International Journal of Computer Science & Information Technology (IJCSIT) Vol 4, No 2, April 2012

REVIEW ON COMMON CRITERIA AS A SECURE SOFTWARE DEVELOPMENT MODEL

Mehmet Kara[1]

[1]TUBITAK BILGEM UEKAE, Kocaeli, Turkey
mkara@uekae.tubitak.gov.tr

ABSTRACT

Standards, models, frameworks and guidelines have been developed for secure software development such as such as Common Criteria, SSE-CMM, Microsoft SDL, 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. But there is not a generally accepted model for a secure software development lifecycle. Common Criteria provides objective evaluation methodology to validate that a product satisfies a specified set of security requirements. In this paper Common Criteria secure software development approach is examined and compared with other well known standards and models.

KEYWORDS

Common Criteria, Secure Software Development, Vulnerability, Confidentiality, Integrity, Availability.

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, security is among the non-functional requirements which are more seriously taken into account nowadays [1].

Software development process has been continuing for a long time. Characteristic of the first software projects was missed schedules, blown budget, and flawed products. But 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 [15]. In addition to new programming languages, new programming techniques, a few standards and models have been developed to solve these problems such as CMMI, FAA-iCMM, OpenSAMM.

In addition to poor secure software development methodologies, exponential increase in the internet enabled applications, unconscious internet users and hackers caused new problems. One of the most important of these problems is software vulnerabilities used by hackers and unconscious users. Vulnerabilities are weaknesses in software that allow hackers to compromise the integrity, availability or confidentiality of processed data or that 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 disclosure of vulnerability.

DOI : 10.5121/ijcsit.2012.4207

Number of software vulnerabilities according to years is shown at figure 1. Main reason for the fast increase in the number of vulnerabilities is the widespread use of the Internet and new computer applications [3]. The other reason is the emphasis given to functionality than security during the software development phases. Fixing 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.

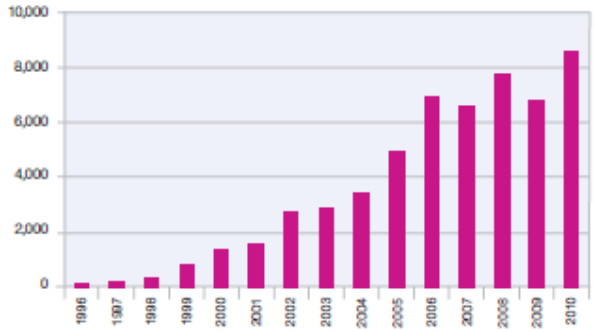

Figure 1. Number of software vulnerabilities

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, "before, during, and after" approach should be considered at software development.

Secure software development models, frameworks and guidelines are used for software development processes such as System Security Engineering Capability Maturity Model (SSE-CMM), Microsoft Security Development Lifecycle (SDL), Open Software Assurance Maturity Model (OpenSAMM), Common Criteria (CC)[13, 10]. But there is not a generally accepted model for secure software development. Current models provide guidance for particular areas such as threat modelling, 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 [11]. Several advances have recently been made in the definition of processes for secure software development. However, there has been no objective and comprehensive comparison of these methodologies with general secure software requirements [7,12,14]. 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.

When we look to CC development process TCSEC, ITSEC and CTCPEC are well known security testing and evaluation models. Trusted Computer System Evaluation Criteria (TCSEC), also called the Orange Book was developed in the United States in the early 1980’s. TCSEC sets basic requirements for evaluating the effectiveness of computer security controls built into a computer system. Main purpose of the TCSEC evaluate, classify and select computer systems which is being considered for the processing, storage and retrieval of sensitive or classified information. In the succeeding decade, various countries especially in Europe began initiatives to develop evaluation criteria that built upon the concepts of the TCSEC.

Information Technology Security Evaluation Criteria (ITSEC) is published under auspices of the European commission in 1991. The ITSEC shall be used as a guideline for evaluation and certification of the security of IT products. At the same years Canada used Trusted Computer Product Evaluation Criteria (CTCPEC) methodology for IT product security evaluation.

Different methodology usage in United States, Europe and other countries caused product certification and evaluation problems. One IT product produced in Europe sold outside the Europe its security certification was not accepted by purchaser country. The same scenario was valid for United States and other countries.

