CHAPTER 2

Literature Review

Security vulnerabilities cause huge losses to the business organizations in terms of revenue, privacy, time, etc. Consequently, most of the organizations try to protect their endpoints from various authorized/unauthorized entities. Security is considered as one of the important aspects of software quality during execution which can be observed at run time. The diverse research areas in the field of security exist and include security flaw taxonomies, security models, security through software development, network security, security management, security metrics, security design tools etc.

A number of security flaw taxonomies have been developed to organize the security flaws that can be used by software developers to understand the basic kinds of errors. Some of taxonomies are Protection Analysis (PA) taxonomy, Research in Secured Operating System (RISOS) security taxonomy, Landwher’s taxonomy, Aslam’s taxonomy, etc. Landwher’s taxonomy (NRL) focuses on the problems of building and operating systems that can support security policies of the organization. It classifies the security flaws on the basis of genesis (how), time of introduction (when), and location (where) [L+94].

RISOS aims at studying the operating system flaws. The operating systems under study have been IBM’s OS/MVT, UNIVAC’s 1100 series, and Bolt Beranek and Newman’s TENEX system for PDP10. It classifies operating system flaws into seven categories viz. incomplete parameter validation, inconsistent parameter validation, implicit sharing of privilege/confidential data, asynchronous validation/ inadequate serialization, inadequate identification/authentication, violable prohibition/limit, and exploitable logic error [A+76].
PA taxonomy utilizes “pattern-directed protection evaluation” strategy. It considered security flaws from 6 operating system, namely CGOS, MULTICS, Unix and the operating system under study in RISOS project. Based on more than 100 errors that could permit system penetration from operating systems, PA developed classification scheme for the errors. The errors have been categorized in 5 categories namely, domain errors, incomplete destruction of data within a deallocated object, validation errors, naming errors and serialization errors [BH78]. PA and RISOS mainly focuses on the vulnerability classification rather than attacks. The classes of security flaws considered in both the taxonomies are approximately equivalent [HH04]. The taxonomy suffered the limitation as most of the vulnerabilities may fall across multiple classes [BB96].

The Aslam’s scheme categorizes and groups the faults which are based on implementation flaws of various versions of Unix operating system. The flaws have been categorized into configuration errors, synchronization errors, condition validation errors, and environment faults. The purpose of this scheme is to assist static analysis efforts [ASL95]. Aslam’s taxonomy has not been considered useful for static analysis as it did not consider input validation errors and serialization errors [H+10].

NRL taxonomy classifies the vulnerabilities according to genesis, i.e. how problem entered the system; time i.e. at which point problem entered the system; and location i.e. where in the system the problem is evident. Further refinements lead to other categories of flaws such as malicious and non-malicious error, validation error, domain error, etc. [L+94]. It utilized the flaws generated from various operating systems such as Unix, Univac, IBM OS/360, IBM MVA, Apple Macintosh etc. It suffers the limitation of categorizing the flaw to be malicious or non-malicious as it requires the decision of the programmer [H+10]. Moreover, the taxonomies considered the flaws originating in operating systems only.

Another taxonomy of vulnerabilities is based on three categories viz. main memory (dynamic and static memory), input output (network interface and file system) and cryptographic resources (randomness and, cryptographic algorithms
and protocols). The constraints and assumptions associated with each resource defined in three categories are further defined [BA07]. These taxonomies help in testing the weak areas of the system. The models normally evaluate security of existing systems but do not provide any remedy to the information security issues [BLA00].

In order to develop secured system, the designers must firstly decide security needs and secondly what does security means to them [LAN81]. For this, security models can be used. The security models are the means for formally expressing the rules of the security policy in an abstract detached way. There are various formal models for computer security. Examples include High-water-mark model, Access matrix model, Bell and LaPadula model, Information flow models etc. [WEI69, BL73, FEN74, DEN75]. Model by Grohn uses Bell and LaPadula model as a base and include integrity levels and compartments and also modified the directory structure of the base model [GRO76]. UCLA and Take grant models use the concept of access matrix model. Clark-Wilson model emphasizes on integrity in commercial systems. Security models often describe the security policies to be followed, but the actual implementation cannot follow the model exactly [JZ02]. The models are mainly based on safe guarding the systems through access control. Thus, there is a need to classify security in IT environment as a whole.

Traditionally, security has been implemented through trusted kernel that enforces all the security policies such as Bell and LaPadula Model. The secured system not only depends on design but also on the correct implementation [LINK11]. The software applications are majorly attacked as most as the software industry largely focus on securing operating system and not on secured applications. Attacking an operating system does not directly benefit a criminal, but attacking a database gives important data to the cyber criminal. The other areas of interest to the hacker include antivirus software and backup software [HL06]. The software systems use Access Control Lists (ACL) to manage large number of subjects and objects [BV06]. The software system may also be protected by use of firewalls, routers, spammers etc. Thus, the security
mechanisms reside on operating systems, application programs and also some system programs.

