Open Access License Notice

REVIEW ON BUILDING SECURITY IN AS A SECURE SOFTWARE DEVELOPMENT MODEL

Kenneth Sigler

Oakland Community College, Auburn Hills, Michigan

ABSTRACT

Standards, models, frameworks and guidelines have been developed for secure software development such as Building Security In, SSE-CMM, Microsoft SDL, and OpenSAMM. Current standards and models provide guidance for particular areas such as threat modelling, risk management, secure coding, security testing, verification, patch management, configuration management etc. However, there is not a generally accepted model for a secure software development lifecycle. Building Security In provides an objective evaluation methodology to validate that a product satisfies a specified set of security requirements. In this paper Building Security In secure software development approach is examined and compared with other well-known standards and models.

1. INTRODUCTION

Software applications are increasingly ubiquitous, heterogeneous, mission-critical and vulnerable to security incidents, so that it is absolutely vital that information systems are properly ensured from the very beginning, due to the potential losses faced by organizations that put their trust in all these information systems and because it is cost-effective and also brings about more robust designs. Therefore, "…the trend is that security be among the non-functional requirements which are more seriously taken into account nowadays." [5]

The software development process has been continuing for a long time. Characteristic of the first software projects were missed schedules, blown budget, and flawed products. "Although when we looked to the past there wasn't a magic solution, a single development, in either technology or management technique, promises magnitude improvement in productivity, in reliability, in simplicity." [1] In addition to new programming languages, new programming techniques, a few standards and models have been developed to solve these problems such as CMMI and OpenSAMM.

In addition to poorly managed secure software development methodologies, exponential increase in the internet enabled applications, unsuspected internet users and hackers caused new problems. One of the most important of these problems is software vulnerabilities used by hackers and unsuspected users. Vulnerabilities are weaknesses in software that allow hackers to compromise the integrity, availability or confidentiality of processed data or software. Some of the most severe vulnerabilities allow hackers to run malicious code, potentially compromising the computer, its software, and the data that resides on the computer.

Various sources including software vendors, security software vendors, independent security researchers, and those who create malicious software can cause exposure of vulnerability. The number of software vulnerabilities since the mid 1990's is shown at figure 1. "The primary reason for the dramatic increase in the number of vulnerabilities is the widespread use of the Internet and new forms of computer applications. The other reason is the emphasis given to functionality over the underlying principles of security during the software development phases." [4] Identifying vulnerabilities in released software requires much time and work force so vulnerability

detection in the early stages of development decreases the cost of fixing them. Consequently, it is important to employ such processes throughout the lifecycle. Hence the need for a defined security process framework.

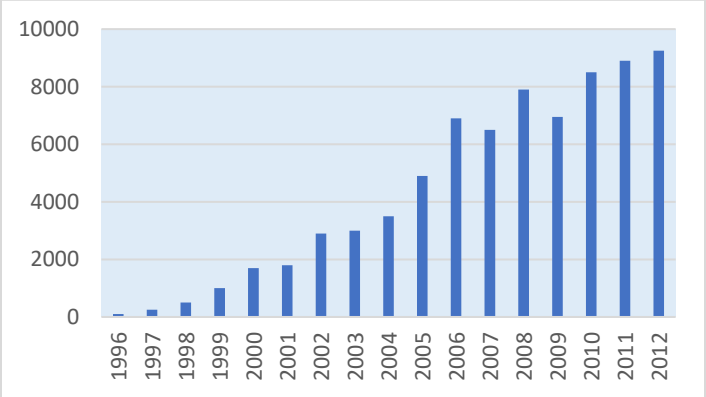

Figure 1. Number of software vulnerabilities

In the traditional software development lifecycle, security testing is often added later, and security verification and testing processes are postponed until after the software has been developed. Software vulnerabilities are an emergent property which appear during the design and implementation cycles. So a "before, during, and after" approach should be considered throughout software development.

Secure software development models, frameworks and guidelines are used for software development processes such as the System Security Engineering Capability Maturity Model (SSE-CMM), Microsoft Security Development Lifecycle (SDL), Open Software Assurance Maturity Model (OpenSAMM), and Building Security In Maturity Model (BSIMM-V). However, there is not a generally accepted model for secure software development. “Current models provide guidance for particular areas such as threat modeling, risk management, secure coding, security testing, verification, patch management, configuration management etc. It is crucial for these to be combined into an integrated and more comprehensive construction method.” [10] Several advances have recently been made in the definition of processes for secure software development. However, there has been minimal comprehensive comparison of these methodologies with general secure software requirements. Therefore, it is difficult for consumers and developers to understand their strengths and weaknesses and, hence, it is hard to make an ‘informed’ decision about which one is more appropriate for the job.

