Sādhanā Vol. 42, No. 9, September 2017, pp. 1481–1493
DOI 10.1007/s12046-017-0696-7

An integrated framework for software vulnerability detection, analysis and mitigation: an autonomic system

MANOJ KUMAR[1,*] and ARUN SHARMA[2]

[1]University Teaching Department of Computer Science and Application, Makhanlal Chaturvedi National University of Journalism and Communication, Bhopal 462011, India
[2]Department of Information Technology, Indira Gandhi Delhi Technical University for Women, Delhi 110006, India
e-mail: m_pachariya1@yahoo.com; arunsharma2303@gmail.com

MS received 30 June 2016; revised 12 January 2017; accepted 25 January 2017; published 15 July 2017

Abstract. Nowadays, the number of software vulnerabilities incidents and the loss due to occurrence of software vulnerabilities are growing exponentially. The current existing security strategies, the vulnerability detection and remediating approaches are not intelligent, automated, self-managed and not competent to combat against the vulnerabilities and security threats, and to provide secured self-managed software environment to the organizations. Hence, there is a strong need to devise an intelligent and automated approach to optimize security and prevent the occurrence of vulnerabilities or mitigate the vulnerabilities. The autonomic computing is a nature-inspired and self-management-based computational model. In this paper, an autonomic-computing-based integrated framework is proposed to detect, fire the trigger of alarm, assess, classify, prioritize, mitigate and manage the software vulnerability automatically. The proposed framework uses a knowledge base and inference engine, which automatically takes the remediating actions on future occurrence of software security vulnerabilities through self-configuration, self-healing, self-prevention and self-optimization as per the needs. The proposed framework is beneficial to industry and society in various aspects because it is an integrated, cross-concern and intelligent framework and provides more secured self-managed environment to the organizations. The proposed framework reduces the security risks and threats, and also monetary and reputational loss. It can be embedded easily in existing software and incorporated or implemented as an inbuilt integral component of the new software during software development.

Keywords. Autonomic computing; software security; software vulnerability; vulnerability detection; vulnerability mitigation; vulnerability optimization.

1. Introduction

Nowadays, software systems are playing vital and multi-faceted roles in our daily life. The software components are the heart and prominent constituent of the almost all modern and complex systems. Therefore, interdependent and networked software systems are widely used by several organizations for their business decisions. These software systems are used to manage and control the business operations and performance. Almost all industries have expanded their business horizons, improved the business performance and earned huge profit through the usage of the software systems but simultaneously several organizations have also incurred huge loss in terms of money and reputations due to security breaches, low security standard, violation of security, security threats and attacks, and induced software vulnerabilities in legacy and modern systems [1–3].

In the present scenario, the software systems are being assembled by integrating several Commercial Off-The-Shelf (COTS) components by their manufacturers. These software systems have high security risks due to insertion of known or unknown software vulnerabilities in the COTS during their development. The insertion of the software vulnerabilities in the COTS during software development is one of the prominent reasons and factors for enhancement of incidents of security breaches and reduction of the performance of software systems assembled by integrating these COTS. These risky software systems are used to operate and control business. The high extent usage of these risky software system increases the number of occurrences of vulnerability incidents, security threats, attacks and the monetary and reputational losses incurred due to security breach and vulnerability occurrence. These risky software systems are not capable of reducing the security risks,

*For correspondence

attacks, threats and financial losses, and preventing the occurrences of vulnerabilities. These risky software systems are not able to provide secure environment to the organizations because they can be easily exploited by hackers or attacker to get entry into the software systems to break the security. It implies that the vulnerability offers a possible entry point to the software system [3–5].

The software vulnerability is a weakness or flaw or defect left unaddressed or untested in the software components during their development, knowingly or unknowingly. Vulnerabilities may be exploited at any point of time and tend to violate security policies. Several incidents of threats and attacks are reported to the Computer Emergency Response Team (CERT). These vulnerability incidents have caused huge loss of money and reputation. The loss due to Code Red Worm was estimated to be $2.6 billion. In all, 4129 number of incidents of vulnerabilities were reported to CERT Coordination Center in 2003, which was 70% increment of the incidents of 2002. Currently, the rate of reporting vulnerabilities to CERT Coordination Center is 4,000 per year; the number of vulnerability incidents and the monetary loss are growing exponentially. This is very serious concern for CERT, software industry and also for security research community [3, 5, 6].

The exponential growth rate of incidents of occurrence of software security breaches and huge irreparable loss of the organization have forced and motivated the software industry to develop software 100% free from vulnerabilities. The vulnerability-free software is highly needed to reduce security breaches, risks and the irreparable loss, and to improve the performance of software system. Though the software industry has employed the penetration testing to develop the software 100% free from vulnerabilities, it is impractical to develop a software that is completely free from vulnerabilities [7–9]. The software industry also employs the patch management for reducing the security risks and optimizing the performance of software system but the penetration testing, patch management and legacy security tools are not sufficient to attain the target level of security. In vulnerability detection and management, the scanning of operating environment and software systems is the prime activity and requires automation and continuation because the delay in scanning the system for vulnerabilities makes environment and system more vulnerable for a long period. The delay in scanning of environment and system enhances the irreparable loss, possibility of violation of security policy and the number of vulnerability incidents [10–12]. The study of literature and CERT reports proves that the legacy security systems, the existing vulnerability detection and remediating approaches are outdated, and are not intelligent, automated and self-managed. Hence, the outdated systems are not competent to provide secured, intelligent and self-managed software environment to the organizations. Therefore, cost-effective, scalable, intelligent, cross-concern, automatic and autonomic alternative-knowledge-based approaches for vulnerability detection, assessment and optimization of security and vulnerability, vulnerabilities analysis, mitigation and management activities are highly required to reduce the security risks, threats, occurrence of vulnerabilities incidents and the monetary loss [13–15].

The autonomic computing is a nature-inspired and self-management-based computational model. The Self-Star property of autonomic computing efficiently manages and controls the components of complex systems without any input from the user. The self-configuration, self-healing, self-optimization, self-protection (CHOP), self-regulation, self-awareness, self-learning, self-creation, self-governance, self-description and self-organization form the common thread of Self-Star of the autonomic computing. In this paper, an autonomic computing-based integrated framework for software vulnerability detection, assessment, classification, prioritization, mitigation and management is proposed. The proposed framework incorporates the CHOP properties to manage the software vulnerabilities. In the proposed framework, nature-inspired intelligent techniques are used for automatic identification, firing the trigger of alarm, assessment and analysis of software vulnerabilities. The proposed framework is beneficial to industry and society in various aspects because it is an integrated, intelligent, cross-concern, autonomic, nature-driven novel framework, which automatically detects the vulnerabilities, assesses their criticality and assigns priorities. The proposed framework uses knowledge base and inference engine, which automatically takes the remediating actions on future occurrence of software security vulnerabilities through self-configuration, self-healing, self-prevention and self-optimization as per the needs. The proposed autonomic integrated framework fulfils the security needs of organization and reduces the loss due to vulnerabilities occurrence. The proposed cross-concern framework can be embedded easily in existing software and incorporated or implemented as inbuilt integral component of the new software during software development.