The security has to be applied in different areas since none of the security policies are foolproof. The various areas of security are physical security, hardware security, security through operating system, incorporating software security during SDLC, database security, security through various applications, network security, internet security, security by the laws of anti computer crime etc. [RAN04, LEV95, ZIP05]. Developers can estimate the cost of security related activities using Sustainable Software Security Process (S3P) by applying Analytic Hierarchy Process and an automated search heuristic to the models. These models are created using S3P. It supports the developers in determining what security activities are appropriate for them [BS09].

A number of methods and standards have been mentioned in the literature to evaluate security processes, product and activities such as IT Act 2000, Common Criteria (CC), NIST’s Special Publication SP 800-55, Office of Management and Budget-Federal Information Security Management Act (OMB FISMA) Reporting Requirements and PART/ PMA, SSE-CMM (Systems Security Engineering Capability Maturity Model), ISO/ IEC 15939, ISO/ IEC 27001/ 27004 etc. CC defines set of international and national standards for security evaluation for information security with initial standard known as TCSEC (Trusted Computer System Evaluation Criteria) or Orange Book [WAL03]. These standards mainly focus on information security.

IT compliance is a key to business risk management. It provides the laws, regulations and standards for the operation of the business [HLA07]. Sarbanes-Oxley is a US federal law that provides IT issues and control in financial services. Federal Financial Supervisory Authority is a German compliance for minimum requirements for risk management. IT Act 2008 focuses on data protection in cyber world in India.

The formal approaches to address the flaws throughout the software development process can help reduce the security risks [HM04]. A variety of
tools and techniques to address security have been mentioned in the literature. Misuse cases document and decide how software should react to illegitimate use. It can be generated through informed brainstorming. Attack patterns may be useful in generating abuse and misuse cases [HMA04]. Threat modeling allows early identification of security threats. It is a conceptual tool to analyze data flow to find security vulnerabilities and the methods for exploitation. It is being used in design phase during SDL [HL06, SIM11]. Threat scenario is a series of events that model the risk assessment process by focusing on impact or cost, vulnerability and threats. It can be performed with the help of attack trees [LSS11]. Least Privilege allows specifying minimum set of privileges. It can be applied at two levels. Firstly, applications should be allowed to run with minimum privileges; and secondly, the users must be assigned minimum rights to work on the system. It can define access control on the basis of the roles of the users. A system with fewer privileges may be difficult to compromise. Moreover, software engineers can incorporate quality in a software using models such as Unified Modeling Language (UML) during the early phases of SDP. Focusing on quality can help in detection and avoidance of vulnerabilities.

Several frameworks have been developed that focus on securing software systems during the development process. Security Oriented Software Development Framework (SODA) makes use of specific techniques and tools such as security architecture, security design guidelines, security design principles, security patterns, threat modeling etc. in a more realistic way for a normal developer [MJ08]. Security vulnerabilities can also be identified on the basis of input, data, processing and output and suggestion for strategies to reduce and eliminate these vulnerabilities during software development lifecycle is mentioned in [BH09]. Secure Software Development Model (SSDM) integrates secure engineering with software engineering to produce more effective software systems. SSDM allows threat modeling in requirements gathering stage [SOA06, KZ09]. Execution phase audit shows that security flaws are mainly due to buffer overflow, misplaced trust, race condition and poor random number generator [CB05]. Another secured software development model
considers five process areas namely, secure development processes, secure management processes, organization security focus, discovery of security vulnerabilities and risks, and corrective security actions [LINK12]. Most of the frameworks focus on security tools and techniques without considering security during pre-development stages.

Development process must focus on the methodology to reduce/ eliminate security flaws during development process. Thus, security is also incorporated formally in the software development lifecycle by integrating it with the security assessment and mitigation tool. The tools developed are Software Security Assessment Instrument (SSAI) and Defect Detection and Prevention (DDP) risk management tools [GIL04]. A method has been presented to prevent vulnerabilities from being introduced during software development process based on formal modeling of vulnerability causes [BS08]. Another method Process to Support Software Security (PSSS) having some activities grouped in sub-processes is used to help obtain secure software by integrating it into software development life cycles. PSSS supports knowledge management on security inspections along with the checklist to security inspections on software requirements [NBA09]. Such classifications are not able to provide detailed guidelines to include security in the software development process.