The goal of this paper is to evaluate and compare Microsoft SDL, SSE-CMM, OpenSAMM, and BSIMM-V secure software development approaches. BSIMM-V is hardware/software security process evaluation model which is used for security testing, security requirements definition and other secure system issues. “Here's what happens when you measure a new firm using the BSIMM measuring stick. You can directly compare how your software security initiative stacks up against the other 67 firms in BSIMM-V.” [5] In this paper BSIMM-V is used as secure software development guidance and compared with Microsoft SDL, SSE-CMM, and OpenSAMM.

The structure of this paper is as follows: Section 2 describes the software development models, Section 3 contains the BSIMM-V secure development approach. Section 4 explains secure software development lifecycle objectives and provides a comparison with and other models according to those objectives. Section 5 presents results.

2. SOFTWARE DEVELOPMENT MODELS

2.1 Capability Maturity Model Integration

Capability Maturity Models to provide a reference model of mature practices for a specified engineering discipline. These models were originally created by Carnegie Mellon University and continue to evolve today. Organizations can compare its practices according to the model to identify potential areas for improvement. The CMMs provide goal-level definitions for and key attributes of specific processes such as software engineering, systems engineering, security engineering, but they do not generally help operational guidance for performing the work. In other words, they do not explain processes, they explain process characteristics; they define the "what's" that should be done, but not the "how's". "CMM-based evaluations don't mention, directly, the product evaluation or system certification. They are focus process improvement efforts on problem identified areas." [3] Capability Maturity Model Integration (CMMI) purposes to increase the maturity of organizations processes to improve long-term business performance. CMMI provides the latest best practices capabilities for product life cycle. This model provides improvement in systems engineering, software engineering, integrated product and process development, supplier sourcing, and acquisition.

2.2 SSE-CMM -- Systems Security Engineering Capability Maturity Model

The intention of all best-practice models is to provide the common point of reference needed to coordinate and execute an assurance process. "The Systems Security Engineering Capability Maturity Model (SSE-CMM), also known as ISO/IEC 21827, specifies an optimum set of behaviors, which an organization can adopt to ensure secure system and software engineering practice. Like the CMM the model also provides a maturity scale that an organization can use to improve its capabilities over time." [7]

2.4 Microsoft Security Development Lifecycle

"Microsoft Security Development Lifecycle has adopted for software development that needs to withstand security attacks." [6] Microsoft's software development process includes a series of security-focused activities and deliverables to each phase of software development. Microsoft Security Development Lifecycle (SDL) was formed with the Trustworthy Computing (TwC) directive of January 2002. At that time, many software development groups at Microsoft started to find ways to improve the security of existing code.

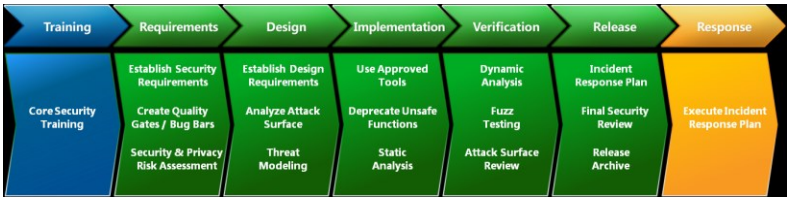

Figure 2. Simplified security development model

Microsoft SDL was designed as an integral part of the software development process at Microsoft and published as a mandatory policy in 2004. The development, implementation, and constant improvement of the SDL adopted at Microsoft to provide policy for software designed, development, and testing for security. Over time, the Microsoft SDL has matured into a well-defined methodology.