This paper is organized as follows. Section 2 discusses the literature review of software security estimation, vulnerability detection, estimation and its optimization. Section 3 describes the basics of the autonomic computing. Section 4 describes the proposed autonomic computing-based intelligent framework for software vulnerability detection, assessment, prioritization, classification, mitigation and management. The final conclusion that can be drawn from this study is presented in section 5. Section 6 provides some pointers and guidelines for future research.

2. Existing works in the area of software vulnerability mitigation and management

The research in the area of software security quantification, vulnerability detection, estimation, analysis, mitigations and management has been carried out for years. Still, several issues are alive and left unaddressed. Hence, the

current state of art of the research in the area of software security and vulnerability management is still immature and ongoing. A brief literature study of research is as follows.

2.1 *Vulnerability identification*

Kannan *et al* [15] suggest a software vulnerabilities identification approach and point out that the disclosure of vulnerability is very important in vulnerability mitigation and management (VMM). Gilliam *et al* [17] strongly recommended that it is essential to incorporate the implementation of security in software during software development life cycle to protect corporate resources. Jiadong *et al* [18] proposed a clustering and model analysing method for detection of software vulnerabilities. They pointed out that the existing static vulnerability detection methods have high false positive and false negative rates. Hence, they used clustering technology to mine the pattern from the set of vulnerability sequences and constructed the Vulnerability-Pattern Library (VPL) to improve the efficiency of proposed method. Experimental results show that proposed method has lower false positive and false negative rates. Wang *et al* [19] proposed a new method for detecting software vulnerabilities based on rapid density clustering called DSVRDC. They used *rd-entropy* and *s-order*. In this proposal, they classified the vulnerability sequences using a density-based clustering approach. They used the variation of *s-order* for analysing the software vulnerabilities sequences. The proposed method improves the efficiency in terms of the accuracy.

2.2 *Vulnerability analysis and assessment*

Alhazmi *et al* [6] proposed an approach to categorize the software vulnerabilities on the basis of the cause or severity of vulnerability. Alves-Foss and Barbosa [20] developed a System Vulnerability Index (SVI) method to measure the susceptibility of computer system to common attacks. This approach helps system administrators to assess the starting point for security policy. Wang and Wulf [21] designed a security measurement framework for assessing security strength of a software system. Of course, this framework is not standard and universal. Gilliam *et al* [22] proposed and developed a Software Security Assessment Instrument (SSAI), to aid developers in assessing and assuring the security of software in early stage of development and maintenance. Butler [23] developed the SAEM approach to estimate and prioritize multi-criteria risk. Gilliam *et al* [17] developed an SSAI to produce secure software. Halkidis *et al* [24] proposed a quantitative approach for evaluating the known patterns of security using security guidelines. Hallberg *et al* [25] developed a framework for system security assessment. They claimed that their framework is able to categorize existing security vulnerabilities and threats. For the categorization of purpose, they used the CAESAR method to calculate overall system security values. Alhazmi *et al* [26] proposed the Vulnerability Density Metric (VDM) to measure the vulnerability of software. VDM is defined as the number of vulnerabilities per unit size of code. Chandra *et al* [27] proposed a framework to estimate software security in early stage of software development life cycle. Chen *et al* [28] presented a quantitative threat modelling method using Attack Path Analysis (APA). Chowdhury *et al* [29] defined a number of code level security metrics to assess the level of security of a given program using code inspections. Alshammari *et al* [30] proposed seven design metrics for the software security of an object-oriented software. These metrics help designers to determine and fix security vulnerabilities at an early stage of software development. Agrawal and Khan [31] developed an algorithm to measure Attribute Vulnerability Ratio (AVR) of an OOD. The proposed approach is implemented and validated on a case study of Automated Teller Machine (ATM).

Alkussayer and Allen [32] developed a framework to assess software architecture. It can also be used to determine the satisfaction level of the projected security requirements. Christopher *et al* [33] proposed an integrated framework for measurement and analysis of software security for distributed environment. Agrawal and Khan [34] proposed the design metric to quantify the vulnerability of object-oriented design (OOD). The proposed metric is used to determine whether the design of one version of a software system is more vulnerable than another with respect to propagation of vulnerability. They also pointed out that coupling is responsible for propagation of vulnerabilities in the design of object oriented software. Agrawal and Khan proposed [35] the Vulnerability Confinement Capacity (VCC) metric to find out the role of encapsulation in improving security of an OOD. The encapsulation protects software system from unauthorized access. Their previous works have proved that inheritance and coupling support vulnerability propagation and hence decrease security.

Alhazmi *et al* [36] examined the feasibility of quantitatively characterizing some aspects of security. They predicted the number of undiscovered vulnerabilities present in a software system using dynamics of vulnerability discovery. They used several major operating systems as complex software for validation of the proposal. They found that the density of vulnerabilities in a program is a useful measure and can be used to minimize the vulnerability of software security. Khan and Han [37] introduced a component security characterization framework. This framework characterizes security properties using software security profiles.

2.3 *VMM*

In a separate study, Wang *et al* [38] presented the ontology for vulnerability management. In this study, the attack

patterns were ranked based on the vulnerability information present in the ontology. Agrawal and Khan [39] proposed a framework for security vulnerability minimization using design of object-oriented software. The proposed framework minimizes vulnerability by restricting the flow of vulnerable information. Agrawal and Khan [40] proposed a framework to identify, analyse and mitigate vulnerabilities during the development life cycle. They prepared the security checklists for various phases of software development life cycle to examine the vulnerability.

3. Autonomic computing

Autonomic computing is a self-managing computing model. It is derived from the autonomic nervous system of human. In human body, the autonomic nervous system regulates body subsystems without conscious input from the individual. In a similar way, an autonomic computing-based software system controls the functioning of its subsystems without user intervention. In this framework, the CHOP properties of autonomic computing system (ACS) are incorporated into the complex software system to develop inbuilt self-managed and self-run characteristics without external or user intervention for software vulnerability mitigation and management (SVMM) [3, 4, 16, 41]. The ACS has the following key components or elements:

- ACS maintains complete, correct and specific knowledge about all its components.
- ACS has ability of self-configuration on occurrence of unpredictable conditions.
- For optimal functionality, ACS must have self-monitoring ability.
- ACS must have ability of self-healing of the system, i.e., to provide an alternative way to function when any problem occurs in the system.
- ACS must have the ability of self-protection from vulnerabilities.
- ACS must be able to adapt to environmental conditions.
- Open standards are the basis for ACS. It must not use the proprietary technologies; it must anticipate demand while remaining transparent to the user.

