Mapping Software Security Metrics Concerning Design Principles

Sheikh Mustaq Ahmmed
Chair of Software Engineering: Dependability, TU Kaiserslautern
Kaiserslautern, Germany

***Abstract* - In today's advance technology, security problems become more important. Security measurement and monitoring helps system developers to design and assure secure systems. Today security metrics are used in a variety of fields as software development process. Security metrics for software products provide quantitative measurement for the degree of trustworthiness for software systems. Good metrics should be specific, measurable, repeatable and time defendant. Besides security plan to endure attack, software needs to be clearly blueprinted according to secure design principles. The method of this paper is to discuss the software security metric accompanying with design principles and ascertain metrics characteristics.**

***Index Terms* - software security, software security metrics, design principles**

I. INTRODUCTION

The expanding use of information system radically improve the functionality with respect to safety, cost and reliability. The growth of technology and vision of increased public access to computing, communication, and storage resources have made these systems more vulnerable to attacks. A system cannot be considered of high assurance if it has imperfect security. Security problems involving computers and software are frequent, widespread, and serious. In an era riddled with asymmetric cyber-attacks, claims about system reliability, integrity and safety must also include provisions for built-in security of the enabling software.

Security cannot be ensured entirely, but it can be assumed that the system is secured. Security remains more art than science, and nothing is more suggestive of this certainty. Software security is no exception: nearly every major business-critical application deployed today contains vulnerabilities, buffer overflow and cross-site scripting are commonplace, and so are many others. These harmful problems accidental and caused by insufficient awareness of the developers. So ensuring software security in today's world is a big challenge, as organizations worldwide increase their reliance on software controls to protect their computing environments and data, the term security grows in importance. The tremendous potential costs associated with security incidents, the emergence of increasingly complex regulations, and the continued operational costs associated with staying up-to-date with security patches all require that organizations give careful consideration to how they address software security.

Generally, software developed and implemented has bugs, and many of the bugs available within the software system have security implications. Security engineering as a discipline is still in its infancy. The field is hampered by its lack of adequate measures of goodness. Without such a measure, it is difficult to judge progress and it is particularly difficult to make engineering trade-off decisions when designing systems. A widely accepted management principle is that an activity cannot be managed if it cannot be measured. Software security also falls in this rubric [1]. Most of the security vulnerabilities in software are the result of security bugs or defects within the software. In most cases, these defects are created by two primary causes including non-conformance, or a failure in satisfying requirements, and an error or omission in the software requirements. Software security is a concept that still lack of unambiguous definitions. There are several methods for measuring software security, among them security metrics is promising and dynamic.

The standard term security metric is used for security measurement. Security metrics are important factors and are used in many applications such as architecture design and secure operations. These metrics can be used effectively in quality and security assurance applications. The main uses of security metrics can be summarized in three categories [1]:

- Management and strategic decisions such as program planning, resource allocation and service selection.
- Quality assurance in software development process and identifying program vulnerabilities and analyzing security flaws of the programs.
- Monitoring security status of IT systems and providing methods for security improvement.

In security engineering and software security assurance activities, e.g., testing, monitoring, analysis, the human audience of security metrics consists of most of the personnel associated with software security. The following roles have been identified as being important for software security [2]: security requirements developer, threat analyst, software architect, developer/programmer, tester, verifier, reviewer, auditor, application development manager, configuration manager and tool developer.

The main goal of this paper is to map the security metrics with design principles. The structure of this seminar paper is as follows: In section 2, I have discussed the background of software security, security metrics and design principles for secure software systems. In section 3, mapping security metrics with design principles and have discussed how the mapping is done with suitable examples. Finally I have concluded.

II. BACKGROUND

To comprehend the analysis completely basic terms need to be explained. In the following subsections a brief discussion over the key terms that are used will be provided.

Software Security

Software security is an idea implemented to protect software against malicious attack and other hacker risks so that the software continues to function correctly under such potential risks. Security is necessary to provide integrity, authentication and availability.

On the road to making such a fundamental change, we must first agree that software security is not security software. This is a subtle point often lost on development people who tend to focus on functionality. Obviously, there are security functions in the world, and most modern software includes security features. Software security is a system- wide issue that takes into account both security mechanisms (such as access control) and design for security (such as robust design that makes software attacks difficult). Sometimes these overlap, but often they don't.