Basic steps of SDL model are shown at Figure 2. Microsoft has added a lot of new properties and capability since 2002. "Significant in the list of capabilities include: bug bar, cryptographic standards, runtime verification testing, banned API (Application Programme Interface), privacy standard for development, online service requirements, cross site scripting defences, SQL injection defences, XML parsing defences, Address space layout randomization, cross site request forgery defence, fuzzying (network), operational security reviews, third party licensing security requirements, external tool release, sample code compliance with SDL, and external tool releases." [6]

2.5 OpenSAMM Model

The Software Assurance Maturity Model (SAMM) is an open model to enable organizations formulate and implement a strategy for software security. This model attempts to solve the specific software security risks facing the organization. The resources provided by SAMM will aid in:

- Evaluating an organization's existing software security practices
- Building a balanced software security assurance program in well-defined iterations
- Demonstrating concrete improvements to a security assurance program
- Defining and measuring security-related activities throughout an organization

SAMM can be utilized by any size organization using any methodology of software development. Additionally, this model can be applied organization-wide to be utilized in all business areas, or an individual project. OpenSAMM, offers a roadmap and well-defined maturity model for secure software development and deployment. At the same time it offers some good tools for self-assessment and planning.

Each SAMM business function defines three security practices. Each security practice in turn builds assurance into the related business function. In total, there are twelve security practices that a independent silos for improvement that map beneath the business functions of software development. Governance, Construction, Verification and Deployment critical business functions make up the top level of the SAMM hierarchical model.

Governance includes processes that cross-cut groups involved in development as well as business processes that are established at the organization level. Strategy and metrics, policy and compliance, education and guidance regulate this business function at organization or project level.

Construction involves the processes and activities related to how an organization defines goals and creates software within development projects. In general, this will include security requirements, threat assessment and secure architecture.

Verification involves the processes and activities related to how an organization checks and tests errors encountered during the software development phase. The practices beneath this business function focus on design review, security testing and code review.

Deployment contains the processes and activities related to how an organization manages release of software that has been created. This can involve product delivery to end users, deploying products to internal or external hosts, and normal operations of software in the runtime environment.
The OpenSAMM model is shown in Figure 3.

"The model has a strong resemblance to CoBIT (Control Objective for Information and Related Technology), which aims to measure every security objective. In the CoBIT model, security operation maturity levels take a value from 0 to 3:

- A level of 0 means the operation is not applied.
- A level of 1 means an organization does not have a systematic approach to security but does have a basic-level application.
- A level of 2 means the operation is applied at the appropriate maturity level within the organization.
- A level of 3 means the operation is applied perfectly at the organization." [7]

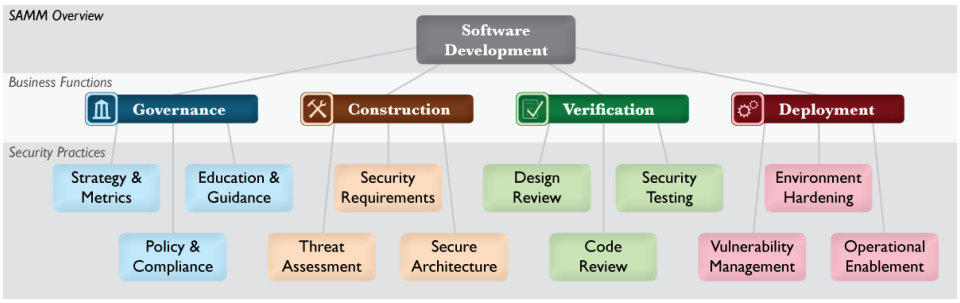

Figure 3. OpenSAMM model

3. BUILDING SECURITY IN MATURITY MODEL

"The BSIMM is a measuring stick for software security. The best way to use the BSIMM is to compare and contrast your own initiative with the data about what other organizations are doing contained in the model. You can then identify goals and objectives of your own and look to the BSIMM to determine which further activities make sense for you." [2] BSIMM is the work of three software security experts named Gary McGraw, Brian Chess, and Sammy Migues. They began by analyzing several leading software security initiatives from software vendors, technology firms, and the financial services industry. The model uses a software security framework (SSF) to organize software security tasks. This framework helps an organization determine how its own security practices compare with others and how to advance them over time.

2.1 The BDIMM Study

The BSIMM has had five major releases:

- The most recent release was published in October 2013. BSIMM-V increased it's analysis to 67 organizations measuring across 111 distinct activities.
- The fourth major release was published in September 2012. BSIMM4 included analysis of 51 organizations and a total set of 132 distinct measurements.
- BSIMM3 was published in September 2011 and included analysis of 42 organizations for a total set of 81 distinct measurements.
- BSIMM2, published in May 2010, included analysis of 30 organizations and 42 distinct measurements. Some firms included large subsidiaries that were independently measured.
- The original study, published in March 2009, included analysis of nine organizations and nine distinct measurements.