4. Proposed autonomic computing-based integrated framework for software vulnerability detection and mitigation

The complexity of software systems, the amount of irreparable loss, numbers of cybercrimes and vulnerabilities incidents are growing day by day. The usage of artefacts of penetration testing, patch management, firewall and antivirus is not sufficient to combat the security threats, and to reduce the irreparable loss and the number of the vulnerability incidents. Hence, vulnerability-free software should be developed to reduce security risks, irreparable loss and improve the performance of software system. However, it is very difficult to develop a software that is completely free from vulnerabilities. Therefore, an alternative approach is required to be devised to optimize security and prevent the occurrence of vulnerabilities. The integrated framework may be one of the best alternative remedies for software vulnerability detection, assessment, classification, prioritization, risk analysis, mitigation and management.

In this paper, the autonomic computing-based integrated framework is proposed to identify, analyse, classify, mitigate and manage software vulnerability. The proposed framework uses the CHOP properties to manage the vulnerabilities. In this framework, the nature-inspired intelligent techniques are used for automatic identification and measurement of software vulnerabilities. The proposed framework uses a knowledge base and inference engine, which automatically directs the self-executer to take corrective actions on future occurrence of security vulnerabilities through self-configuration, self-healing, self-prevention and self-optimization as the needs. The proposed framework will reduce vulnerabilities incidents, cost of maintaining security, risk and loss. The proposed framework is also helpful in preventing the occurrence of security threats and vulnerabilities.

The SVMM process is the cyclical approach of discovering, classifying, remediating and mitigating software vulnerabilities. Figure 1 presents phases of the SVMM process.

Figure 2 presents the autonomic computing-based integrated framework for vulnerability detection, assessment, classification, risk analysis, mitigation and management. In the proposed framework, an intelligent scanner performs the self-monitoring and self-scanning of the software system for vulnerability identification. The vulnerability analysis is carried out automatically by the autonomous components of integrated framework. The VMM is a cross-cutting concern of security requirement. The remediating actions on occurrence of vulnerabilities are automatically taken by the software systems through the self-configuration, self-healing, self-optimization and self-protection activities. The self-monitoring, automated identification and assessment of vulnerability, knowledge management (database of vulnerabilities and rules, inference engine) self-effecter, autonomic VMM are the integrated components of proposed autonomic computing-based framework for software security vulnerabilities mitigation and management. Figure 2 presents and outlines the integrated intelligent autonomic framework for software security VMM. The details of individual components of the proposed integrated intelligent autonomic framework are as follows.

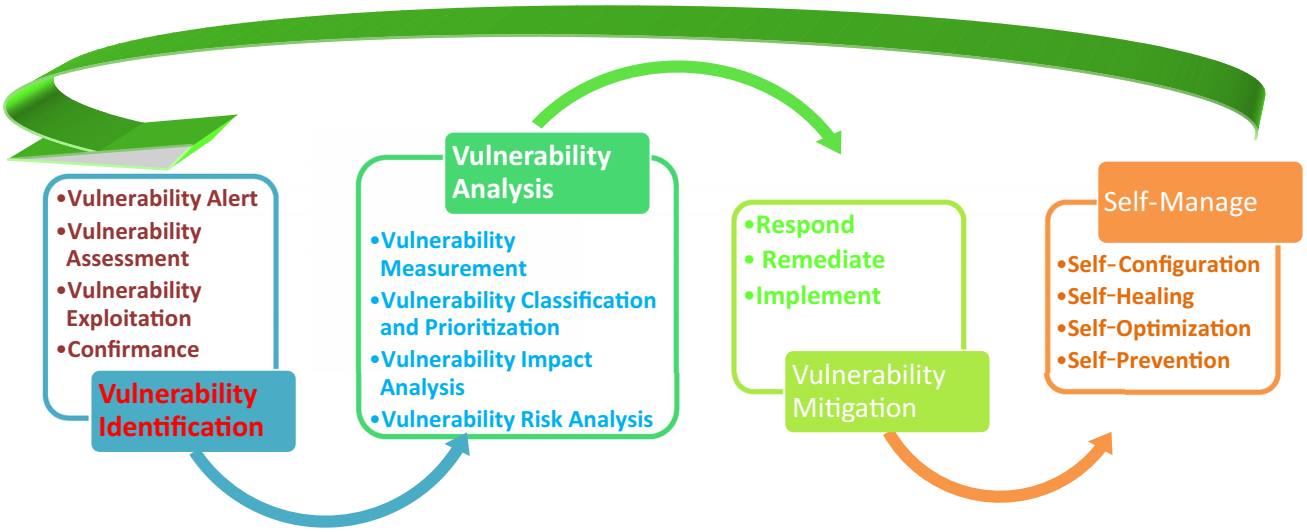

Figure 1. Software security vulnerability mitigation and management.