The figure 1 [27] represents the basic security concept. The owners always have a desire to maximize the usefulness of the assets and to minimize the risks whereas; the attacker's main goal is to abuse the assets. Vulnerability is the intersection of three elements: a system susceptibility or flaw, attacker access to the flaw, and attacker capability to exploit the flaw. Owners always impose countermeasures to reduce the risks on assets, but attackers always gives rise to threats which increases the risks on assets. However, it is impossible to completely secure any system from attackers, but it is definitely possible to minimize them.

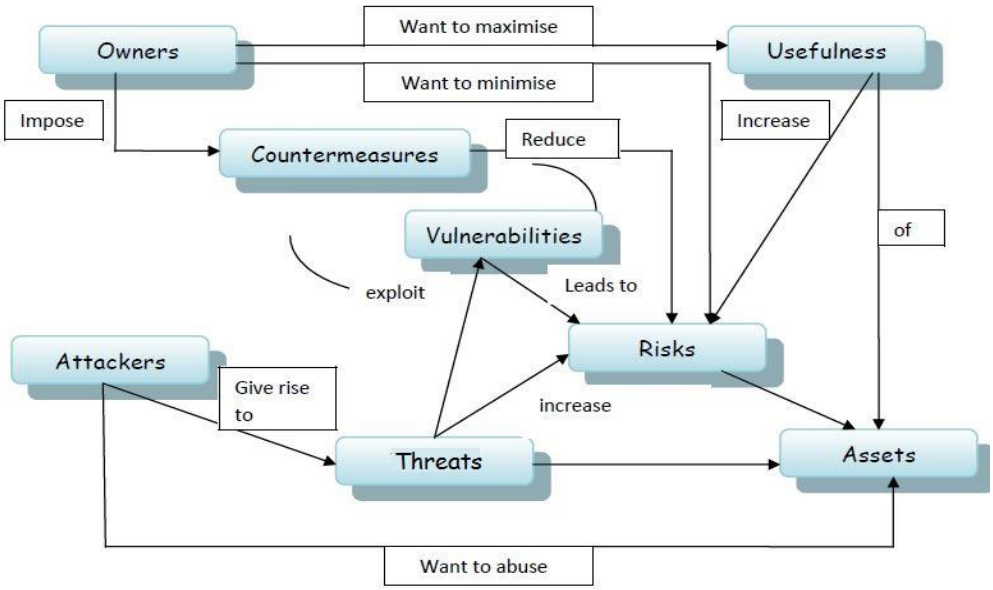

Figure 1: The concept of software security

Any compromise to integrity, authentication and availability makes a software unsecured. Software systems can be attacked to steal information, monitor content, introduce vulnerabilities and damage the behavior of software. Malware can cause DoS (denial of service) or crash the system itself. Buffer overflow, stack overflow, command injection and SQL injections are the most common attacks on software. Buffer and, stack overflow attacks overwrite the contents of the heap or stack respectively by writing extra bytes.

To avoid and make the software secure, have to code our system with good programming structure and also follow secure design principles as well.

Software Security Metrics Collection

Metrics is a term used to denote a measure based on a reference and involves at least two points: the measure and the reference. Security in its most basic meaning is the protection from or absence of danger. Literally, security metrics should tell us about the state or degree of safety relative to a reference point and what to do to avoid danger. Contemporary security metrics largely fail to do so. They tell us little about the actual degree of "safety" of our systems or processes, much less about the organization as a whole. They say little about the appropriate course of action, and they are typically not specific to the needs of the recipient.

Software security has become an enormous field. Though there are very few security metrics in the software industry to measure the software security, but it is a very good practice to use software security metrics for measuring effectiveness of a software. Researchers are still working into this field. Following are some of the software security metrics:

- **Relative Vulnerability Metric—** [3] This metric compares the calculated ratio of exploitable vulnerabilities detected in a system's software components when an intrusion prevention system (IPS) is present, against the same ratio calculated when the IPS is not present.
- **Static Analysis Tool Effectiveness Metric**—[4] The metric combines the actual number of flaws with the tool's false positive and false negative rates, and then weights the result according to the intended audience for the resulting measurements.
- **Relative Attack Surface Metric**—[5] The metric will define a mathematical model for calculating the attack surface of a system based on an entry point and exit point framework for defining the individual entry and exit points of a system. These entry and exit points contribute to the attack surface according to their accessibility, attack weight, damage potential, effort, and attack-ability.
- **Predictive Undiscovered Vulnerability Density Metric**—[6] This metrics is the extrapolation of Vulnerability Discovery Rate metrics. It gives measure to undiscovered or hypothetical vulnerabilities.
- **Flaw Severity and Severity-to-Complexity Metric**—[7][24] This metrics gives a rating reported software flaws as critical, high, medium, or low severity. It also determines whether it is possible to make a direct correlation between the number and severity of detected vulnerabilities and bugs and the complexity of the code that contains them.
- **Security Scoring Vector (S-vector) for Web Applications**—[8] This metrics is used to rate a web application's implementation against its requirements for technical capabilities, structural protection, procedural methods in order to produce an overall security score for the application [24]. Martin listed another set of metrics in his paper on software security evaluation based on a top-Down Mc Call-Like Approach [25][26].
- **Relative Attack Surface Metric**—[24] It is developed and used by Microsoft. This metric measures the attack ability of a system, i.e., the likelihood that an attack on the system will occur and be successful. It is calculated by finding the root attack vectors, which are features of the targeted system that positively or negatively affect its security.
- **Metric Associated with Software Security Properties**—[10] There are five properties found that are realistically measured: (1) smallness and simplicity; (2) separation of concerns; (3) defense in depth; (4) minimization of critical functions/components; and (5) accountability. In future research, the team plans to identify more measurable properties and to identify meaningful metrics that can be used for such measurements, e.g., the Goal Question Metric [11].
- **Metrics Associated With Security Patterns**—[12] Security metrics can be directly associated with software security patterns in order to measure the effectiveness of those patterns in securing the software system. It is likely that the metrics the team defines will measure effectiveness of those security functions in terms of policy enforcement or intrusion/compromise prevention

Design Principles for Secure Software

In software engineering secure by design means that the software has been designed from the ground up to be secured. Malicious practices are taken for granted and care is taken to minimize when a security vulnerability or on invalid user input appears. To protect the software system, it needs to be designed according to secure design principles. Design principles are a body of high- level design principles and decisions that allow a programmer to say "Yes" with confidence and "No" with certainty and also a framework for secure design, which embodies in microcosm the four classic stages of information security: protect, deter, detect, and react. Following points have been identified about design principles:

- Design is a complex, creative process.
- No standard technique to make design secure.
- But general rules derived from experience

Satlzer and Schroeder's The Protection of Information in Computer System identified a set of access control and other protection principles that should be applied when designing a secure system. From Saltzer and Schroeder [Saltzer 75], Section: "Basic Principles of Information Protection" following principles are found:

1. **Least Privilege** [15] The Least Privilege design principle requires a minimalistic approach to granting user access rights to specific information and tools. Additionally, access rights should be time based as to limit resources access bound to the time needed to complete necessary tasks. The implications of granting access beyond this scope will allow for unnecessary access and the potential for data to be updated out of the approved context. The assigning of access rights will limit system damaging attacks from users whether they are intentional or not.
   This principle attempts to limit data changes and prevents potential damage from occurring by accident or error by reducing the amount of potential interactions with a resource. So least privilege provides:
   - Mitigate Maleware Threats
   - Reduce Insiders Threats
   - Increase User Productivity
   - Achieve Compliance
   - Reduce Operating Cost

2. **Complete Meditation [15][29]** Every access to every object must be checked for authority. This principle, when systematically applied is the primary underpinning of the protection system. It forces a system-wide view of access control, which in addition to normal operation includes initialization, recovery, shutdown and maintenance. It implies that a foolproof

method of identifying the source of every request must be devised. It also requires that proposals to gain performance by remembering the result of an authority check be examined skeptically. If a change in authority occurs, such remembered result must be systematically updated. "This principle restricts the caching of information. This often leads to simpler implementations of mechanisms" [16].

3. **Fail -Safe Defaults** ``This principle calls for defaulting, in security relevant decision- making, towards denial rather than permission. Thus, the default situation will be to deny all requests that do not explicitly conform to the conditions under which such requests are permitted" [15]. However, following point should to be noted:
   - Unless a subject is given explicit access to an object, it should be denied access to that object.
   - If the subject is not able to complete its action/task, it should undo those changes it made in the security state of the system before it terminates. If the program fails the system is still safe.