The CC 1.0 were developed through a combined effort of six countries: the United States, Canada, France, Germany, the Netherlands, and the United Kingdom. This effort built on earlier standards, including Europe's ITSEC, the United TCSEC, and the Canadian CTCPEC. The CC is an international standard (ISO/IEC15408) for IT security. A CC evaluation allows an objective evaluation to validate that a particular product satisfies a defined security requirements. The focus of the CC is evaluation of a product or system, and less on development of requirements. Nevertheless, its evaluation role makes it of interest to those who develop security requirements.

The goal of this paper is to evaluate and compare Microsoft SDL, SSE-CMMI, OpenSAMM and CC secure software development approaches. CC is hardware/software security evaluation standard Which is used for security testing, security requirements definition another secure system issues [1]. In this paper Common Criteria is used as secure software development guidance and compared with Microsoft SDL, SSE-CMMI, OpenSAMM.

The structure of this paper is as follows: Section 2 describes the software development models, Section 3 contains CC secure development approach. Section 4 explains a secure software development lifecycle objectives and compare with standards and models according to these 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 created by Carnegie Mellon University. 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 don't explain processes, they explain process characteristics; they define the what should be done, but not the how should be done. "CMM-based evaluations don't mention directly product evaluation or system certification. They are focus process improvement efforts on problem identified areas" [2].

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 Federal Aviation Administration integrated Capability Maturity Model (FAA-iCMM)

The FAA-iCMM is widely used in the Federal Aviation Administration. The FAA-iCMM gives a best practices model for enterprise-wide improvement. This model includes outsourcing and supplier management. Also integrated enterprise management, information management, deployment/transition/disposal, and operation/support areas are covered at the latest version. The ISO 9001:2008, EIA/IS 731, Malcolm Baldrige National Quality Award and President's Quality Award criteria, CMMI-SE/SW/IPPD and CMMI-A, ISO/IEC TR 15504, ISO/IEC 12207, and ISO/IEC CD 15288 standards and models can be integrated with FAA-iCMM [9].

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

The SSE-CMM purposes to improve and evaluate the security engineering capability of an organization. This model provides a comprehensive model for evaluating security engineering practices against the generally accepted security engineering principles. This model can be used to measure and improve performance in the application of security engineering. The SSE-CMM is published as an ISO standard which is ISO/IEC 21827 and version 3 is now available [4].

2.4 Microsoft Security Development lifecycle

Microsoft Security Development Lifecycle has adopted for software development that needs to withstand security attacks [5]. 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.

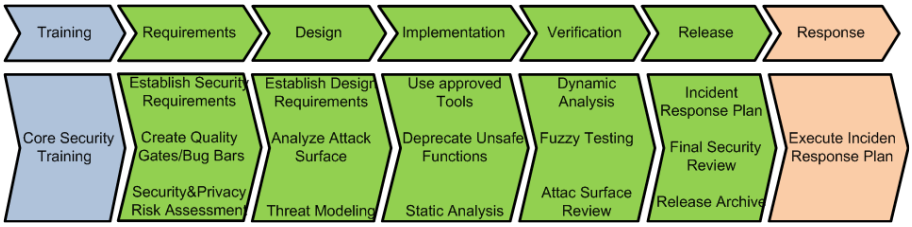

Figure 2. Simplified security development model

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. As a result of this policy software is designed, developed, and tested for security. The Microsoft SDL has been getting matured into a well-defined methodology.

Basic steps of SDL model is shown at Figure 2. Microsoft has added a lot of new property and capability since 2002. Important of this capabilities such as bug bar, fuzzing, 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, external tool releases [5].

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 try 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 was can be utilized by any size of organizations using any style of software development. Additionally, this model can be applied organization-wide, for all of business, 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 Business Function defines three Security Practices. Each security practice build assurance for the related Business Function. So overall, there are twelve Security Practices that are the independent silos for improvement that map underneath the Business Functions of software development

Governance, Construction, Verification and Deployment critical business functions take part at the top level of the SAMM hierarchical model.

Governance includes concerns 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 are regulated this business function at organization or project.

Construction concerns 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 is concern the processes and activities related to how an organization checks and tests error produced at software development phase. This is focused on design review, security testing and code review.

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

OpenSAMM model is shown at figure 3. OpenSAMM is created model like CoBIT (Control Objective for Information and Related Technology) to measure quantitatively every security objective. In this model security operation maturity level take a value between '0' and '3'. '0' means operation is not applied, '1' means there is not a systematic approach but there is basic level application at organization. '2' means operation is applied enough maturity level at organization. '3' means operation is applied perfectly at organization. According to this model project or time based audits could be improve organization security level [6].