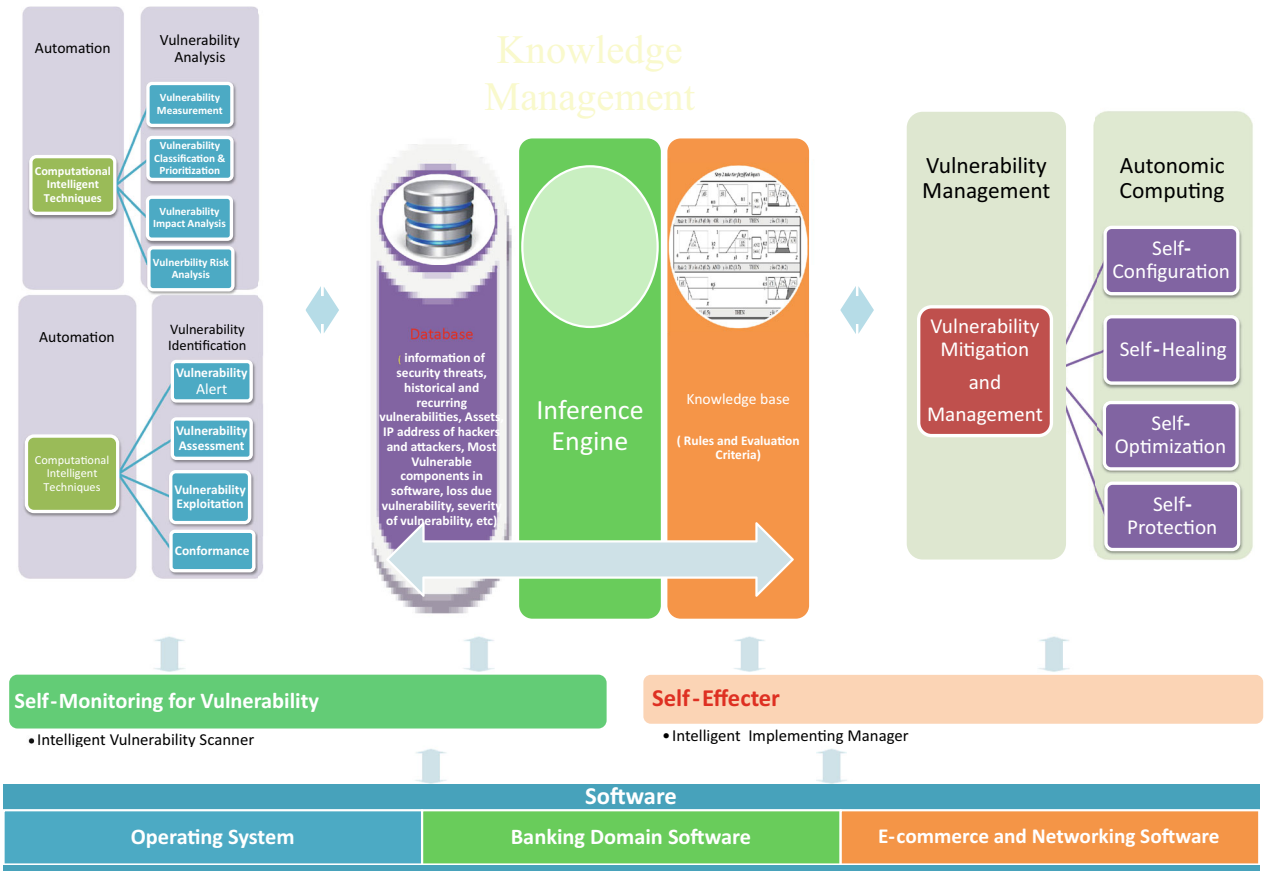

Figure 2. Autonomic computing-based framework for software security vulnerability mitigation and management.

4.1 *Self-monitoring for vulnerability*

The software system can be scanned manually or automatically by employing an online or offline scanner. Most of the existing vulnerability scanning approaches carry out static code analysis and review to identify weak points in the software. The existing vulnerability scanning approaches can scan only the source code of software. They are not competent enough to scan the exe file of software

system but the source codes of software systems are not available. Almost all vulnerability scanners are offline but at any point of time, hackers or attackers may exploit the weak point or development flaws to break the security. The existing vulnerability scanners cannot be used to carry out any type of vulnerability assessment.

Since the source codes of software systems are not available and the attackers or hackers never tell the time of attacks or threats, the existing vulnerabilities scanning approaches are not suitable for effective and efficient scanning and monitoring of software systems for vulnerability. The cost of existing vulnerability scanning approaches is also very high. Therefore, there is a need to develop a cost-effective, scalable, more accurate, intelligent, automatic scanning approach for self-monitoring of the software systems for known vulnerabilities. The genetic algorithms, clone detection, case-based reasoning (CBR) and other nature-inspired approaches may be the best suited for self-monitoring activity of vulnerability discovery.

In this framework, an automatic intelligent scanner is used for self-scanning and self-monitoring of the software system. The intelligent scanner is considered as an integral part of the software system. The proposed framework uses nature-driven intelligent techniques to scan the software system itself automatically for detection of occurrence of software vulnerability. This component of the proposed integrated framework searches the software security vulnerability, i.e., identifying the weak points of software or development flaws in software system for maintaining high security using database of known flaws, security holes and artefacts of security testing. The intelligent scanner explores the software system for occurrence of these flaws. The intelligent scanner provides an approach to identify the postern, vengeful code and other security threats in complex software. The self-monitoring component of proposed framework uses static and dynamic analysis of source and binary codes of software system. The intelligent scanner also automatically determines the weak points or missing functionalities of software system, whose absence could lead to security breaches.

The proposed integrated framework automatically generates a report and summary of its findings also. The report is prepared on the basis of business value, priority, criticality, severity of vulnerabilities and level of risks associated with them. The report also contains information about associated, effective and possible remediating actions and the time duration in which remediating actions are to be taken and completed. This report can be used by other autonomous component vulnerability identifiers for identification of type of vulnerability and firing the trigger of alert.

4.2 *Identification of vulnerability*

The vulnerability alert, assessment, exploitation and conformance of security requirements are autonomic sub-components of the vulnerability identification component of the proposed integrated framework.

4.2a *Vulnerability alert*: The report generated by an intelligent scanner during self-monitoring of software systems is the input for the vulnerability identification component. The vulnerability alert sub-component uses vulnerability description report for automatically firing the trigger of alert for occurrence of software vulnerability in software systems. The alert sub-component of proposed framework fires the trigger of alert for occurrence of vulnerability in software system after discovery of the preconditions of occurrence of the pre-existing flaws or known vulnerability with the help of a knowledge-based system (KBS).

4.2b *Vulnerability assessment*: In the proposed framework, a standard metric is used to measure and assign the criticality and severity of the vulnerability. The vulnerability assessment techniques identify the type of vulnerability present in software while the intelligent vulnerability scanner determines the occurrence of vulnerability. The vulnerabilities can be classified on the basis of the business value, severity and priority of vulnerability, and the importance of the assets lost due to vulnerabilities using an intelligent approach.

4.2c *Vulnerability exploitation*: The penetration testing is performed to exhibit the way of exploiting for real attack through destructive flaws. The prime purpose of penetration testing is to find out the unauthorized access or malevolent activity of software system by exploiting the known weakness points or vulnerabilities of software system. It is helpful in identifying the flaws, which may create the gesture for threat to software system. The penetration testing determines the exploitable flaws and also measures severity of vulnerability. Therefore, in this framework, the artefacts of penetration testing and KBS are used to find out the exploitable flaws in software system, which are prone to hacking and attacking.

4.2d *Conformance of security requirements*: The compliance of security requirements is the foremost activity of VMM. In this framework, the intelligent sub-component automatically determines the compliance of the organizational security requirements by measuring deviations between the implemented and the required security. It is carried out automatically, quickly, accurately and efficiently to protect the internal components of software system, sensitive data and reputation of organization.

4.3 *Vulnerability location finder*

In the proposed framework, the intelligent vulnerability location finder uses both the KBS and code analyser to automatically locate the weak points in software system. It can be accomplished by carrying out the following activities (figure 3).

- Automatically discover the known vulnerabilities using KBS and domain knowledge. The mapping of known vulnerabilities with the source code of software system is carried out automatically using KBS.
- Automatically find the dangerous system calls because the user-defined functions call the system calls and the system calls are the privileged entry points to get into the software system.
- Automatically identify the possible entry points of software system for attacks or threats.
- Find the tentative location of vulnerability or weak point in software system.
- Automatically carrying out the reachability analysis between entry point and tentative location of vulnerability using the call relationships with the help of system dependency graph.
- Automatically measure the exploitability risk of the individual vulnerability.
- Finally, the intelligent vulnerability location finder provides reachability category information. The reachability category information may contain one of the following options:

a. reachable with dangerous system calls,
b. reachable with non-dangerous system calls,
c. not reachable.