4. **Separation of Privilege** [15][29] Where feasible, a protection mechanism that requires two keys to unlock it is more robust and flexible than one that allows access to the presenter of only a single key. The reason is that, once the mechanism is locked, the two keys can be physically separated and distinct programs, organizations, or individuals made responsible for them. From then on, no single accident, deception, or breach of trust is sufficient to compromise the protected information. This principle is often used in bank safe-deposit boxes. It is also at work in the defense system that fires a nuclear weapon only if two different people both give the correct command. In a computer system, separated keys apply to any situation in which two or more conditions must be met before access should be permitted.
   For example, systems providing user-extendible protected data types usually depend on separation of privilege for their implementation. In a word -**A system should not grant permission based on a single condition**.

5. **Open Design** "The Open Design Principle is a concept that the security of a system and its algorithms should not be dependent on secrecy of its design or implementation" [15].
   Security should not depend on security-through obscurity, i.e., the ignorance of potential attackers, but rather on assurance of dependability and/or the possession by users of specific, easily protected, authorization credentials. This permits the software to be examined by a number of reviewers without concern that the review may itself compromise the software's security. The practice of openly exposing one's design to scrutiny is not universally accepted. The notion that security should not depend on attacker ignorance is generally accepted, [17] but some would argue that obfuscation and hiding of both design and implementation has advantages: they raise the cost to the attacker of compromising the system.

6. **Economy of Mechanism**[15][29] The Economy of mechanism design principle requires that systems should be designed as simple and small as possible. Design and implementation errors result in unauthorized access to resources that would not be noticed during normal use. In this principle following points are important:
   - Security mechanism should be as simple as possible.
   - Fewer errors ; less checking and testing.
   - Interface to other module are particular suspect.

7. **Least Common Mechanism**"Minimize the amount of mechanism common to more than one user and depended on by all users" [15].
   Every shared mechanism (especially one involving shared variables) represents a potential information path between users and must be designed with great care to be sure it does not unintentionally compromise security. Further, any mechanism serving all users must be certified to the satisfaction of every user, a job presumably harder than satisfying only one or a few users.
   For example, given the choice of implementing a new function as a supervisor procedure shared by all users or as a library procedure that can be handled as though it were the user's own, choose the latter course. Then, if one or a few users are not satisfied with the level of certification of the function, they can provide a substitute or not use it at all. Either way, they can avoid being harmed by a mistake in it. Key points for this mechanism is given below:
   - Mechanisms used access resources should not be shared.
   - Because of limits sharing this principle also restrictive.

8. **Defense in Depth** — [15][28] The Defense in Depth design principle is a concept of layering resource access authorization verification in a system reduces the chance of a successful attack. This layered approach to resource authorization requires unauthorized users to circumvent each authorization attempt to gain access to a resource.
   When designing a system that requires meeting a security quality attribute architects need consider the scope of security needs and the minimum required security qualities. Not every system will need to use all of the basic security design principles but will use one or more in combination based on a company's and architect's threshold for system security because the existence of security in an application adds an additional layer to the overall system and can affect performance. That is why the definition of minimum security acceptably is need when a system is design because this quality attributes needs to be factored in with the other system quality attributes so that the system in question adheres to all qualities based on the priorities of the qualities.

9. **Psychological Acceptability** — [15][29][30] The Psychological Acceptability design principle refers to security mechanisms not make resources more difficult to access than if the security mechanisms were not present.

III. MAPPING OF SECURITY METRICS WITH DESIGN PRINCIPLES

In this section mapping has been done by considering the properties of design principles and software security metrics and I try to rationalize the mapping with suitable examples. Following table 1 provides a mapping between the design principles and the software security metrics where tick mark indicates which security metric provides a way to measure which design principle.

	Relative Vulnerability Metric	Static Analysis Tool Effectiveness Metric	Relative Attack Surface Metric	Predictive Undiscovered Vulnerability Density Metric	Security Scoring Vector(S-vector)for Web Applications	Flaw Severity and Severity-to-Complexity Metric	Metric Associated with Software Security Properties	Metrics Associated with security Patterns
Least Privilege	✓		✓					
Complete Mediation								✓
Fail-Safe Defaults							✓	
Separation of Privilege		✓						
Open Design	✓							
Recording Compromises					✓			
Defense in Depth				✓			✓	
Economy of Mechanism							✓	
Security-Aware Error and Exception Handling						✓		
Least Common Mechanism					✓			

Table 1: Mapping security metrics with design principles

Least Privilege and Relative Attack Surface Metric

Consider a UNIX operating system which has root and non root user. The UNIX operating system does not apply access controls to the user root. That user can terminate any process and read, write, or delete any file. Thus, users who create back-ups can also delete files. So for accessing the resource from UNIX operating system, attacker should connect to the system as a root user because according to least privilege principles non root user has some restrictions.