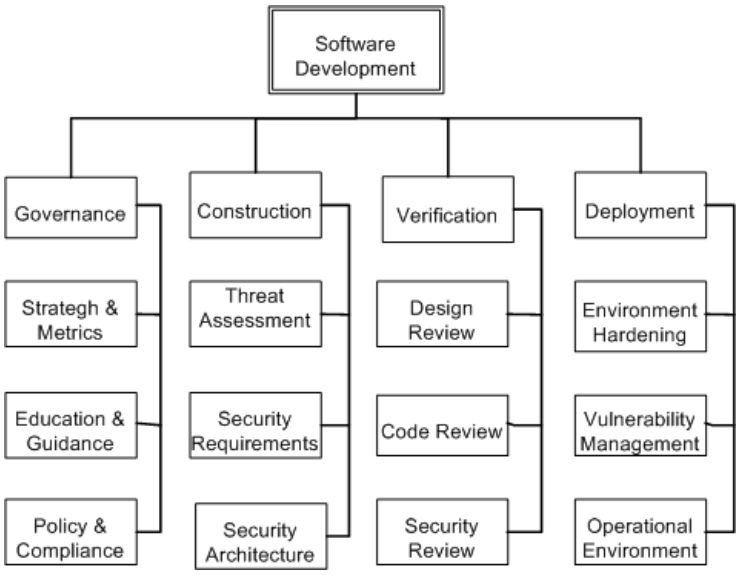

Figure 3. OpenSAMM model

3. COMMON CRITERIA SECURE SOFTWARE DEVELOPMENT APPROACH

The CC (ISO 15408) provides guidance for the development, evaluation and/or procurement of IT products with security functionality. The CC permits comparability between the results of independent security evaluations. This compatibility is provided a common set of requirements for the security functionality of IT products and for assurance measures applied to these IT products. These IT products may be implemented in hardware, firmware or software [8].

In the evaluation process, firstly Target of Evaluation (TOE) is checked for CC conformance than carried out vulnerabilities and functional tests. In addition to functional and vulnerability tests, CC provides methodology and discipline capability to design process, minimum vulnerability, maximum security and reliability for targeted assurance level. CC include risk analysis, configuration management, methodological design, life cycle, development tools and techniques, development security, flow remediation, functional tests in development and disciplines for Evaluation Assurance Level 4(EAL4) and upper level evaluations[8].

3.1 Risk Analysis

Risk analysis is mandated at product design level CC standard ASE (Security Target) assurance class. If product is evaluated according to CC designer must be define TOE, TOE operational environment, non TOE assets (software, hardware, firmware), user profile, legal and organizational security issues for TOE. All assets will be protected by TOE and effected by TOE availability are listed. All probable threats, threat agents and threat scenarios are detailed suitable with organizational policy. Last step at risk analysis process every threat, assumption and organizational security process are related and checked with IT and non IT security requirements by designer.

This risk analysis process mandated at design level to minimize security threat risks. Every threat is concerned providing targeted assurance to the TOE or TOE operating environment.

3.2 Configuration Management

Purpose of Configuration Management (ALC_CMC) assurance family is minimizing human mistakes at design and implementation level. Complex TOEs are increase project personnel number and require discrimination of rolls and responsibilities. Configuration Management tools control rolls, responsibilities, documents access according to configuration management plan such as CVS, SVN etc. These controls provide more security and reliable TOE preventing wrong code execute, outdated design document, false delete, wrong version create.

3.2 Methodological Design

Methodological design is performed ADV (Development) assurance class in CC. IT product is described step by step sub systems, modules, security functions respectively. All detail of internal and external interfaces between sub systems, modules and security functions are described. In this description security functions, interface behaviours and error messages are defined. Upper level CC assurance packets design (EAL5, EAL6, EAL7) is require semiformal (UML) or formal language (Z model). This approach is minimized the logic error in informal design level and provide maximum security and reliability at design level.

3.3 Life Cycle Model

Purpose of life cycle model is to define requirements for providing security, reliability and privacy for TOE being planned for development. These are a set of critical properties for computer applications that are vital to running an enterprise, such as accounting, authorization,, payroll, supply chain management, and resource planning applications.