4.4 *Vulnerability analysis*

The vulnerability measurement, classification, prioritization, risk analysis and impact analysis are the common concerns of vulnerability analysis (figure 4). In this framework, the vulnerability analysis is carried out by automatically conducting the vulnerability assessment and exploiting the artefacts of penetration testing. The vulnerability analysis component of proposed integrated framework performs the following steps:

- apply the nature-inspired intelligent techniques to measure the severity and criticality level of vulnerability automatically using vulnerability and security metrics,
- automatically define, identify and classify the system resources and software vulnerability,
- automatically assign the importance and weight value to the resources and priorities to known software vulnerabilities,
- automatically identify the potential vulnerability along with associated risks, and store them in database,
- automatically perform the risk analysis and also analyse their consequences,
- design efficient and effective strategies for treating the severe vulnerabilities on priority basis using KBS,
- devise and implement approaches to automatically optimize the consequences or impact of vulnerability occurrence,
- automatically prepare the vulnerability disclosure report.

4.4a *Vulnerability assessment*: The vulnerability assessment component of proposed integrated framework employs the appropriate and efficient soft computing techniques to estimate the severity and criticality level of vulnerability and security level automatically using different vulnerability and security metrics. The vulnerability assessment can also be used to predict the effectiveness of security measures used for software vulnerability detection, mitigation and management. The assessment of

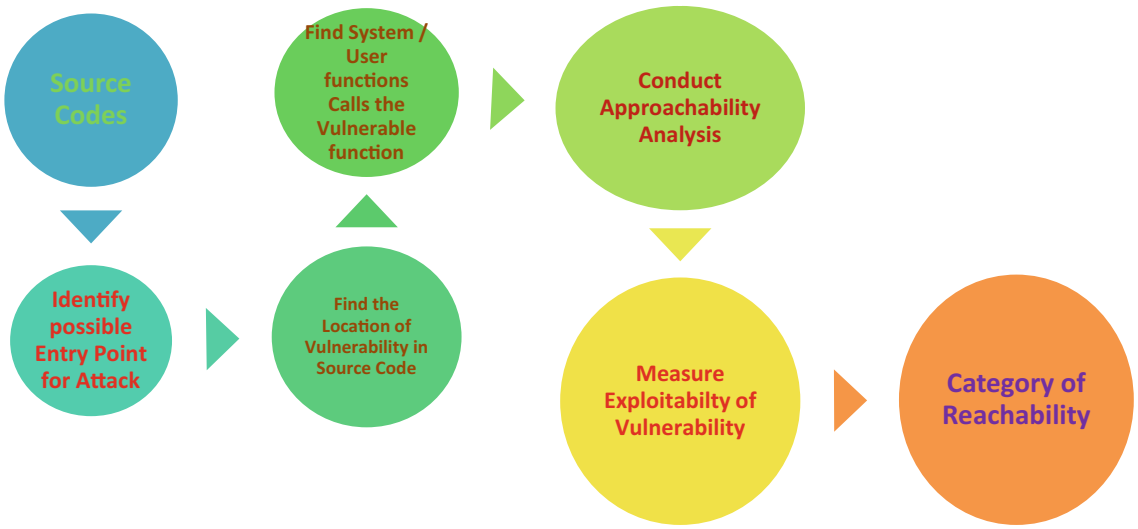

Figure 3. Finding the location of vulnerability in software system.

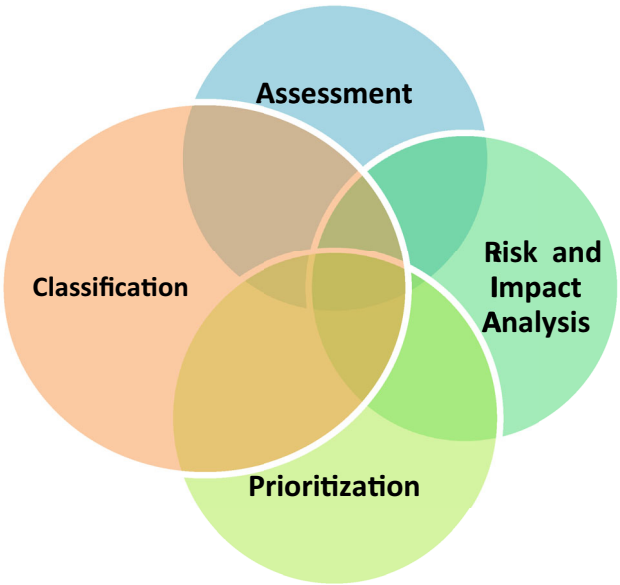

Figure 4. Vulnerability analysis.

vulnerability is based on multiple criteria. In this proposed framework, detective-, descriptive- and discovery-based approaches are used to carry out automatic assessment of vulnerabilities (figure 5). Advanced intelligent techniques such as artificial neural networks, genetic algorithm, grey wolf optimization, etc. are the most suitable for vulnerability assessment purpose and may be explored.

4.4b *Vulnerability classification*: For efficient and effective classification of vulnerabilities, the exploration and exploitation of descriptive-, detective- and discovery-based intelligent techniques along with automation of multifaceted vulnerability assessment are highly needed. It will also help in mitigation and management of software vulnerabilities. The automation of vulnerability assessment will increase the performance of classification, and also reduce the cost and effort of VMM.

In this framework, the computational intelligent techniques are used to automatically classify the vulnerabilities with the help of KBS. The vulnerability classification component of proposed framework exploits the artefacts of penetration testing, vulnerability occurrence data, data of

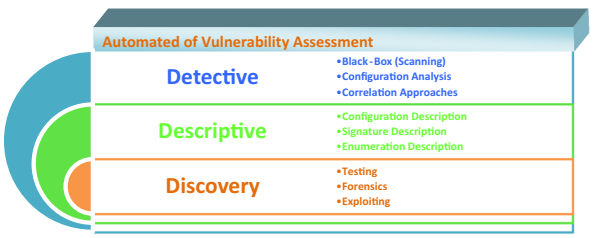

Figure 5. Automated software vulnerability assessment approaches.

current trends of vulnerability, data of recurring and exploitable historical vulnerability and their association with incidents and impacted assets stored in knowledge base. Since the fitness or assessment of vulnerability is a multifaceted concept, the intelligent classifier assigns a class to the known vulnerability using multifaceted fitness score. The fitness of vulnerability is measured using criticality, severity, loss incurred, business value of impacted asset, level of associated risks and performance data of vulnerability assessment techniques. The intelligent classifier automatically classifies the vulnerabilities into low, medium and high classes. The vulnerability analysis report is the input for the vulnerabilities classification approaches. The pair-wise analysis of attribute value is carried out for multifaceted vulnerability assessment and classification.

