Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2010 Proceedings

Americas Conference on Information Systems (AMCIS)

8-2010

Re-examining the Information Systems Security Problem from a Systems Theory Perspective

Diana K. Young
University of Texas at San Antonio, Diana.young@utsa.edu

Wm. Arthur Conklin
University of Houston, waconklin@uh.edu

Glenn Dietrich
University of Texas at San Antonio, glenn.dietrich@utsa.edu

Re-examining the Information Systems Security Problem from a Systems Theory Perspective

Diana K. Young
University of Texas at San Antonio
Diana.young@utsa.edu

Wm. Arthur Conklin
University of Houston
waconklin@uh.edu

Glenn Dietrich
University of Texas at San Antonio
Glenn.Dietrich@utsa.edu

ABSTRACT

This theoretical paper discusses a recent shift in cyber attackers' interest away from traditional network and operating systems vulnerabilities and towards application level security flaws in end user systems. The authors argue that this shift signals a strong need to re-examine the way that security is addressed during the systems development process. Most of the systems development methodologies currently used do not contain formal processes for dealing with the interconnected complexity and risks associated with today's computing environments. Using systems theory as a theoretical lens, the fundamental processes of current systems development methodologies are analyzed and weaknesses in their ability to deal with these environmental factors are discussed. The authors then present a proposed holistic framework for integrating security into existing systems development methods. The paper concludes with a discussion of the need for more scholarly research in this area and suggestions for future research directions are offered.

Keywords

Information security, systems theory, systems development, systems development methodologies, secure information systems, secure development lifecycle

INTRODUCTION

The January 2010 disclosure that Google and two dozen other companies had been hacked and their source code penetrated, clearly demonstrated a shift in hackers' interest away from traditional network vulnerabilities and towards application level security flaws (MITRE 2010a; OWASP 2010; Zorz 2009). Research has shown that organizations have become much more vigilant at securing their networks (Richardson 2008). However, in 2009, the SANS Institute reported that "application vulnerabilities such as SQL injection and cross-site scripting flaws in open-source as well as custom-built applications accounted for more than 80% of the new vulnerabilities discovered" (SANS 2009a p. 1). Furthermore, these two application level security flaws were listed as the top two most critical vulnerabilities on both the Open Web Application Security Project's (OWASP) and the CWE/SANS's lists of the most dangerous application security risks (MITRE 2010a; OWASP 2010). These reports, along with the high number of application level security flaws on the Common Vulnerabilities and Exposures (CVE) listing, clearly indicate that companies have not been effective at designing and building secure software applications (MITRE 2010b).

The growing number of identified application level security flaws represents a strong signal that it is time to re-examine the way that security is addressed during the systems development process. While the interconnectivity of the Internet has drastically changed the environment in which applications operate, most of the systems development methodologies used to build applications have not been updated to include formal processes that address the security threats inherent in internetworked environments. Strong evidence exists demonstrating that poor software development practices can result in critical security incidents (Higgins 2009). Despite this, the relationship between software development and security has been historically overlooked in both research and practice (Baskerville 1993; Conklin and Dietrich 2005; Futcher and Solms 2008; Higgins 2009; Raghavan and Zhang 2009; SANS 2009b).

While formative steps have been made in relating security to specific software development tasks (Andriole 2009; Mouratidis, Weiss, and Giorgini 2006; Raghavan and Zhang 2009; Siponen, Baskerville, and Heikka 2006), questions remain as to whether or not these steps are being applied in practice. Our examination of the extant literature yielded a single empirical example of an organization that has incorporated security into their systems development processes. In 2004, Microsoft implemented the Secure Development Lifecycle process as a required and integral component of their overall systems development methodology (Howard and Lipner 2006). Over time, that change has resulted in significant reductions in the number of newly discovered vulnerabilities in Microsoft's software products (SANS 2007). This result indicates that formal integration of security related processes into traditional system development processes has the potential to positively influence the level of security inherent in the developed software products.

Our review of the existing literature further found that little research has focused on addressing security throughout the entire software development process from project initiation through system implementation. While some research efforts have focused on addressing security during the requirements engineering process (Fabian, Gurses, Heisel, Santen, and Schmidt 2010; Haley, Moffett, Laney, and Nuseibeh 2006; Zou and Pavlovski 2008) and others have focused on addressing security during the logical and physical design processes (Mouratidis et al. 2006; Siponen et al. 2006), we found a single article which discussed security in relation to the entire software development process. (Mouratidis and Giorgini 2007).