On the other hand Relative attack surface metric, a resource's access rights determine whether the attacker can use the resource in an attack. And to invoke the system's method, send (receive) data into(from) the system attacker need full access into the system. RASM consider only entry point attacks and exit point attacks, i.e., attacks that require the attacker to either send data into the system or receive data from the system. Hence the attacker uses a channel in an attack if the attacker can connect to the system through the channel. Similarly, the attacker uses a method in an attack if the attacker can send (receive) data items into (from) the system by invoking the method, and the attacker uses a data item in an attack if the attacker can send (receive) the data item into (from) the system.

A strong properties of relative attack surface metric is that it ensures the accessibility. According to the given example, we can say that with RASM we are able to measure security aspect which systems are maintained the least privilege principle.

Separation of Privilege and Static Analysis Tool Effectiveness Metric

On Berkeley-based versions of the UNIX operating system, users are not allowed to change from their account to the root account unless two conditions are met. The first is that the user knows the root password. The second is that the user is in the wheel group (the group with GID 0). Meeting either condition is not sufficient to acquire root access. Meeting both conditions is required. According to this situation, decision cannot be taken based on a single condition. In Static Analysis Tool Effectiveness Metric security is ensured by combining true positive and false positive rate, so here developers always consider two data set and

try to minimize the false positives. So SATEM method can measure the effectiveness of a system which maintain Separation of Privilege design principle.

Least Privilege and Relative Vulnerability Metric

Consider the sudo and visudo command in UNIX operating system. The two best advantages about using sudo command are: restricted privileges and logs of the actions taken by users. In Linux computers we need to use sudo to act as root. So sudo command strongly supports the least privilege principle.

On the other hand visudo command provide the user to access the user as a root without asking the root password. Once we enter visudo command, we will see something like figure 2:

```
# /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
#

Defaults        env_reset

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL) ALL
```

Figure 2: Output of *visudo* command

Here *root ALL = (ALL) ALL* means the root user can execute from ALL terminals, acting as ALL(any) and run ALL (any) command.

Relative Vulnerability Metric it's determined the vulnerability ratio in a software components when intrusion prevention system (IPS) is present and not present. According to the given example here are two situations, root password is needed for accessing, changing and updating to linux file systems and root password is not needed. RVM also deals with similar type of situation. So, we can measure the vulnerabilities of the UNIX operating system with the principle of RVM.

Complete Mediation and Metrics Associated with Security Patterns

When a UNIX process tries to read a file, the operating system determines if the process is allowed to read the file. If so, the process receives a file descriptor encoding the allowed access. Whenever the process wants to read the file, it presents the file descriptor to the kernel. The kernel then allows the access. If the owner of the file disallows the process permission to read the file after the file descriptor is issued, the kernel still allows access. This scheme violates the principle of complete mediation, because the second access is not checked. Metric Associated with Security Patterns associated with partitioned application, secure assertion, server sandbox which describes software security functionality. Here every time metric check the security pattern for effectiveness of security functions, so it's difficult to violate the principle of complete meditation technique.

Economy of Mechanism and Metrics Associated with Software Security Properties

The finger protocol transmits information about a user or system. Many client implementations assume that the server's response is well-formed. However, if an attacker were to create a server that generated an infinite stream of characters, and a finger client were to connect to it, the client would print all the characters. As a result, log files and disks could be filled up, resulting in a denial of service attack on the querying host. Security assurance is very simple here because it's happen for incorrect assumptions about the input to the client. Metrics Associated with Software Security Properties has five properties, smallness and simplicity is one of them. According to Economy of Mechanism we can measure the security with method of MASSP.

Defense-in-depth and Metrics Associated with Software Security Properties

An application with username/password-based authentication may not benefit from increasing the required password length from eight characters to 15 characters as the added complexity may result in users writing their passwords down, thus decreasing the overall security to the system; however, adding a smart-card requirement to authenticate to the application would enhance the security of the application by adding a complementary layer to the authentication process.