4.4c *Vulnerability prioritization*: All vulnerabilities are not equally important for remediation and mitigation. Some of them are severe in nature and have high consequences and impact on resources and software system itself also. In this framework, the vulnerability prioritization component automatically prioritizes the software vulnerabilities on the basis of their fitness score and class of belongingness. The fitness values of known software vulnerabilities are assessed using criticality, severity, loss incurred, business value of impacted asset and level of associated risks. The degree of belongingness of the vulnerabilities to different classes is computed using the multifaceted fitness score. The maximum value of the degree of belongingness of the vulnerability is considered to be the final class of the vulnerability. Immediate remedial and mitigation actions are to be carried out for the vulnerabilities of the high class, i.e., the most harmful vulnerability. In this framework, a list of assets, resources and critical software components is prepared for remediating actions. The components of software system having the most harmful vulnerabilities are identified as critical software components. These software components require immediate update through the latest version of patch of software components.

4.4d *Risk and impact analysis*: If the cost of software systems is high and they cater sensitive information and business decisions then it essential to know the risks, impacts and consequences of compromising with the vulnerability, the remediating actions to be performed for mitigation and management of vulnerabilities and when the remediating action would be taken. The organizations want to know risk, impact and consequences of occurrence of software on their business. Therefore, a careful analysis for risk, impact and consequences is required to fix errors and configure the software system. Hence, the analysis process must be automatically carried out minimize the errors in assessment.

In this framework, an automatic analysis for risk, impact and consequences is carried with the help of KBS. The proposed framework also determines the time frame for execution and completion of the remediating actions and

re-scanning also. The proposed framework also rings the tocsin for taking immediate remediating actions against the occurrence of high severity vulnerability also.

4.5 *Knowledge management*

The KBS is an integral component of proposed autonomic computing-based integrated framework. In this proposed framework, the KBS component is also used for sharing vulnerability information and remediating actions to self-scanner, intelligent vulnerability identifier and analyser and VMM components for effective identification of vulnerability, firing the trigger of alert of occurrence of vulnerability, vulnerability analysis and vulnerabilities remediation. The databases of vulnerability information, knowledge base and inference engine are the key sub-components of KBS of the proposed integrated framework. It uses the CBR and knowledge base to identify, estimate, classify, remediate, mitigate and manage software vulnerabilities, automatically.

In KBS, the database and knowledge base are used for capturing information about vulnerabilities, their remediating actions and time duration for completion of remediating, and a rule-based inference system is also developed. The information of security threats, historical and recurring vulnerabilities, detailed information of assets, IP address of hackers and attackers, the most vulnerable components in the software, loss due to vulnerability, severity and priority of vulnerability, etc. are stored in the database. The update in database is carried out as the changes occur in facts and information.

The knowledge base represents facts about the vulnerability world in the form of logical assertions and pre-conditions of occurrence of vulnerability, and inference rules. The inference rules are in the form of IF-THEN rules. In this framework, the KBS component is developed as an inbuilt integral part of software system for the following prominent reasons:

- To find out the unauthorized access or malevolent activity of software system by exploiting the known weakness points or vulnerabilities of software system.
- To determine the weak points or missing functionalities of software system automatically.
- To fire the trigger of alert for occurrence of vulnerability in software system after discovery of the pre-conditions of occurrence of the pre-existing flaws or known vulnerability.
- To update the database and knowledge base on occurrence of new types of software vulnerabilities.
- The normalized database of remedial actions taken against occurrence of vulnerabilities is designed and implemented. It is used to provide reference for remedial actions for future occurrence of vulnerabilities.
- After execution of remediating actions, a re-scanning of software system is performed by the self-monitoring component to verify the performance and functionality of software system.
- To measure the effectiveness of patch and vulnerability management after executing the autonomic corrective actions.
- To know the criticality and severity of the vulnerability and associated risks and the irreparable loss.
- To generate vulnerability disclosure report.
- To ensure the compliance of security requirement.
- To manage the patches efficiently.
- For exploration and exploitation of the artefacts of penetrating testing.

4.6 *VMM*

SVMM process is about discovering, classifying, remediating and mitigating software vulnerabilities. In this proposed framework, the remediating actions are determined by the KBS. The intelligent and automated vulnerability analyser identifies the vulnerability and associated risks, fires the trigger of alert and classifies and prioritizes vulnerabilities. The intelligent vulnerability analyser, with the help of KBS, also provides pointers to VMM component. The autonomic-feature-based VMM component selects an effective and feasible remedy for vulnerabilities from self-protection, self-configuration, self-healing and self-optimization. These remedial actions may use patch management, self-configuration and timeline framework.

During self-monitoring, the intelligent scanner notices whether any component of software system is not available or delaying in response. This delay in response should be recorded in database along with vulnerability for future reference. In future, such type of incident may occur again; for maintaining the stability or target security and performance, the alert component of integrated framework will immediately fire the trigger of alarm and direct the VMM component to take effective and feasible remedial action with the help of KBS.

The SVMM component of the proposed framework employs the CHOP properties of the ACS to manage the vulnerabilities. The SVMM component of the proposed framework uses knowledge base and inference engine to automatically select the appropriate and possible remediating actions on occurrence of software vulnerability in software system. The SVMM component also directs the self-executer or implementer to execute the directed and planned remediating actions. The possible and effective remediating actions are performed by the self-executer by executing self-configuration, self-healing, self-prevention and self-optimization operations as per the needs. The details of subcomponents of VMM component of proposed framework are as follows.

In this framework, the detective, the alert and the diagnostic components are used for adequate mitigation and

management of software vulnerabilities. The mitigation operations are performed by the source or binary codes of different subsystems of software. In this framework, an intelligent source code analyser is used to carry out automatic code analysis or review to eliminate the software flaws. For VMM, the self-managing activities such as self-protection, self-configuration, self-optimization and self-healing are carried out with the help of patch management, runtime libraries, bypassing the source codes that represent security flaws and eliminating the necessary pre-conditions of creating vulnerabilities.

4.6a *Self-protection*: The self-protection component of proposed framework automatically prevents the occurrence of vulnerabilities in software system. It is an integral component of software system. This component provides more secure software system to societies and organizations. The self-protection component of framework provides a shield to the software system from vulnerabilities and threats. The self-protection is runtime preventive mechanism of vulnerability mitigation. This component automatically prevents the occurrence of vulnerability and threats. The self-scanning component scans the software system for vulnerability. However, the self-protection component detects the necessary pre-conditions of occurrence vulnerabilities, and automatically takes immediate vulnerability remediating action to break or eliminate the necessary condition of vulnerability occurrence. The self-protection component of software automatically monitors and senses malicious behaviour of itself at runtime.