This, we believe, represents a gap in current software development methods. Systems theory has long emphasized that for open complex systems, the environment in which those systems operate can have a potent impact on their performance (Churchman 1968). In today's highly interconnected computing environments, the probability that an application will encounter nefarious usage is greatly heightened. As such, the software development processes used to create applications that will run in these environments should contain formal methods to ensure that the resulting system is able to detect and robustly respond to the security threats encountered in that environment.

The purpose of this theoretic paper is four fold. First we review the body of existing literature concerning systems development and security. Second, we examine the current systems development paradigm using systems theory as a theoretical lens to identify gaps in existing approaches. Third, we present a proposed a holistic framework that incorporates security into the overall software development process. Finally, we offer suggestions for future research directions.

LITERATURE REVIEW

Systems Development Methods

The terms systems development methodology, method, approach, framework, life cycle, and plan have all been used to refer to a formalized set of activities used to facilitate the management, production, and delivery of system development projects (Carugati 2008; Hackathorn and Karimi 1988; Stefanou 2003; Whitten and Bentley 2007; Wynekoop and Russo 1995). SDMs have traditionally been adopted by organizations to reduce the complexities and risks inherent in the systems development process (Stefanou 2003). While a wide variety of specific SDMs have been developed and implemented, most methods contain, at a minimum, steps for completing the requirements elicitation, design, development, and testing phases of a systems development project (Baskerville 1993; Spence and Bittner 2005; Whitten and Bentley 2007; Xu and Brinkkemper 2007). Definitions of these phases are presented in Table 1 below.

Software Development Phase	Phase Definition
Requirements Elicitation	A process of studying a specific business problem domain in order to identify the requirements of a solution to the problem
Design	A process consisting of the translation of gathered requirements into potential solutions, selecting the optimal solution, and then specify the technical specification for the selected solution to the problem
Development	A process of transforming the technical specifications of a solution into a set of working system components that together represent the physical instantiation of a solution to the problem
Testing	A process of validating that the developed system solution satisfies the specified requirements specified for the business problem

Table 1. Software Development Phase Definitions
Adapted from (Whitten and Bentley 2007)

A plethora of SDMs have been developed and implemented in practice (Wernick and Hall 2004). To date, the superiority of one approach over another has not been established. In fact, most researchers and practitioners agree that different SDMs may be appropriate for different types of projects (Karlsson and Agerfalk 2009). Furthermore, little empirical work has been done to measure the benefits derived from using one particular SDM over another during a system development project (Wynekoop and Russo 1997). Researchers have, though, identified two main categories of SDMs: sequential methods and incremental methods (Xu and Brinkkemper 2007).

Sequential methods evolved out of the SDLC concept that was first introduced in 1971 by Daniels and Yeates (Stefanou 2003). These methods are sometimes referred to as following the Waterfall Model as they divide an entire systems development project into a series of highly structured phases to be completed sequentially. These methods focus heavily on the use of documentation and functional decomposition to facilitate management of the development effort. Incremental SDM approaches also contain processes for completing the requirements elicitation, design, development, and testing phases of a project. However, they differ fundamentally from sequential approaches in the scope of work addressed during a project phase. Whereas sequential methods deal with an entire system during each project phase, incremental methods focus on the development and delivery of only one increment of a system during each phase.

Security and Systems Development

Interest in security has increased dramatically in the IS research domain since the 1970s. The number of security related articles in a set of nine high ranking IS research journals increased from three during the years 1970 - 1979 to 100 during the years 2000 – 2007 (Zafar and Clark 2009). The subject range of these articles was quite diverse with nearly 80% of the discourse relating to security governance, integrity of data, privacy, and threat mitigation issues. Quite notable in the study, though, was the lack of research concerning the relation between security and systems development activities. Of the 137 articles included in Zafar and Clark's study, only four articles related security to development activities and only Tryfonas (2007) discussed the need to incorporate security requirements into existing systems development processes.

Baskerville (1993) reviewed the state of security analysis and design practices by comparing and contrasting improvements in the field of security to the evolution of the methods used in systems development. In his analysis, he noted that general systems development methodologies had evolved through three distinct generations, with each successive generation improving and augmenting known best practices. The first generation, termed checklist methods, is characterized by the mapping of pre-defined sets of solutions to functional problems. The second generation, termed the mechanistic engineering, is characterized by the partitioning a systems' functional requirements into small manageable pieces and the development of independent solutions for each partition. Finally, the third generation, termed logical transformation, is characterized by the use of abstract problem and solution modeling to logically explore the problem space and select the best holistic solution to the problem. Baskerville noted that a key benefit of third generation methods was their intense focus on conceptually modeling the system design prior construction whereas previous generations were rooted entirely in the physical instantiation

of the system. He felt that this difference provided developers using third generation methods a strong link between the stated requirements of the system and the design specifications of the system to be built. This link, he argued, resulted in systems which were more likely to fit users' expectations.