Although the names of the organizations are irrelevant to this discussion, the participants were not necessarily software developers. Most of the participants were Fortune 500 companies that depend on secure software to help them accomplish their business missions and objectives.

The work of McGraw, Chess, and Migues on the BSIMM model shows that measuring an organization's software security program is useful for planning, structuring, and executing a software security initiative. Over time, companies that participate in the BSIMM project show measurable improvement in their software security initiatives. The measurements in the study began with an in-person interview with the executive in charge of the software security program. The information from the interview was then compiled into a scorecard that identified which of the 111 BSIMM activities the organization carried out. The BSIMM does not publish scorecards for each participating organization, but BSIMM-V does provide a composite scorecard of activities performed by all of the companies studied. That scorecard is reproduced in Figure 4.

BSIMM-V Scorecard for: FIRM — Raw Score: 37

Governance			Intelligence			SSDL Touchpoints			Deployment		
Activity	BSIMM-V Firms	FIRM	Activity	BSIMM-V Firms	FIRM	Activity	BSIMM-V Firms	FIRM	Activity	BSIMM-V Firms	FIRM
[SM1.1]	44	1	[AM1.1]	21	1	[AA1.1]	56	1	[PT1.1]	62	1
[SM1.2]	34		[AM1.2]	43		[AA1.2]	35	1	[PT1.2]	51	1
[SM1.3]	34	1	[AM1.3]	30		[AA1.3]	24	1	[PT1.3]	43	
[SM1.4]	57	1	[AM1.4]	12	1	[AA1.4]	42		[PT2.2]	24	1
[SM1.6]	36		[AM1.5]	42	1	[AA2.1]	10		[PT2.3]	27	
[SM2.1]	26		[AM1.6]	16		[AA2.2]	8	1	[PT3.1]	13	1
[SM2.2]	31		[AM2.1]	7		[AA2.3]	20		[PT3.2]	8	
[SM2.3]	27		[AM2.2]	11	1	[AA3.1]	11				
[SM2.5]	20		[AM3.1]	4		[AA3.2]	4				
[SM3.1]	16		[AM3.2]	6							
[SM3.2]	6										
[CP1.1]	42	1	[SFD1.1]	54		[CR1.1]	24		[SE1.1]	34	
[CP1.2]	52		[SFD1.2]	53	1	[CR1.2]	34	1	[SE1.2]	61	1
[CP1.3]	45	1	[SFD2.1]	26		[CR1.4]	50	1	[SE2.2]	31	1
[CP2.1]	24		[SFD2.2]	29		[CR1.5]	23		[SE2.4]	25	
[CP2.2]	28		[SFD3.1]	9		[CR1.6]	25	1	[SE3.2]	10	
[CP2.3]	28		[SFD3.2]	13		[CR2.2]	10		[SE3.3]	9	
[CP2.4]	25		[SFD3.3]	9		[CR2.5]	15				
[CP2.5]	35	1				[CR2.6]	18				
[CP3.1]	14					[CR3.2]	4	1			
[CP3.2]	11					[CR3.3]	6				
[CP3.3]	8					[CR3.4]	1				
[T1.1]	50	1	[SR1.1]	48	1	[ST1.1]	51	1	[CMVM1.1]	59	1
[T1.5]	29		[SR1.2]	43		[ST1.3]	55	1	[CMVM1.2]	59	
[T1.6]	23	1	[SR1.3]	45	1	[ST2.1]	27	1	[CMVM2.1]	50	1
[T1.7]	33		[SR1.4]	27	1	[ST2.4]	13		[CMVM2.2]	44	
[T2.5]	9		[SR2.2]	23		[ST3.1]	11		[CMVM2.3]	30	
[T2.6]	13	1	[SR2.3]	19		[ST3.2]	8		[CMVM3.1]	6	
[T2.7]	9		[SR2.4]	19		[ST3.3]	6		[CMVM3.2]	6	
[T3.1]	4		[SR2.5]	22	1	[ST3.4]	5		[CMVM3.3]	2	
[T3.2]	4		[SR3.1]	8		[ST3.5]	7				
[T3.3]	8		[SR3.2]	12							
[T3.4]	9										
[T3.5]	5										