4.6b *Self-healing*: The patch management is one of the most vital steps of the vulnerability management. In patch management, time frame is important for managing the vulnerability. In this framework, the patching is carried out for maintaining the operational availability, confidentiality, performance and integrity of target software system. Since the developing, updating, selecting and execution of deployable patch package for target system is costly, time consuming and tedious activity, testing the patch before actual deployment and analysing post-implementation results is highly required for vulnerability management. In this framework, the intelligent self-healing component is used for automatic selection and timely installation of deployable patch package. The self-healing component of proposed integrated framework is an intelligent and autonomic component and used for patch management. The self-healing component automatically selects appropriate version of patch for target software. The self-healing component also directs the self-executer to copy or install the appropriate version of patch on impacted component of target software system.

4.6c *Self-configuration*: The self-configuration operation is carried out to prevent wrong results due to configurations setting errors. Some average-severity vulnerabilities change the environmental configuration setting, which mandates for smooth operation of software system. If the configuration settings of software systems are not compatible with operating environment then the operational availability, confidentiality, performance, reliability and integrity of target software system break. The self-configuration component of the proposed framework is used to automatically ensure configuration setting of operating environment of software system. The self-configuration component searches for deviations of configuration settings with compatible or initial configuration settings. It automatically measures the similarity or dissimilarity between the current configuration settings and the compatible or initial configuration settings. On the basis of dissimilarity value, the self-configuration component automatically fetches the appropriate configuration setting from the KBS and directs the self-executer to apply the configuration setting to target software system as per the direction of self-configuration component of proposed framework.

4.6d *Self-optimization*: The self-optimization component of proposed framework is used to maintain the performance of software systems in terms of efficiency, response time, waiting time, availability, reliability and so forth. The self-optimization-component-proposed framework updates or upgrades the software system for enhancing the performance of the software system.

4.7 *Self-effecter*

The self-executer component of integrated framework is responsible for automatic execution or implementation of remediating actions as directed by VMM component with the help of KBS as a result of detected vulnerabilities. The self-executer will follow the defined timeline framework while executing defined and planned directive remediating actions against detected vulnerability. If any problem occurs during the execution of defined remedial actions, it should be reported to KBS and VMM system by the self-executer. Therefore, the alternative remedial actions should also be defined and the self-executer will execute or implement the defined alternate remedial actions. The automatic verifier should track the execution of remedial actions and ensure its successful completion.

After successful implementation or execution of defined remediating actions, the self-monitoring component of integrated framework will rescan the software system. This rescanning will be carried out by an intelligent scanner. If the intelligent scanner reports the same problems before vulnerability detection, the follow-up actions should be taken automatically on the basis of CBR and KBS.

The proposed integrated framework will also ensure that after successful execution of remedial action, the software system has the identical configuration setting as it was before the occurrence of vulnerability. If the

software system does not have compatible or initial configuration setting, the proposed framework directs the self-configuration and self-optimization components to recover the initial configuration setting or update the current setting by compatible configuration setting to maintain the system performance. The self-optimization component updates the software for maintaining the performance of the software system. The self-configuration operation is carried out to prevent wrong results due to configurations setting errors and the self-optimization operation is carried out to maintain the performance of software systems in terms of efficiency, response time, waiting time, availability, reliability and so forth.

4.8 *Vulnerability management policy*

The vital step in vulnerability management is to define and fix the policy for vulnerability management. In this framework, the self-configuration, self-healing and self-protection are used for effective VMM. The remediating action of self-optimization is carried out to boost up the performance of software system. The self-managing and self-controlling activities are performed to determine the level of security of an organization by minimizing the level of vulnerability and maximizing the level security. The KBS component of proposed integrated framework is used to define the standards and guidelines for vulnerability level, security level and access control policy for connected components of proposed integrated framework. The proposed KBS helps in classifying vulnerabilities on the basis of severity of threats and level of risks. The time and effort for remediating the vulnerability are determined automatically on the basis of severity, criticality and associated risks and loss incurred due to occurrence of vulnerability. The allocated time frame for completion of remediating actions determines the schedule for re-scanning of software system.

5. Conclusion

The software industry has made several efforts to develop vulnerability-free software systems but failed in achieving the objective of developing software systems 100% free from vulnerabilities. Hence, software industry is looking for alternative adequate remediating approach to combat the security threats, to reduce security risks and irreparable loss, and to improve the performance of software system.

In this paper, an autonomic computing-based integrated framework is proposed to identify, analyse, classify, prioritize, analyse risks, impacts on assets and consequences, mitigate and manage the software vulnerability. The proposed framework is beneficial to industry and society in various aspects. Some of them are as follows:

- The proposed framework provides inbuilt self-managing ability to software systems to fight with security threats.
- Proposed framework is a cross-cutting-concern, platform-independent framework of integrated components of activities of VMM.
- It provides online automatic intelligent scanning and monitoring of source and binary codes of the software system for detection of the occurrence of vulnerability.
- It also provides an intelligent and automated assessment, classification and prioritization of vulnerabilities and assets.
- The proposed framework automatically conducts analysis for risks, impact on assets and consequences of software vulnerability on software systems and business.
- The proposed framework stores and updates the data and knowledge base, and also devises and evolves an inference engine for automatically executing the possible effective remediating actions within the time frame.
- It provides autonomic ability to software system through self-monitoring, self-configuration, self-healing, self-prevention, self-optimization and self-execution.
- The proposed framework will fulfil the security requirement of the organization and reduce the number of vulnerabilities incidents, cost, risks and loss.

6. Future scope

Though the proposed framework provides an inbuilt autonomic facility to existing and new software systems and is beneficial to the organizations and society, the commercial implementation of proposed integrated framework has not been carried out and is pending. For global usability and scalability, it should be developed as a cross-cutting concerns and platform-independent software using aspect-oriented or component-oriented software development paradigm.

References

[1] Agrawal A, Chandra S and Khan R A 2009 An efficient measurement of object oriented design vulnerability. In: *Proceedings of the International Conference on Availability, Reliability and Security*, ARES'09

[2] Dai H, Murphy C and Kaiser G 2010 Configuration fuzzing for software vulnerability detection. In: *Proceedings of the International Conference on Availability, Reliability and Security*, pp. 525–530

[3] Kumar M 2016 A paradigm shift towards incorporation and exploration of autonomic computing for software vulnerability

detection, mitigation and management. Communicated to *Informatica*

[4] Sharma A, Chauhan S and Grover P S 2011 Autonomic computing: paradigm shift for software development. *CSI Commun.* 35(6): 16–18

[5] Howard M and Lipner S 2003 Inside the Windows security push. *IEEE Security Privacy* 1(1): 57–61

[6] Alhazmi O H, Woo S W and Malaiya Y K 2006 Security vulnerability categories in major software systems. In: *Proceedings of Communication, Network, and Information Security 2006*, 138–143

[7] Bansiya J and Davis C G 2002 A hierarchical model for object-oriented design quality assessment. *IEEE Trans. Softw. Eng.* 28: 4–17

[8] Bansiya J 1997 *A hierarchical model for quality assessment of object-oriented designs.* PhD Thesis, University of Alabama in Huntsville