Using the taxonomy described above, Baskerville (1993) contrasted the state of software development processes with the state of security development processes and argued that the latter had not progressed past the mechanistic engineering phase of development. He noted that the use of checklists had characterized most early security practices and that some firms did meet the requirements of the mechanistic engineering phase through their use of specific risk analysis, disaster recovery, and network protection activities. However, he noted that only formative progress had been made in moving security analysis and design practices to the logical transformation phase of systems development. He felt this lack of progress was due to a strong disconnect between overall systems development processes and security design processes. On that issue, Baskerville stated that, "those responsible for researching systems development methods seem to assume security is a separate issue: an implementation or computer science problem" (1993 p. 379). He argued that progress will not be made in securing systems until security is holistically incorporated into the systems development process as an explicit and fundamental objective of that process.

Conklin and Dietrich (2005) also noted that security is often neglected in the design and development of systems. They discussed the efficacy of four strategies for implementing security in software systems. The strategies discussed range from augmentation (adding security features after the core software functionality has been developed and implemented) to integration (building security in as a specified requirement of the system). The authors used systems theory to analyze the strengths and weaknesses of each approach. They noted that augmentation is often selected as the ideal means for adding security functionality to software after the module has been deployed. However, they pointed out that this approach often makes it possible for users to shut off or simply navigate around the added security functionality. With integration, the authors explain, users cannot bypass security restrictions; security is treated as any other functional requirement of the systems and is incorporated in the analysis, design, construction, and deployment of the system so that users are only able to traverse secure paths within the system. This approach, they argued, is the most effective of the four approaches but it can also negatively impact the cost and time necessary to build and deliver system solutions.

Security and Requirements Engineering

Within the IS domain, security has generally been characterized as a non-functional system feature. Due to this, little research in the discipline has focused on best practices for defining security requirements during the development process. In Zafar and Clark's (2009) review of security articles in IS journals, Tryfonas (2007) was the only article they found that specifically related security to the requirements elicitation phase of the software development process.

The topic of security requirements has received more attention in both the computer science and software engineering disciplines. Haley et. al. (2006) defined security requirements as the implementation of security goals which constrains the functionality of a system. Using this definition, Fabian et. al. (2010) noted that security requirements often conflict with the stated functional and non-functional requirements of a system. To resolve such conflicts, they presented a framework for gathering, analyzing, and reconciling the functional, non-functional, and security requirements of a system. While their framework acknowledges that the resulting security requirements can impact subsequent phases of systems development efforts, they do not specifically address how security should be incorporated into those phases. Fabian et. al.'s framework for secure requirements engineering is presented below as Figure 1.