According to this, the system doesn't rely on a single solution. Use multiple complementary security products, so that a failure in one does not mean total industry. This might be firewall, antivirus or a strong authentication server. Now look at the properties Metric Associated with Software Security Properties, there we found this metric is directly deal with defense-in-depth principles and its maintain the GQM properties. Also MASSP suggests that never relay on single solution.

Least Common Mechanism and Security Scoring Vector (S- vector) for Web Applications

Suppose, a web site provides electronic commerce services for a major company. Attackers want to deprive the company of the revenue they obtain that web sites. They flood the site with messages, and tie up the electronic commerce services. Legitimate customers are unable to access the web site and, as a result, take their business elsewhere.

Here, the sharing of the internet with the attackers' sites caused the attack to succeed. The appropriate countermeasure would be to restrict the attackers for access the segment of the internet connected to the web site. Techniques to do this include proxy servers such as the Purdue SYN intermediary or traffic throttling. The former targets suspect connections; the latter reduces load on the relevant segment of the network indiscriminately.

Least Common Mechanism method suggests that access resources should not be shared, and the above example also maintains this properties. Most of the web site does not share their full resources. S-vector is used to secure web resources, i.e. it can handle the security issue which is followed least common mechanism design principles.

IV. CONCLUSION

With this technological advancement, security is a relatively new concept for many of the organizations. Within the security realm, quantify security is still a relatively new theme. Every software system has security vulnerabilities and risks to certain degrees. It is critical for software security researchers and practitioners to identify these security risks, assessing the probability of their occurrences and the damage they could cause, and then develop security policy and mechanism to prevent or reduce the potential damages from the exploits of those security vulnerabilities. There are no well-established processes or methods to measure software security. To a great extent, different organizations have developed and deployed their own methods in measurements.

Security Metrics provides a way to measure your security program. It facilitates collecting and documenting program status and reporting on the current situation and gap analysis and is also widely recognized that metrics are important to information security because metrics can be an effective tool for information security professionals to measure the security strength and levels of their systems, products, processes, and readiness to address security issues they are facing. Metrics can also help identify system vulnerabilities, providing guidance in prioritizing corrective actions, and raising the level of security awareness within the organization. With the knowledge of security metrics, an information security professional can answer typical questions like “Are we secure?” and “How secure are we?” in a formal and persuadable manner.

A security metric measures or assesses the extent to which a system meets its security objectives. Since meaningful quantitative security metrics are largely unavailable, the security community primarily uses qualitative metrics for security. So, my research goal is to find out the security metrics and map along with secure software design principles. The metrics that are written in this paper are general and can be customized in different applications and areas.

REFERENCES

[1] W.jansen, ``Directions in security metric research," National Institute of Standards and Technology, NISTIR 7564, 2009.

[2] Reijo M. Savola," A Security Metrics Taxonomization Model for Software-Intensive Systems," Journal of Information Processing Systems, 2009

[3] Crispin Cowan, ``Relative Vulnerability: An Empirical Assurance Metric", Presented at the 44th International Federation for Information Processing Working Group 10.4 Workshop on Measuring Assurance in Cyberspace (Monterey, CA, 25-29 June 2003)

[4] Brian Chess and Tsipenyuk Katrina, ``A Metric for Evaluating Static Analysis Tools", Presented at Metricon 1.0 (Vancouver, BC, Canada, 1 August 2006).

[5] Pratsuya Manadhata and Jeannette M. Wing (CMU), An Attack Surface Metric (Pittsburgh, PA: CMU, July 2005). Available from: \url{http://www.cs.cmu.edu/~wing/publications/CMU-CS-05-155.pdf}

[6] O.H. Alhazmi, Y. K. Malaiya, and I. Ray (Colorado State University), ``Security Vulnerabilities in Software Systems: a Quantitative Perspective," in Proceedings of the IFIP WG 11.3 Working Conference on Data and Applications Security, Storrs, CT, August 2005. Available from: \url{http://www.cs.colostate.edu/~malaiya/635/IFIP-10.pdf}

[7] Pravir Chandra, ``Code Metrics", Presented at Metricon 1.0 (Vancouver, BC, Canada, 1 August 2006).