Legend:
Activity — 111 BSIMM-V activities, shown in 4 domains and 12 practices
BSIMM Firms — count of firms (out of 67) observed performing each activity
the most common activity within a practice
a common activity not observed in this assessment
a common activity observed in this assessment
a practice where firm's high-water mark score is below the BSIMM-V average

Figure 4 BSIMM-V weighed activity scorecard

The BSIMM-V data in this study provide some interesting analytical results. Figure 5 reproduces a spider chart that shows the average maturity levels in each of the 12 practices. The chart has 12 sections; each is associated with one of the 12 practices. Level 3 is considered more mature than level 0.

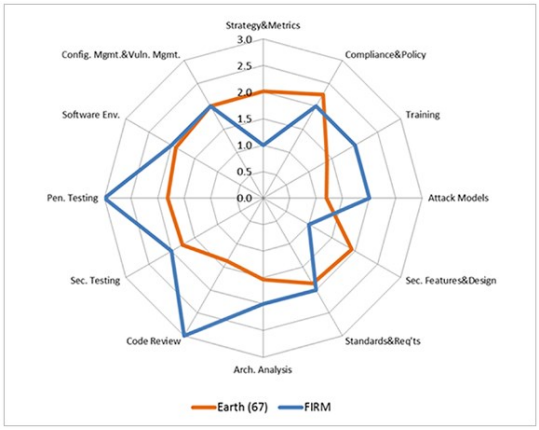

Figure 5 BSIMM-V average maturity level by practice

By studying the aggregate chart, you can see that the greatest maturity appears to fall within the Compliance & Policy practice, at a level of approximately 2.4. The least mature areas are Training, Attack Models, Architecture Analysis, and Code Review; all have maturity levels within a range of approximately 1.25 to 1.5. An organization that wants to use this model as a basis for improving its software assurance program might also be interested in reviewing similar maturity levels measured by each of the 12 industries represented in the study. For instance, a financial institution might be interested in the maturity level for Compliance & Policy because it operates within a highly regulated industry.

2.2 The BSIMM in Context

The SSF, used by the BSIMM, consists of four domains: Governance, Intelligence, Secure Software Development Lifecycle (SSDL) Touchpoints, and Deployment. Each domain has its own set of business goals and is broken down to define three practices designed to satisfy one of the business goals. The success of implementing each practice is based on completing prescribed activities within that practice. Each of the 111 BSIMM activities is associated with one of the following 12 practices:

- *Governance – Strategy and Metrics*—Transparency of expectations and accountability for results
- *Governance – Compliance and Policy*—Prescriptive guidance for all stakeholders and software development lifecycle activities that can be audited
- *Governance – Training*—Creation of a knowledgeable workforce that corrects errors in processes
- *Intelligence – Attack Models*—Creating customized knowledge about attacks relevant to the organization, building data classification schemes, and identifying likely attackers
- *Intelligence – Security Features and Design*—Creation of customized, proactive guidance and knowledge of security features, frameworks, and patterns
- *Intelligence – Standards and Requirements*—Creation of prescriptive guidance for shareholders and documentation of software security choices
- *SSDL Touchpoints – Architecture Analysis*—Detection and correction of security flaws, classifying risk, and performing design review
- *SSDL Touchpoints – Code Review*—Detection and correction of security flaws, enforcing coding standards, and using automated and manual reviews
- *SSDL Touchpoints – Security Testing*—Detection and correction of security flaws using methods like fuzz testing, enforcing adherence to standards, and the reuse of approved security features
- *Deployment – Penetration Testing*—Detection and correction of security flaws, providing sanity checks and internal/external tests
- *Deployment – Software Environment*—The ability to make authorized changes and to detect unauthorized changes and activity using processes like application behavior monitoring and diagnostics
- *Deployment – Configuration Management and Vulnerability Management*—The ability to track authorized changes to applications and to detect unauthorized changes and activities with an emphasis on incident response

Each practice is divided into three maturity levels that defines the activities that need to be addressed first and which require prioritizing. “Although the BSIMM is not a complete ‘how to’ guide for software security, it provides a plethora of ideas and general guidance. Each activity includes a stated objective, a description, and a brief example to illustrate how at least one organization accomplished its objective. For example, an activity in the training practice advises the software security group (SSG) to have an advertised lab period during which developers can drop in and discuss secure development or coding issues. This activity provides an informal resource to other departments.” [7]

The SSG is an internal group devoted to software security and 67 organizations that took part in the research for BSIMM-V agree that the success of their programs depends on having an SSG. The group should include senior executives, system architects, developers, and administrators.

