and iterative to, more recently, agile and CI/CD, which increase the speed and frequency of deployment.

SDLCs typically involve the following phases:

Preparation and prerequisites
Design and architecture
Coding and test planning
Testing and outcomes
Maintenance and release

Previously, organisations would only execute security-related tasks as part of testing—at the end of the SDLC. Because of this late-game strategy, they would not discover faults, weaknesses, and other vulnerabilities until they were significantly more expensive and time-consuming to address. Worse, they would not discover any security flaws at all.

According to IBM's Systems Sciences Institute, fixing a fault discovered after implementation costs six times as much as fixing one discovered during design. Furthermore, according to IBM, the cost of resolving flaws discovered during the testing phase might be 15 times that of fixing those discovered during the design phase.

So it's significantly better, not to mention faster and less expensive, to incorporate security testing throughout the SDLC, rather than just at the end, to help uncover and mitigate vulnerabilities early on, effectively building security in. Architecture analysis during design, code review during coding and build, and penetration testing prior to release are all examples of security assurance activities.

The following are some of the major benefits of a secure SDLC approach:

Because security is a constant concern, your programme is more secure.

Security concerns are known to all stakeholders.

Design defects are detected early, before they are coded into reality. You save money because of early defect detection and resolution. You reduce your organization's overall underlying business risks.

When software began to play more active roles in numerous businesses in the last century, software security challenges arose. However, the software development process did not follow a mature engineering approach and did not include an explicit security step.

Even though the software development process has embraced a systematic engineering method such as Software Development Life Cycle (SDLC), the number of security vulnerabilities has increased over time, particularly with the broad adoption of new applications such as social network platforms.

Many studies have identified the root cause of the rise in security issues; one of the most prominent reasons is that security is not considered at any level of the SDLC. Researchers have proposed solutions to this problem since the 1980s and continue to do so now. Integrating security into each phase of the development process is one of the greatest recommendations that have gained the most attention from diverse parties. This concept has been offered by researchers for the past 30 years.

Despite the fact that various security approaches have appeared to tackle security concerns and integrate security with SDLC for a long time, the number of security challenges continues to grow. The source of rising security concerns is that software development firms continue to ignore security in their SDLC, owing to problems in implementing one of these methodologies, such as time and cost.

The following are the most commonly recommended techniques for integrating security with SDLC: the Open Web Application Security Project (OWASP) organization's Comprehensive Lightweight Application Security Process (CLASP), McGraw's Touchpoints, and Microsoft's Security Development Lifecycle (SDL).

1.1 McGraw's Touchpoints:

McGraw proposed the Touchpoints approach as a collection of security measures to be used during the software development process. They consist of both harmful and positive acts. Destructive activities include attacks, exploits, and breaking software; these are referred to as black hat (offence) activities. Constructive activities include design, defence, evaluation, and functionality, and are referred to as white hat (defence) activities. Both sorts of activities (offensive and defensive) are required, and they include:

- 1) Code review.
- 2) Architectural risk analysis.
- Penetration testing.
- 4) Risk-based security tests.
- 5) Abuse cases.
- 6) Security requirements.
- 7) Security operations.

1.2 OWASP's CLASP

Since 2006, the OWASP organisation has promoted the CLASP strategy, describing it as a well-organized and systematic approach for incorporating security issues into the early stages of the SDLC. It is a prescriptive strategy that includes paperwork and tools to help in implementation. It is an activity driven by a set of processes that are explicitly specified as the best practises for integrating security into existing or new SDLCs. The CLASP was built by OWASP to be adaptable and

simple to integrate into existing development processes. It is made up of three major parts: CLASP Views, CLASP Resources, and Vulnerability Use Cases.

The main characteristic that distinguishes the approach is its activities that are divided into separated subprocesses and assign to specific project role (project manager, security auditor, developer, architect, tester, and others). Also, it contains a large vulnerability lexicon which is structured from many perspectives that help the development team in each phase.