Poorly controlled development and maintenance of the TOE does not meet all of its Security Functional requirements. Therefore, a model for the development and maintenance of a TOE must be established as early as possible in the TOE's life-cycle.

Necessary control over the development and maintenance of the TOE can be provided a life cycle model. If the model enables sufficient minimization of the risk the TOE will meet its security requirements. CC require a lifecycle model in design level such as waterfall model, spiral model etc.

3.5 Tools and Techniques

Tools and techniques is reference the tool selection that is used to develop, analyse, implement and manage the TOE. It helps to prevent misdefined, inconsistent or incorrect development tools from being used to develop the TOE. This includes, but is not limited to, programming languages, configuration management tools, documentation, standards, and other parts of the TOE such as supporting runtime libraries, interfaces and third party software or software.

Tools and techniques distinguishes between the implementation standards applied by the developer and the implementation standards for “all parts of the TOE” which include third party software, hardware or firmware.

CC evaluate tools and techniques with ALC_TAT assurance class in development process.

3.6 Development Security

Development security is interested in physical, procedural, personnel, environmental and other security and safety measures that may be used in the development environment to protect the

TOE and its parts. It includes the physical security of the development location and any procedures used to select development staff

CC evaluate IT development environment with Development Security (ALC_DVS) assurance class. These controls lead to produce more secure and reliable IT products.

3.7 Flaw Remediation

Flaw remediation purposes that discovered security flaws be monitored, recorded and corrected by the developer. Future compliance with flaw remediation procedures cannot be determined at the time of the TOE evaluation, but it is possible to evaluate the policies and procedures that a developer has these capabilities.

This family guarantee that the TOE will be maintained and supported in the future. This is provided TOE developer using tracking and correcting flaws mechanism. Additionally, requirements must be included distribution of flaw corrections.

CC evaluate IT flaw remediation process with Flow Remediation (ALC_FLR) assurance class. This leads to produce more secure and reliable IT products.

3.8 Tests

IT product is tested methodologically using Test (ATE) assurance family in CC. Every security function is tested internal and external interfaces and behaviours described at design documents. CC standard provide methodological test method.

The emphasis in this class is on confirmation that the TOE security function operates according to its design descriptions. This class does not address penetration testing, which is based upon an analysis of the TOE security function that specifically seeks to identify vulnerabilities in the design and implementation of the TOE security function. Penetration testing is addressed separately in the Vulnerability Assessment Class (AVA).

4. SOFTWARE DEVELOPMENT MODELS SECURITY APPROACH

In this section, security properties of CC, SSE-CMMI, Microsoft SDL, and OpenSAMM standards and models are compared according to security properties of the models 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.

CC is product evaluation focused standard and it expects that the producer completes required criteria according to the security target. In this paper CC is dealt with as secure software development guidance. SSE-CMMI recommends a model that security should be considered at system development level. However SEE-CMMI is not directly secure software development focused model. CC, 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. SSE-CMM, Microsoft SDL and OpenSAMM check personnel education and awareness. Since CC is a product focused standard it is not directly interested in this subject. On the other hand in the product evaluation process, evaluators can visit development environments and check personnel education and awareness.

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

Table 1. Comparison of secure software development standards and models

	Common Criteria	**SSE-CMM**	**Microsoft-SDL**	**OpenSAMM**
Security Training and Awareness	X	✓	✓	✓
Physical and Logical Security	✓	✓	X	X
Secure Configuration Management	✓	✓	X	X
Law, policy and procedure compliance	X	✓	X	✓
Threat Modeling	✓	✓	✓	✓
Risk Analysis	✓	✓	✓	✓
Security Requirements Definition	✓	✓	✓	✓
Security Architecture	✓	✓	✓	✓
Secure Design	✓	✓	✓	✓
Source Code Analysis	X	X	✓	✓
Vulnerability Analysis	X	✓	✓	✓
Security Verification	✓	✓	✓	✓
Vulnerability Management	✓	✓	✓	✓
Secure Development Techniques and Applications	X	✓	✓	✓
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. CC 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 are either directly or indirectly affect the organization software or hardware products. Microsoft SDL and Common Criteria do not directly address law, policy and procedure compliance but SSE-CMMI and OpenSAMM consider these items.