4. SOFTWARE DEVELOPMENT MODELS SECURITY APPROACH

In this section, security properties of BSIMM, SSE-CMM, Microsoft SDL, and OpenSAMM models are compared according to security properties as given in Table 1. While creating this table all possible security properties were considered in secure software development lifecycle and were compared with secure software development models.

BSIMM is software security lifecycle evaluation model and it expects that the organization adopting the model, complete required criteria according to prescribed activity. In this paper BSIMM is approached from a secure software development guidance perspective. SSE-CMM recommends a model that security should be considered

at the system development level. However SEE-CMM is not directly secure software development focused model. BSIMM, Microsoft SDL and OpenSAMM models are directly focused on secure software development.

It is very important for all members of software development teams to receive appropriate training to stay informed about security basics and recent trends in security and privacy. Individuals who develop software programs should attend regularly security training. BSIMM, SSE-CMM, Microsoft SDL and OpenSAMM all check personnel education and awareness.

Physical security and logical security (network security and application security controls, access controls) are checked by SEE-CMMI. But BSIMM, Microsoft SDL and OpenSAMM do not directly examine physical and logical security.

Table 1. Comparison of secure software development standards and models

	BSIMM-V	**SSE-CMM**	**Microsoft-SDL**	**OpenSAM**
Security Training and Awareness	√	√	√	√
Physical and Logical Security	X	√	X	X
Secure Configuration Management	√	√	X	X
Law, policy and procedure compliance	√	√	X	√
Threat Modeling	√	√	√	√
Risk Analysis	√	√	√	√
Security Requirements Definition	√	√	√	√
Security Architecture	√	√	√	√
Secure Design	√	√	√	√
Source Code Analysis	√	X	√	√
Vulnerability Analysis	√	√	√	√
Security Verification	√	√	√	√
Vulnerability Management	√	√	√	√
Secure Development Techniques and Applications	√	√	√	√
Operational Environment Security	√	√	√	√
Secure Integration with Peripheral	√	√	√	√
Secure Delivery	√	√	X	√

Secure configuration management provides secure access to source code, secure control and management of software development documents. BSIMM and SEE-CMMI handle secure configuration management. But Microsoft SDL and OpenSAMM do not directly address this subject.

An organization might have a wide variety of law, policy and compliance requirements. These requirements either directly or indirectly affect the organization software or hardware products. Microsoft SDL does not directly address law, policy and procedure compliance but BSIMM, SSE-CMMI and OpenSAMM address these requirements through activities defined within each model.

Threat modelling is used to realise meaningful security risk. This activity prescribes that development teams consider, document, and discuss the security implications of designs in the operational environment and structured fashion. Threat modelling enables consideration of security issues at the asset or application level. Threat modelling is a team exercise, developers, testers, and represents the primary security analysis task performed during the software design phase. All four models carry out application specific threat modelling at the point of construction.

A risk analysis involves identifying the most probable threats to software and analyzing the related vulnerabilities of the software to these threats. All and models address this activity.

A very important part in the software development process for the achievement of secure software systems is known as security requirements which provides techniques, methods and standards for performing this task in the information system development cycle. The software development process must be repeatable and contain systematic procedures. This approach ensures that the set of requirements obtained is complete, consistent and easy to understand and analyzable by the different actors involved in the development of the system. All of four models consider that threat, assumption and organizational security process are related and checked with Information and Communication Technology (ICT) and non ICT security requirements by designer.

Secure architecture and design needs to be taken into account in one of the the very early phases of the software development lifecycle. Failure to do so, can lead to design-level security flaws. In order to ensure adequate attention in all appropriate stages of the software development lifecycle, security architecture and design is needed as a necessary part of software engineering. All of models consider the importance of security architecture and design.