Criteria / Approaches	Touchpoints	CLASP	SDL
Number of the Activities	7 Activities	24 Activities	16 Activities and Optional Activities
Activities Dependence	Independent	Independent	Dependent
Nature of the Approach	Lightweight	Lightweight	Heavyweight
Organization Size Suitability	Small and Large	Small and Large	Large
Activities Type	Destructive and Constructive	Constructive	Constructive
Educating and Training	Does not Contain	Contains Separated Activity	Contains Separated Activity
Adaptation	Easy	Easy	Difficult
Activities Distribution	Link to Artifacts	Link to Roles	Link to SDLC Phases
Security Testing Type	Black and White Boxes	White Box	Black Box
Guidance	Very Weak Resources	Rich Resources	Rich Resources
Privacy	Does not Contain	Does not Contain	Contains Separated Activity
Security Policy	Does not Contain	Contains Separated Activity	Does not Contain

1.3.**Microsoft's SDL:**

SDL was created by Microsoft and is utilised internally inside the Windows Vista project. They defined it as a set of necessary security actions that must be carried out in a specified order. However, Microsoft has demonstrated that security activities carried out as part of a software development process achieve higher security objectives than actions carried out piecemeal or ad hoc based on practical experience.
Microsoft SDL includes 16 actions, the majority of which include resources such as reference, training materials (presentations, films, or webcasts), templates or examples that can be downloaded, and tools that can be used to implement the activities.

2.Comparing Touchpoints, SDL and CLASP:

2.1 Number of the Activities

Each technique has a varied amount of activities; the Touchpoints approach has the fewest, with only seven. The CLASP includes 24 activities; the security advisor may remove some of them based on their assessment. The SDL includes 16 core activities as well as optional activities that can be used in critical applications to boost security.

2.2 Activities Dependence

Because activities are dependent, the selection process and order of execution are left open and flexible. CLASP and Touchpoints activities are distinct. The SDL, on the other hand, is dependent on events that are properly organised and staged.

2.3 Nature of the Approach (Heavyweight and Lightweight) :

The Nature of the Method denotes the approach's features, such as the required team size, the flexibility of the approach to be utilised in any SDLC process, the time and money required to implement it, and activity dependency. The SDL is deemed heavyweight and severe. Touchpoints and CLASP, on the other hand, are lighter and more adaptable.

2.4 Organization Size Suitability

Because it is heavyweight and rigid, it is assumed that the SDL is appropriate for large organisations. Touchpoints and CLASP are suitable for both small and large companies due to their lightweight and adaptability.

Approaches	Strengths	Limitations
Touchpoints	• Contains constructive and destructive activities. • Lightweight. • Seven clear activities. • Easily adaptive.	• Missing education activity. • Missing guidance and resources.
CLASP	• Independent activities. • Role-based activities. • Easily adaptive. • Contains rich guidance or resource.	• Missing destructive activities. • Requires a well-experienced or trained security team.
SDL	• Providing tools and resources. • Contains very important unique activities.	• Heavyweight. • Difficult to adapt. • Missing destructive activities. • Missing activities in the operational phase.

2.5Activities Type (Destructive and Constructive)

There are two types of activities: constructive (defence) and destructive (attack) (offense). The Touchpoints encompass both constructive and destructive actions; it includes Penetration Testing, which is destructive, and Code Reviews, which is constructive. The SDL and CLASP, on the other hand, only cover constructive tasks such as security requirements, risk assessments, and code reviews .

2.6 Educating and Training

Because there are a very limited number of developers with relevant security knowledge, educating or training a team prior to the commencement of a project is a highly important job. This activity guarantees that other security operations are completed with high quality. The SDL and CLASP methods value this activity highly; it provides team members with a security awareness programme to help them gain the necessary security knowledge . The SDL only gives the security awareness programme to developers, and it contains a means to assess the developers' security understanding after the programme. The CLASP provides a security awareness programme to the entire development team, which includes people with varied jobs. The most significant downside of the Touchpoints technique is the lack of training activity.

2.7 Adaptation

Adaptation, or the flexibility of the technique to be used in any software development process, is a critical requirement. Because Microsoft created it for their internal development process, it is heavyweight, and their operations are dependant, the SDL is the most difficult to modify in any SDLC and may not adapt. Touchpoints is the simplest because it is lightweight and contains a few independent tasks. Furthermore, because the CLASP is a series of autonomous activities with rich resources that link to the project responsibilities, it is simple to adapt in any SDLC process.

2.8 Activities Distribution

The SDL activities are divided into SDLC phases (education and awareness, project inception, analysis, design, implementation, testing and verication, and deployment and support). The CLASP activities are assigned to SDLC phases based on the function that is in charge of the activity. The operations of the Touchpoints are applied to various software objects .

2.9 Security Testing Type

The SDL testing type emphasises black box testing, whereas the CLASP emphasises white box testing. The Touchpoints technique emphasises black box testing in penetration testing and white box testing in risk-based security tests .

2.10 Guidance

The guidance process is an important step in simplifying the method. Most activities can be guided by the CLASP and SDL's beneficial resources and papers, such as recommended tools and checklists. Touchpoints, on the other hand, has relatively limited resources and materials .