Threat modelling is used to realise meaningful security risk. This approach provides that development teams to 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. OpenSAMM is carried out application specific threat modelling at construction phase. Security Management Family describe security threats to evaluated product at Common Criteria. Threat modelling has been carried out since early phase of Microsoft SDL.

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

A very important part in the software development process for the achievement of secure software systems is that known as security requirements which provides techniques, methods and standards for tackling this task in the information system development cycle. Software development process must be repeatable and 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 three standards consider that threat, assumption and organizational security process are related and checked with IT and non IT security requirements by designer.

Secure architecture and design needs to be take in to account the very early phase of the software development lifecycle. If this needs not to be considered, this 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 standards and models take care of security architecture and design.

Source code analysis is very important for finding vulnerabilities at secure software development. Source code analysis can be made by commercial tools. But Analysis results should be reviewed by experts to eliminate false positives. Source code analysis is not mandatory in the CC standard. If the evaluator wants to perform source code analysis this can be done. SEE-CMMI does not directly address source code analysis. Microsoft SDL and OpenSAMM are especially emphasize importance of source code analyze and encourage to use automatic tools.

Vulnerability analysis includes black box and wide box testing. Vulnerability analysis is not mandatory for developers in the CC standard but evaluators have to do vulnerability analysis during evaluation phases. SSE-CMM, Microsoft-SDL and OpenSAMM require vulnerability analysis.

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 component of security validation. All standards and models include security validation. Microsoft SDL and OpenSAMM have detailed procedures for security validation.

Vulnerability management focused on notification of detected security vulnerabilities and weakness to the development team, deal with vulnerability and software updates. All standards and models include vulnerability management processes.

Secure development techniques cover definition of secure code development procedures and use of the best security practices during the development phase. CC is a product focused standard so it does not directly mention this subject. But evaluators could check this subject at laboratory visits. SSE-CMM, Microsoft SDL and OpenSAMM have criteria for this subject.

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 standards and 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, security standard and models deal with 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 non of them could provide all requirement of secure software development lifecycle. CC which is an international IT security standard (ISO/IEC 15408) allows an objective evaluation to validate that a particular product satisfies a defined set of security requirements. This paper shows CC can be used as a secure software development lifecycle guidance for developers in addition to use evaluation of IT products. Some security functions such as law, policy and procedure compliance can be added to the future versions of the CC standard to provide more secure products and information to the customers.

REFERENCES

[1] Madello D., Medina E. F., Pianttini M., (2007) "A common criteria based security requirements engineering process for the development of secure information systems", *Computer Standards & Interfaces*, Vol. 29, No. 2, pp244-253

[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/

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

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

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

[7] Gregorie J., Buyens K., Win B., Scanfariato R., Joosen W., (2007) "On the Secure Software Development Process: CLASP and SDL Compared", *29th International Conference on Software Engineering Workshop*

[8] Common Evaluation Methodology v3.1 rev 3, (2009) http://www.commoncriteriaportal.org/cc/

[9] Woody C., (2009) "Secure Software Development Life Cycle Processes", *Carnegie Mellon University*, https://buildsecurityin.us-cert.gov/bsi/articles/knowledge/sdlc/326-BSI.html

[10] Khan M. U. A., Zulkernine M., (2008)"Quantifying Security in Secure Software Development Phases" *32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC)*, pp:955-960, 28 July- 1 August 2008, Turku

[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

[12] Brian Chess, Brad Arkin, (2011)"Software Security in Practice", *IEEE Security & Privacy*, Vol.9- No.2 pp88-92

[13] Davis N., (2005)" Secure Software Development Life Cycle Processes: A Technology Scouting Report", CMU/SEI-2005-TN-024 Technical Report, *Carnegie Mellon University*,

[14] Halkidis S. T., Alexander Chatzigeorgiou A., Stephanides G.,(2006) "A qualitative analysis of software security patterns", *Computers & Securit,y* Vol. 25, No. 5, pp379-392

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

Authors

Mehmet Kara is a Principal Researcher at UEKAE. He received his Ph.D. in Electronics and Communications Engineering from Kocaeli University. His research interests include most aspects of network security and in particular, the areas of secure network design, network security testing, intrusion detection/prevention, firewalls, and secure software development and malicious code analysis. Mehmet took part in the foundation of the Common Criteria laboratory in Turkey and he is a Common Criteria evaluator of this laboratory since 2003. Mehmet has served as program committee member and reviewer in various conferences. His research has been published in national and international journals and conferences.