Source code analysis is very important for finding vulnerabilities during secure software development. Source code analysis can be aided with the use commercial tools. But analysis results should be reviewed by experts to eliminate false positives. Source code analysis is not mandated in the SSE-CMM standard. Although, if the evaluator wants to perform source code analysis it can be done. However, BSIMM, Microsoft SDL and OpenSAMM emphasize the importance of source code analyze and encourage to use automatic tools.

Vulnerability analysis includes black box and wide box testing. Vulnerability management focuse on notification of detected security vulnerabilities and weakness to the development team, deal with vulnerability and software updates. All standards and models include activities associated with vulnerability analysis and management processes.

The purpose of security validation is the confirmation of security requirements and application design. So security reviews and tests are applied in security lifecycle management processes. Design review, unit security tests, integration security tests and security acceptance tests are sub components of security validation. All standards and models include security validation. Microsoft SDL and OpenSAMM have detailed procedures for security validation.

Secure development techniques cover definition of secure code development procedures and use of the best security practices during the development phase. Due to the process management nature of all four models, it comes as no surprise that each provides considerable coverage of this area.

The purpose of operational environment security is to develop security related guidance and provide it to system users and administrators. This operational guidance tells the users and administrators what must be done to install, configure, operate, and decommission the system in a secure manner. In addition to guidance documents this information can be given by program helps or other tools. To ensure that this is possible, the development of the operational security guidance should start early in the life cycle. All four models include operational environment security.

The concern of secure delivery is the secure transfer of the finished product from the development environment into the responsibility of the user. Delivery requirements must be defined to provide assurance during distribution of the product to the user. Other than Microsoft SDL, the other three models provide guidelines for secure delivery.

5. CONCLUSIONS

Because of poor secure software development processes and tools, resulting software products have weaknesses and vulnerabilities. These vulnerabilities and weaknesses are misused or exploited by unconscious users or attackers. Secure software development standards and models have been developed to minimize weakness and vulnerabilities. They consider management of weaknesses and vulnerabilities during design, implementation and life time of product. But none of them could provide all of the requirements of secure software development lifecycle.

Since BSIMM is based on what organizations are actually doing, it can be seen as a de facto standard. The BSIMM does not provide any real insight into which activities are commonly practiced in organizations and which are not. Also, unlike many official standards, the BSIMM accepts the notion that not all organizations need to achieve the same security goals. No organization needs to carry out all 111 activities, and the average maturity of the 67 participating organizations varies greatly, as the previously reported data demonstrates. However, the model does provide a potential benchmark against which all organizations can be measured and demonstrate progress. This benchmark has made the BSIMM quite appealing over the past several years. Within the past decade, software industry leaders have recognized that software cannot rely on haphazard defenses to protect it; security needs to be built into the product.

REFERENCES

[15] Brooks, F., P., (1987) “No Silver Bullet- Essence and Accidents of Software Engineering” IEEE Conference Vol 20, No:4, pp10-19

[16] Building Security In Maturity Model (2013) http://www.bsimm.com

[2] Carol Woody, (2009), "Secure Software Development Life Cycle Processes" Carnegie Mellon University, ID: 326-BSI, Version: 22, https://buildsecurityin.us- cert.gov/bsi/articles/knowledge/sdlc/326-BSI.pdf

[3] IBM X-Force 2010 Trend and Risk Report, (2011), http://www-935.ibm.com/services/us/iss/xforce/trendreports/

[1] McGraw G., Migues S., West J., (2013) “Software [In]security: BSIMM-V does a number on secure software dev”, http://searchsecurity.techtarget.com

[5] SDL Progress Report 2004-2010, Microsoft, (2011) http://www.microsoft.com/download/en/details.aspx?id=14107

[8] Shoemaker D., Sigler K., (2014) “CyberSecurity: Engineering a Secure Information Technology Organization”, Cengage Learning

[6] Software Assurance Maturity Model, (2009) “A Guide to Building Security into Software Development Version. 1.0”, http://www.opensamm.org

[4] System Security Engineering Capability Maturity Model, (2003) http://www.sse-cmm.org/docs/ssecmmv3final.pdf

[11] Thuraisingham B., Hamlen, K. W., (2010) “Challenges and Future Directions of Software Technology: Secure Software Development”, IEEE 34th Annual Computer Software and Applications Conference (COMPSAC), pp19-20, 19-23 July 2010, Soul, South Korea