2.11 Privacy

Privacy is a critical notion in security. The SDL approach is the only one with a specific activity (Security and Privacy Risk Assessment) linked to privacy, whilst the CLASP and Touchpoints do not have any [12].

2.12 Security Procedures

It is a vital activity for critical software that creates security policies for an organisation, process, and product. CLASP is the only model that offers a separate activity for security policy (Identify global security policy). Because the SDL and Touchpoints do not have isolated activities to create a security policy, the security policy may be set in the security requirements

Nonetheless, I discovered that the notions of the following actions are shared by the three systems.

1) Security Requirements

Security Requirements activity is present in all three models, but in different ways; it is an important activity that is used in other SDLC phases, such as the design phase to examine security risks and the testing phase to check specified security requirements. It could refer to access control, privacy, law enforcement, abuse cases, anti-requirements, identifying resources and trust boundaries, and specifying the operating environment.

2) Risk Analysis/Threat Modeling

Risk Analysis (Threat Modeling) is a design phase activity that identifies the most important vulnerabilities and the most difficult defects. Unfortunately, because of some difficulties, only a few people do it correctly, i.e. most people do not understand it, do not know how to do it, are unable to do it, do not know if a tool can be used, do not know which tool is useful, and they focus solely on the architect design and eliminate or ignore the business design.

Examples of design problems that lead to a high- security risk and should be discovered in the Risk Analysis (Threat Modeling) activity are the following: error handling, sharing objects, incorrect or missing access control mechanisms, lack of auditing or logging, and time of check to time of use (TOCTOU), race conditions, ordering, and timing errors especially in multithreaded systems.

3) Static Source Code Review

In the name and details, Static Source Code Review is a shared activity between the three techniques (explicitly). Most software development firms use it exclusively since it detects the most frequent vulnerabilities without the need for professionals and in the shortest amount of time and money.

The other ways, which are Create Quality Gates/Bug Bars, Fuzz Testing, Incident Response Plan, and Release Archive, do not include important software. On the other side, it is intended for Microsoft, which makes it heavyweight and difficult to implement in many SDLCs. It also does not include any harmful action or operations in the operating phase.

OWASP Software Assurance Maturity Model

Governance, design, implementation, verification, and operations are the five business functions of SAMM. Each function has three security procedures, each of which has three stages of maturity.

OWASP BSIMM:

The Building Security In Maturity Model (BSIMM) assists enterprises in the planning, implementation, and measurement of software security projects. A BSIMM assessment provides an objective, data-driven review on which leaders wishing to improve their security postures can base resource, time, budget, and priority decisions. The yearly BSIMM report provides analysis based on hundreds of assessments from many industry verticals and acts as a significant benchmark for security professionals, academic curricula, and analysts. BSIMM also has a vibrant community where members discuss best practises and special content, as well as cooperate with security peers.

NIST SSDF

NIST's SSDF, also known as NIST 800-218, provides a foundational set of secure development principles that can be applied across the Software Development Lifecycle (SDLC). SSDF differs from BSIMM and SAMM in that it does not specify its own unique techniques, but instead draws from current secure software development guidance and sources such as BSIMM, SAMM, OWASP ASVS, and existing NIST guidelines, among others.

SSDF, like BSIMM, specifies safe software development techniques but does not specify how to implement them. This enables a dynamic and adaptable framework focused on secure software results rather than precise implementation specifics. SSDF also uses plain English, making it a useful tool for addressing secure development methods across communities of business owners/executives, software makers, and users.

ASVS:

The ASVS, which stands for Application Security Verification Standard, was created to standardise industry terminology for assessing the security of applications/products. Once everyone is on the same page with the terminology, organisations can buy software with confidence that it is compliant with a pre-defined security level; and it can be confident that it is compliant with this level because it was verified according to common / standard requirements, and if this is performed by an external vendor, because this is a well-defined standard.

The ASVS is generally a new initiative because most organisations have not yet begun implementing it, but it can be a great business driver for them because it aims to set a higher security level pan-organizationally, which will help organisations communicate in a uniform manner (across departments and divisions) and with similar external bodies.

Furthermore, the ASVS mandates Security Code Review, ranging from totally automated (basic level) to a combination of automated and manual, to assist assure software security - an essential issue that Comsec has been addressing with CODEFENDTM.

From our initial study, the SAMM appears to be a good model that may assist firms in integrating security into their development lifecycle, and it is very comparable to the Comsec in-house designed model. This is especially important for SMEs or businesses that lack the maturity to implement a large-scale and comprehensive model like the MS-SDL.