It has been observed that the developers do not include security aspects during all the development stages of software [LEV95]. Even if security is included, it is normally incorporated late or during some phase in SDP [PET04]. One of the reasons is that the product development by necessity must optimize the use of limited resources such as time, money and personnel [SCH04]. This applies to both aspects of secured software and information system development, viz. the quality aspects of security (reducing vulnerabilities that could pose a risk to security), and the security features that can be seen as requirements of an IS or a software. Further, security should be made part of the system according to user needs. The former prevents threats generating from the system itself (by reducing vulnerabilities in the system) and the latter prevents threats coming from outside
the system, e.g. vulnerabilities in the system or persons using it that an outsider accidentally or deliberately abuses [RAM04].

Security can also be compromised by the threats originating due to environmental variables, date, numeric overflows, default settings and information exposure [G+03]. These problems also depend on the type of development language being used. The flaws in software arise because security knowledge is not widespread among the developers [PEI05]. Moreover, the developers do not deliberately write insecure code but most of the time they get late information regarding the aspects to be covered in security.

The security issues must be considered during the SDLC and should be integrated within software engineering languages, methods, methodologies, and processes. A number of methods have been developed to incorporate security throughout SDP. Secure Development Lifecycle (SDL) is a secure development process which focuses on reducing the security related design and coding defects. For example, IIS 6 has been implemented using SDL [HL06, LINK13]. CLASP or Comprehensive Lightweight Application Security Process supports the development of software with security being the major focus. It is a set of 24 core activities, grouped in seven categories known as best practices. The best practices include institute awareness programs, perform application assessments, capture security requirements, implement secure development practices, build vulnerability remediation procedures, define and monitor metrics, and publish operational security guidelines. The activities are role based process components that can be integrated in any development process. [LINK14]. It has been suggested to implement security on the artifacts generated by the software development process. It has grouped the best practices in seven Touchpoints. The Touchpoints in the order of effectiveness such as code review, architectural risk analysis, penetration testing, risk-based security tests, abuse cases, security requirements, and security operations that contribute to software security. All Touchpoints once implemented can result in secure software development lifecycle. The development methodology is based on risk management, areas of improvement and ancillary resources such as knowledge base [MCG06].
Implementation of SDL may increase the cost of product by 15% to 20%. Also, there is no indication on time required to implement SDL. OWASP focuses only on web-based software. It does not support the informed commercial security policy but is a forum for information technology professionals to share and build expertise [LINK15]. Literature review reveals that the security approaches are not structured in nature but provides security activities during development of software. Even after taking so much care in development of software, security breaches do occur. However, the correct use of software is also very important as the correctness of the software itself [YEE02]. Hence, structured method to incorporate security in the software development process needs to be incorporated and also security must be considered from the user’s perspective [MG05].

Key to secured software is the security requirements. Security requirements may help in supporting privacy. Security Quality Requirements Engineering (SQUARE) is a process that can support organizations to consider security during early stages of software development. The steps include agree on definitions, identify assets and security goals, develop artifacts to support security requirements definition, perform risk assessment, select elicitation techniques, elicit security requirements, categorize requirements, prioritize requirements, and inspect requirements [MS05]. Software Security Checklists (SSC) can support in requirements gathering, design, code issues, maintenance and decommissioning of the systems [G+03]. Unlike use cases that generate functional requirements, misuse case can help in generating security requirements [SO05]. Misuse cases denote use cases with failure and appropriate uses cases may mitigate misuse cases. The methods do not consider functional requirements. Moreover, security requirements are not gathered from the various stakeholders. Considering the stakeholders can bridge the communication gap between the requirements gathering team and the stakeholders.

Proper security metrics and measurement process is a means to facilitate security decisions [AAP08]. The metrics program can be developed by following a procedure focusing on goals and needs of the organization [WES05]. SANS Institute gives a program to establish security metrics that include defining goals
and objectives, decide metrics to generate, develop strategies to generate metrics, establish benchmark and targets, determine how metrics will be reported, create an action plan, and establish a formal program review [PAY06]. A method to quantify security state of a particular SDLC artifact based on prioritizing vulnerabilities depending on the potential damage they can cause is also proposed [KZ08]. The trust worthiness of the software development lifecycle can be evaluated for security assurance by taking input as context, expert opinion, outcomes of the methods and reputation [U+09]. It has been emphasized that the system of security metrics is in the first priority among the tasks for cyber security. Software complexity may or may not be related to security problems. If an empirical relationship can be discovered between software security metrics and security problems at any level (e.g. code, design, or architecture level), these metrics could aid organizations in their efforts to fortify their products early in the development lifecycle [LEW07].

Implementing security has some limitations e.g., encryption algorithms have theoretical limitations, believing “insiders”, secure passwords by users, etc. The trusted SSL issues security certificate but does not authenticate server by default. Even, sometimes the certification authority (CA) signs wrong certificate for the client [VM04]. Security Engineering Process using Patterns (SEPP), a pattern- and component-driven process, is a systematic process to secure software development. The development step makes use of security software components that represent implementation of security mechanisms [HHS08].