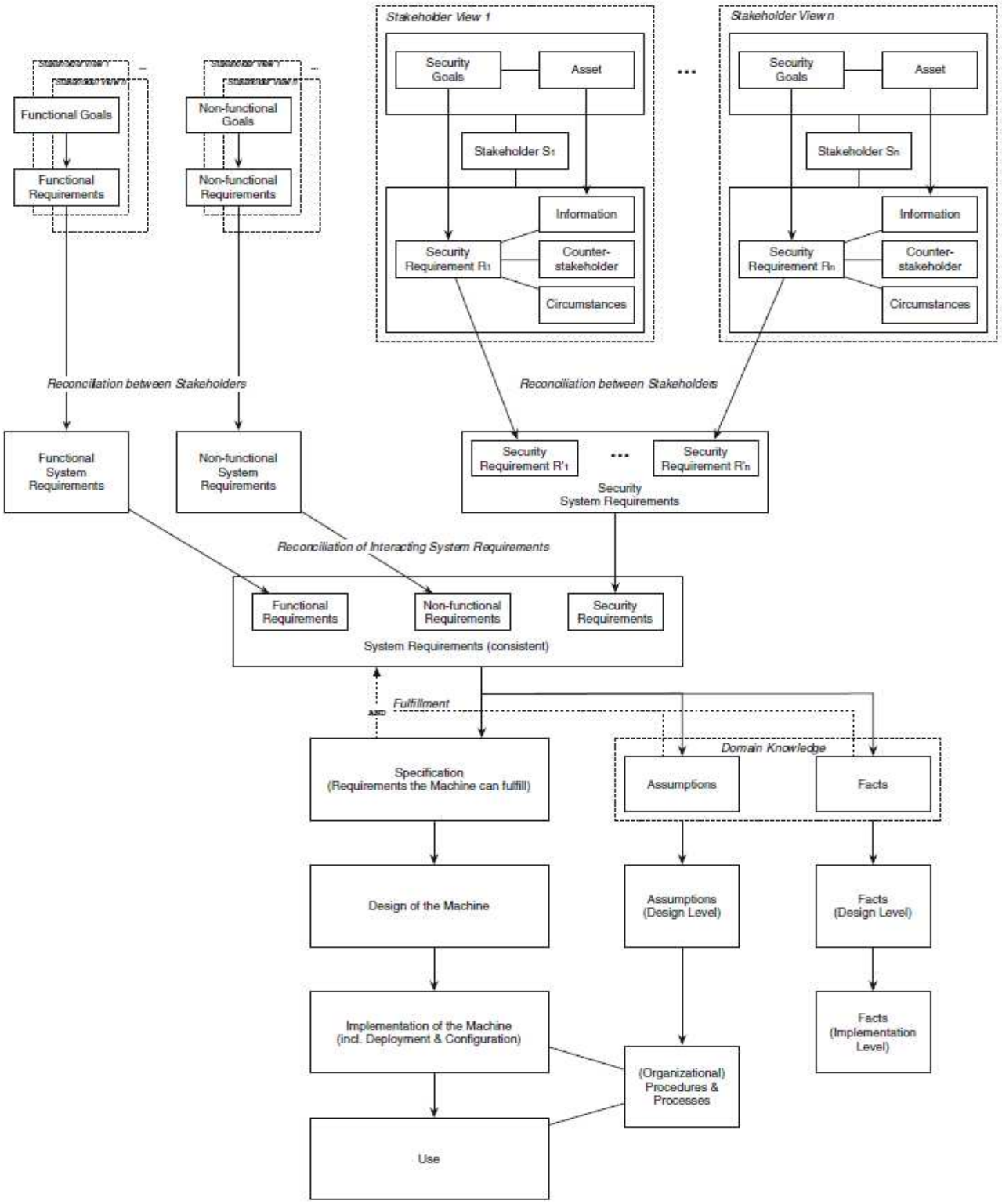

Figure 1. Fabian et. al's (2010) Secure Requirements Engineering Framework

Security and Systems Design

A few stand-alone secure design methods have been proposed which offer suggestions for modeling security problems and solutions during the design phase of a systems development project (Mouratidis, Weiss, and Giorginni 2006, Siponen, Baskerville, & Heikka 2006, and Tryfonas 2007). However, these proposed methods have received little attention in the literature. Siponen et. al. (2006) discussed the need for an adaptable, secure information systems (SIS) design method that was able to integrate with a wide range of SDMs. Using design theory, they put forth six theoretical meta-requirements for SIS methods. These meta-requirements specify that security design methods must focus on securing systems objects by gathering specific organizational system level security requirements and then designing solutions to those requirements using logical modeling tools. Furthermore the meta-requirements state that a design method must be able to easily integrate with whatever SDM the development team chooses to follow for a specific project. Table 2 lists the six meta-requirements discussed in the article.

Meta Requirement	Meta Requirement Description
1	SIS design methods must focus on developing security features to resist threats to system objects. This requires that SIS methods provide mechanisms to systematically identifying objects to be secured, threats to those identified objects, and features to mitigate such threats.
2	SIS design methods must provide processes and notation for collecting, documenting, and prioritizing organizational level security requirements. Each organization operates in a unique environment and as such will have differing objects to protect and different threats to those objects.
3	SIS design methods must provide tools to abstractly represent and operationalize the threats, objects, and security features of a system at the organization, conceptual, and technical levels.
4	SIS design methods must be able to integrate easily with all ISD methods.
5	SIS design methods must allow developers to choose the ISD method which best suits a particular systems development project
6	SIS design methods must be able to adapt to forthcoming ISD methods

Table 2. SIS Design Method Meta-Requirements

adapted from Siponen et. al. (2006)

Siponen et. al. (2006) further analyzed the extent to which existing SIS methods met the six meta-requirements and found that most met only two or few of the meta-requirements. Based on this, they advanced the Meta-notation method for SIS design which met all six of the stated meta-requirements. They then tested the process using action research. In a similar vein, Mouratidis et al. (2006) used action research to demonstrate how agent orientation, security patterns, and the Tropos methodology could be combined to document and model given security requirements for a systems development project. However, questions remain as to whether or not any of these methods have been widely adopted in practice.