Comparitive Analysis of Other frameworks:

One significant distinction between SAMM and BSIMM is that SAMM is a prescriptive model whereas BSIMM is descriptive. As a result, SAMM specifies concrete activities and procedures that businesses can implement to improve software assurance. SAMM is an open-source framework, which means it is not proprietary and can be improved by the community.

What exactly is the ASVS?

The OWASP Application Security Verification Standard (ASVS) Project provides a foundation for verifying technical security controls in online applications, as well as a list of requirements for secure development.

The OWASP Application Security Verification Standard (ASVS) Project's major goal is to standardise the range of coverage and level of rigour available in the market when performing Web application security verification using a commercially-workable open standard. The standard establishes a foundation for evaluating application technical security measures, as well as any other technical security controls in the environment, that are used to protect against vulnerabilities such as Cross-Site Scripting (XSS) and SQL injection. This standard can be used to establish trust in the security of Web applications. The requirements were created with the following goals in mind:

- Use as a metric - Provide application developers and application owners with a yardstick with which to assess the degree of trust that can be placed in their Web applications,
- Use as guidance - Provide guidance to security control developers as to what to build into security controls in order to satisfy application security requirements, and

- Use during procurement - Provide a basis for specifying application security verification requirements in contracts.

Both the ASVS and SAMM standards are regarded somewhat less commonly integrated activities in the information security sector that have now been designated as OWASP projects.

A common origin:

The beginnings of BSIMM (Building Security In Maturity Model) and SAMM (Software Assurance Maturity Model) are similar, dating back to 2008-2009. I'm regularly asked what similarities and differences exist between the two models, so I created this comparison to assist organisations in determining which of these two models may be a better fit for their purposes.

Type of model:

BSIMM is a descriptive model:

The BSIMM is essentially a software security benchmark. The easiest approach to apply it is to compare and contrast your own initiative with statistics from other firms. The BSIMM also serves as an SSI roadmap. You can establish your own goals and objectives, then use the BSIMM to assess whether additional activities are appropriate for you.
The BSIMM's goal is to measure the activities of various types of SSIs across multiple businesses.

Because these initiatives employ different approaches and language, the BSIMM requires a framework that allows us to uniformly characterise any effort. Our software security framework (SSF) and activity descriptions provide a common vocabulary for explaining the key elements of an SSI, allowing us to compare initiatives that use different terms, operate at different scales, exist in different parts of the organisational chart, operate in different vertical markets, or produce different work products.

The BSIMM's only objective as a descriptive model is to observe and report. We like to report that we went to a neighbourhood to investigate what was going on and discovered that "there are robot vacuum cleaners in X of the Y houses we went to." The BSIMM does not state that "all houses must have robot vacuum cleaners," that "robot vacuum cleaners are the only acceptable type of vacuum cleaners," that "vacuum cleaners must be used every day," or any other value judgments. We provide straightforward observations.

SAMM is a prescriptive model:

SAMM is a prescriptive model, an open framework that is easy to implement, well-defined, and measurable. The solution details are simple enough for non-security persons to understand. It assists firms in analysing their present software security practises, developing a security programme in defined iterations, demonstrating progressive improvements in secure practises, and defining and measuring security-related activities.

SAMM was designed with flexibility in mind, so that it may be customised and adopted by small, medium, and big enterprises utilising any development methodology. It enables you to identify where your organisation stands in terms of software assurance and what steps should be taken to advance to the next level of maturity.

SAMM does not require that all organisations achieve the highest level of maturity in every category. Each business can decide the appropriate target maturity level for each Security Practice and adjust the available templates to their own needs.

The significance of OWASP SAMM is in giving a mechanism for your business to understand where it is on its road to software assurance and what is advised to proceed to the next level of maturity. OWASP SAMM does not require all businesses to attain maturity level 3 in all categories. Indeed, you decide the target maturity level for each Security Practice that is best suited to your organization's requirements.

Maturity levels:

"The BSIMM is not a standard maturity model in which a set of tasks is repeated at several depth and breadth levels—do something at level 1, do more at level 2, do better at level 3, and so on." Instead, the BSIMM is made up of a set of distinct activities, with activity levels utilised simply to differentiate the relative frequency with which the activities are observed in organisations. Level 1 denotes often observed activities, level 2 denotes less frequently observed activities, and level 3 denotes infrequently observed activities."