[9] Bishop M and Bailey D 1996 *A critical analysis of vulnerability taxonomies.* Technical Report, CSE-96-11, Department of Computer Science at the University of California at Davis, September

[10] Bishop M 2003 *Computer security: art and science*. Boston: Addison-Wesley

[11] Byres D and Shahmehri N 2007 Design of a process for software security. In: *Proceedings of the Second International Conference on Availability, Reliability and Security (ARES'07)*, 10–13 April, IEEE Press, Vienna, pp. 301–309

[12] McGraw G 2006 *Software security: building security.* Upper Saddle River, NJ: Addison-Wesley

[13] Meland P H and Jensen J 2008 Secure software design in practice. In: *Proceedings of the Third International Conference on Availability, Reliability and Security (ARES'09)*, 4–7 March, IEEE Press, Fukuoka, Japan, pp. 1164–1171, doi:10.1109/ARES.2008.48

[14] Walsh L 2003 Trustworthy yet? *Inf. Security Mag.* February

[15] Kannan K, Telang R and Xu H 2004 Economic analysis of the market for software vulnerability. In: *Proceedings of the 37th Hawali International Conference on System Sciences*

[16] Sahadeva K, Kumar Y S and Sharma A 2012 A new SDLC framework with autonomic computing elements. *Int. J. Comput. Appl.* 54(3): 17–23

[17] Gilliam D P, Wolfe T L, Sherif J S and Bishop M 2003 Softawre secuirty checklist for the software life cycle. In: *Proceedings of the Twelfth IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETIC'03)*, IEEE Press, Allahabad, India

[18] Ren J, Cai B, He H and Hu C 2011 A method for detecting software vulnerabilities based on clustering and model analyzing. *J. Comput. Inf. Syst.* 7(4): 1065–1073

[19] Wang Y, Wang Y and Ren J 2011 Software vulnerabilities detection using rapid density-based clustering. *J. Inf. Comput. Sci.* 8(14): 3295–3302

[20] Alves-Foss J and Barbosa S 1995 Assessing computer security vulnerability. *ACM SIGOPS Oper. Syst. Rev.* 29: 3–13

[21] Wang C and Wulf W A 1997 A framework for security measurement. In: *Proceedings of the National Information System Security Conference (NISSC'97)*, pp. 522–533

[22] Gilliam D, Powell J, Bishop J and Kelly M 2001 Reducing software security risk through an integrated approach. In: *Proceedings of the NASA IV&V Symposium, September 4–7*

[23] Butler S A 2002 Security attribute evaluation method: a cost–benefit approach. In: *Proceedings of the International Conference on Software Engineering (ICSE 2002)*, ACM Press, Orlando, Florida, pp. 232–240, doi:10.1145/581339.581370

[24] Halkidis S T Chatzigeorgiou A and Stephanides G 2004 A qualitative evaluation of security patterns. In: *Proceedings of ICICS 2004*, Lecture Notes in Computer Science 3269, Springer-Verlag, Málaga, Spain, pp. 132–144

[25] Hallberg J, Hunstad A and Peterson M 2005 A framework for system security assessment. In: *Proceedings of the 6th Annual IEEE System, Man and Cybernetics (SMC) Information Assurance Workshop*, IEEE Press, Budapest, pp. 224–231, doi:10.1109/IAW.2005.1495956

[26] Alhazmi O A, Malaiya Y K and Ray I 2005 Security vulnerabilities in software systems: a quantitative perspective. In: *Proceedings of Data and Applications Security 2005*, Lecture Notes in Computer Science 3654, pp. 281–294, doi:10.1007/11535706

[27] Chandra S, Khan R A and Agrawal A 2009 Security estimation framework: design phase perspective. In: *Proceedings of the Sixth International Conference on Information Technology, New Generations, ITNG 2009*, 27–29 April, IEEE Computer Society, pp. 254–259

[28] Chen Y, Boehm B and Sheppard L 2007 Value driven security threat modeling based on attack path analysis. In: *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, 3–6 January, IEEE Press, Big Island, Hawaii, p. 280

[29] Chowdhury I, Chan B and Zulkernine M 2008 Security metrics for source code structures. In: *Proceedings of the Fourth International Workshop on Software Engineering for Secure Systems*, Leipzig, Germany: ACM

[30] Alshammari B, Fidge C and Corney D 2009 Security metrics for object-oriented class designs. In: *Proceedings of the 9th Quality Software International Conference, QSIC '09*,, 24–25 August, pp. 11–20, doi:10.1109/QSIC.2009.11

[31] Agrawal A and Khan R A 2009 An algorithm to measure attribute vulnerability ratio of an object oriented design. *Int. J. Recent Trends Eng.* 2(3): 61–63

[32] Alkussayer A and Allen W H 2010 A scenario-based framework for the security evaluation of software architecture. In: *Proceedings of the International Conference on Computer Science and Information Technology (ICCSIT'10)*, 9–11 July, IEEE Press, Chengdu, China, pp. 687–695

[33] Alberts C, Allen J and Stoddard R 2010 *Integrated measurement and analysis framework for software security.* Technical Report, CMU/SEI-2010-TN-025, CERT® Program, Software Engineering Institute, http://www.sei.cmu.edu/library/abst

[34] Agrawal A and Khan R A 2010 A vulnerability metric for the design phase of object oriented software. In: *Proceedings of the Conference on Communications in Computer and Information Science*

[35] Agrawal A and Khan R A 2011 Assessing and improving encapsulation for minimizing vulnerability of an object oriented design. In: *Communication in Computer and Information Science*, vol. 250, Springer-Verlag, pp. 531–533

[36] Alhazmi O H, Malaiya Y K and Ray I 2015 Measuring, analyzing and predicting security vulnerabilities in software systems. *Comput. Security* 26(3): 219–228

[37] Khan K M and Jun Han J 2002 Composing security-aware software. *IEEE Softw.* 19(1): 34–41

[38] Wang J A, Wang H, Guo M, Zhou L and Camargo J 2010 Ranking attacks based on vulnerability analysis. In: *Proceedings of the 43rd Hawaii International Conference on System Sciences (HICSS'10)*, 5–8 January, IEEE Press, Hawai, USA, pp. 1–10

[39] Agrawal A and Khan R A 2011 A framework for vulnerability minimization—object oriented design perspective. In: *Proceedings of the 2nd International Conference on Computer and Communication Technology, ICCCT*, 15–17 September, IEEE Computer Society, pp. 499–504

[40] Agrawal A and Khan R A 2009 A framework to detect and analyze software vulnerabilities—development phase perspective. *Int. J. Recent Trends Eng.* 2(2): 82–84

[41] Ahuja K and Dangey H 2014 Autonomic computing: an emerging perspective and issues. In: *Proceedings of the International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT),* Ghaziabad, pp. 471–475