[8] Russell R. Barton, William J. Hery, and Peng Liu (Pennsylvania State University), ``An S-vector for Web Application Security Management," working paper (Pennsylvania State University, University Park, PA, January 2004).

[9] John Murdoch (University of York), ``Security Measurement White Paper," vers. 3.0. Washington, DC: Practical Software and Systems Measurement [PSM] Safety and Security Technical Working Group, January 13, 2006.

[10] Riccardo Scandariato, Bart De Win, and Wouter Joosen (Catholic University of Leuven), ``Towards a Measuring Framework for Security Properties of Software," in Proceedings of the Second ACM Workshop on Quality of Protection, October 27–30, 2006.

[11] Victor R. Basili and Gianluigi Caldiera (University of Maryland), and H. Dieter Rombach (University of Kaiserslautern), The Goal Question Metric Approach (1994). Available from: \url{http://wwwagse.informatik.uni kl.de/pubs/repository/basili94b/encyclo.gqm.pdf}

[12] Thomas Heyman and Huygens Christophe, Catholic University of Leuven; ``Software Security Patterns and Risk", Presented at Metricon 1.0 (Vancouver, BC, Canada, 1 August 2006).

[13] McDermott, Attack-Potential-Based Survivability Modeling for High Consequence Systems, op. cit.

[14] A metric first proposed in F. Stevens, ``Validation of an Intrusion-Tolerant Information System Using Probabilistic Modeling"(MS thesis, University of Illinois, Urbana-Champaign, IL, 2004).Available from: \url{http://www.crhc.uiuc.edu/PERFORM/Papers/USAN_papers/04STE01.pdf}
[15] Saltzer, Jerome H. 1\& Schroeder, Michael D. "The Protection of Information in Computer Systems," 1278-1308. Proceedings of the IEEE 63,9. IEEE, September 1975.
[16] Bishop, Matt.Computer Security: Art and Science. Boston, MA: Addison-Wesley, 2003.
[17] The practice of openly exposing one's design and algorithms to scrutiny is widely accepted in the cryptographic community, with the exception of NSA and possibly some of NSA's counterparts in other countries.
[18] G. Agarwal, IT Security Metrics, 08Feb, 2008. Available from: \url{http://cobitexpert.com/index.php?itemid=3}
[19] A. J. A. Wang, Information Security Models and Metrics, 43rd ACM Southeast Conference, ACM, March 18-20 Kennesaw, GA, USA. pp. 178-184, 2005.
[20] G. Jelen, SSE-CMM Security Metrics. NIST and CSSPAB Workshop, Washington, D.C., June 2000.
[21] O. S. Saydjari, Is Risk a Good Security Metric? QoP'06, Alexandria, Virginia, USA. ACM 1-59593-553-3/06/0010, pp. 59-60, , October 30, 2006.
[22] S. C. Payne, A Guide to Security Metrics, SANS Institute Information Security Reading Room, June 2006.
[23] R. Savola, Towards a Security Metrics Taxonomy for the Information and Communication Technology Industry, International Conference on Software Engineering Advances(ICSEA 2007) 0-7695-2937-2/07,2007, IEEE.
[24] Software Security Assurance joint endeavor by Information Assurance Technology Analysis Center(IATAC) and Data and Analysis Center for software (DACS)- State- of – the Art Report(SOAR), July 31, 2007.
[25] S. Martin, Software Security Evaluation Based on a Top-Down Mc Call-Like Approach, IEEE 1988, pp. 414-418.
[26] D. B. Aredo, Metrics for Quantifying the Impacts of Monitoring on Security of Adaptive Distributed Systems, Master Thesis Proposal – II, December 2005.
[27] Gray McGraw, Software security, Security and Privacy Magazine, IEE, Vol 2, No. 2, pp. 80-83, 2004.
[28] Barnum, Sean. Gegick, Michael. (2005). Defense in Depth. Retrieved on August 28, 2011 from https://buildsecurityin.us-cert.gov/bsi/articles/knowledge \\ /principles/347-BSI.html.
[29] Bertino, Elisa. (2005). Design Principles for Security. Retrieved on August 28, 2011 from http://homes.cerias.purdue.edu/~bhargav/cs526/security-9.p
[30] Saltzer, Jerry. (2011). BASIC PRINCIPLES OF INFORMATION PROTECTION. Retrieved on August 28, 2011 from http://web.mit.edu/Saltzer/w\\ww/publications/protection/Basic.html