Each security practise in SAMM has three defined maturity levels and an implicit starting point of zero. The specifics at each level vary between practises, but they all represent

For SAMM, each of the security practices has three defined maturity levels and an implicit starting point at zero. The details for each level differ between the practices, but they generally represent:

0 – Implicit starting point representing the activities in the practice being unfulfilled

1 – Initial understanding and ad-hoc provision of security practice

2 – Increase efficiency and/or effectiveness of the security practice

3 – Comprehensive mastery of the security practice at scale

Comparing	**BSIMM**	**SAMM**
Primary categories	4 high level domains • Governance • Intelligence • SSDL Touchpoints • Deployment	5 business functions • Governance • Design • Implementation • Verification • Operations
Secondary categories	3 practices per domain Governance • Strategy & Metrics • Compliance & Policy • Training Intelligence • Attack Models • Security Features & Design • Standards & Requirements SSDL Touchpoints • Architecture Analysis • Code Review • Security Testing Deployment • Penetration Testing • Software Environment • Config Mgmnt / Vuln Mgmnt	3 security practices per business function Governance • Strategy & Metrics • Policy & Compliance • Education & Guidance Design • Threat Assessment • Security Requirements • Secure Architecture Implementation • Secure Build • Secure Deployment • Defect Management Verification • Architecture Analysis • Requirements-driven Testing • Security Testing Operations • Incident Management

		• Environment Management • Operational Management
Activities	7-12 observed and related activities per practice area	2 streams of activities per security practice that compliment and build on each other
Result comparison	BSIMM 11 has 130 contributing organizations that were interviewed and contributed to their comparison dataset	working on the SAMM Benchmark project to collect data from implementing and assessing organizations for comparison
Frequency of updates	annual	every 2-3 years
Assessment	BSIMM is proprietary to Synopsys. For an assessment you should reach out to them for cost and logistics.	SAMM is an open model and can be self-assessed or conducted by a number of different consulting organizations and individuals.

References:

- [1] F. G. Tompkins and R. S. Rice, "Integrating security activities into the software development life cycle and the software quality assurance process," Computers & Security, vol. 5, no. 3, pp. 218–242, 1986.
- [2]R. Baskerville, "Information systems security design methods: implications for information systems development," ACM Computing Surveys, vol. 25, no. 4, pp. 375–414, Jan. 1993.
- [3] M. Siponen and R. Baskerville, "A New Paradigm for Adding Security into is Development Methods," Advances in Information Security Management & Small Systems Security, pp. 99–111, 2001.
- [4] J. Danahy, "The 'phasing-in 'of security governance in the SDLC," Network Security, vol. 2008, no. 12, pp. 15–17, 2008.
- [5] A.-U.-H. Yasar, D. Preuveneers, Y. Berbers, and G. Bhatti, "Best practices for software security: An overview," 2008 IEEE International Multitopic Conference, 2008.
- [6] G. Raj, D. Singh, and A. Bansal, "Analysis for security implementation in SDLC," 2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence), 2014.
- [7] D. Geer, "Are Companies Actually Using Secure Development Life Cycles?," Computer, vol. 43, no. 6, pp. 12–16, 2010.
- [8] "OWASP CLASP Project," OWASP. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_CLASP_Project. [Accessed: 24-Oct-2016].
- [9] G. MacGraw, Software security: building security in. Addison-Wesley, 2006.
- [10] "Microsoft Security Development Lifecycle," Microsoft. [Online]. Available: https://www.microsoft.com/en-us/sdl. [Accessed: 24-Oct- 2016].

- [11] J. Gregoire, K. Buyens, B. D. Win, R. Scandariato, and W. Joosen, "On the Secure Software Development Process: CLASP and SDL Compared," Third International Workshop on Software Engineering for Secure Systems (SESS'07: ICSE Workshops 2007), 2007.

- [12] N. Sankhwar, A. Tewari, and V. Singh, "SDL, CLASP and TOUCHPOINTS: A Comparison and Alignment of CLASP with Waterfall Model". International Journal of Computer and Communication System Engineering (IJCCSE), vol. 2, no. 3, pp. 365- 376, 2015.

- [13] I. E. Rhaffari and O. Roudies, "Benchmarking SDL and CLASP lifecycle," 2014 9th International Conference on Intelligent Systems: Theories and Applications (SITA-14), 2014.

- [14] J. Alqatawna, A. Madain, A. M. Al-Zoubi, and R. Al-Sayyed, "Online Social Networks Security: Threats, Attacks, and Future Directions," in Social Media Shaping e-Publishing and Academia, N. Taha, R. Al- Sayyed, J. Alqatawna, and A. Rodan, Eds. Cham: Springer International Publishing, 2017, pp. 121–132.

- [15] J. Alqatawna, "The Challenge of Implementing Information Security Standards in Small and Medium e-Business Enterprises", Journal of Software Engineering and Applications, vol. 07, no. 10, pp. 883-890, 2014.