Systems Theory and Systems Development Methods

Systems theory was first proposed in the 1950s as an interdisciplinary means to study the arrangement of parts that together form a whole. In explaining the theory, Bertalanffy stated, "There exist therefore general system laws which apply to any system of a certain type, irrespective of the particular properties of the system or the elements involved" (Bertalanffy 1950 pp. 138). The theory diverged from previous scientific philosophies which focused entirely on decomposing systems into respective parts and then studying the parts independently. Systems theory views a system from a holistic perspective, recognizing that the whole of the system is often greater than the sum of the independent parts. The theory has been widely applied to study complex systems in the natural sciences, social sciences, engineering, technology, and business disciplines.

The fundamental construct of systems theory is the system, which Bertalanffy described as any set of components that interacted among themselves to achieve a goal (Bertalanffy 1950). According to the theory, all systems receive inputs, produce outputs, and operate in an environment. Figure 2 below depicts a simple system and its fundamental components.

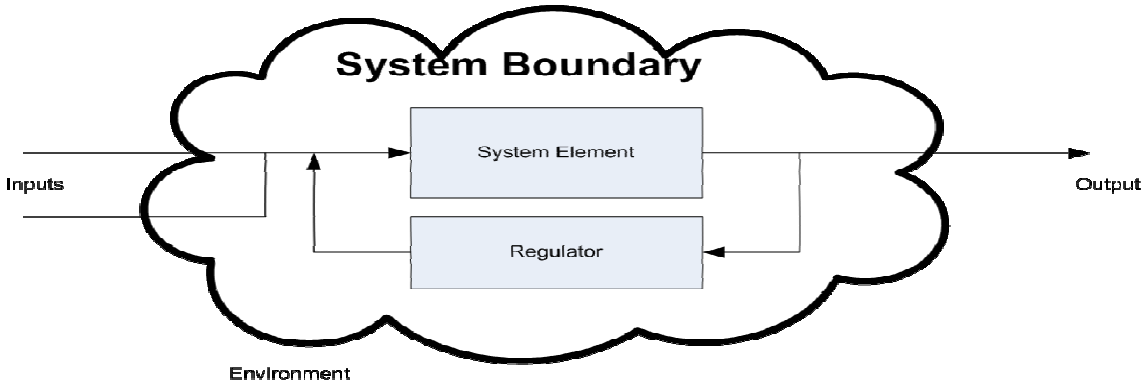

Figure 2. Systems Theory View of a Simple System

Churchman (1968) proposed several critical considerations that should be kept in mind when thinking about a system in terms of systems theory. Of foremost importance, he argued, was to think of the system in holistic terms and understand the totality of its objectives. Furthermore, he believed that every problem a complex system encountered was inextricably linked to its environment and to study one without an appreciation of the other would lead to incorrect conclusions. Finally, Churchman believed that for a system to successfully achieve its objectives, those objectives had to be translated into concrete measures and those measures used to evaluate the systems' effectiveness at achieving its overall objectives.

When examining the current systems development paradigm in terms of Churchman's critical considerations, we believe several gaps in current systems development practices concerning security are illuminated. First, most SDMs do not promote a holistic perspective of the target system during the requirements elicitation process but focus exclusively on the functionality specified by system stakeholders. To think of a system holistically and consider the totality of its objectives during the development process would necessitate early examination of the environment in which the system will operate and consideration of the potential threats that may be encounter in that environment.

Unfortunately, responsibility for IT security is generally not in the domain of individual stakeholders but relegated to a specific team of IT security experts. As such, current SDMs are not designed to include security concerns in the requirements gathering process. If formal methods for eliciting security requirements were incorporated into existing SDMs, we believe one of the fundamental requirements specified for every Internet facing application would be the ability to sense and appropriately respond to security threats such as SQL injection and cross site scripting.

The second gap illuminated concerns Churchman's (1968) recommendation that all system objective to be converted into concrete measures that can be used to evaluate a system's effectiveness at achieving its stated objectives. Traditional system development processes achieve this through their use of requirements to drive subsequent phases of development. For instance, most SDMs contain quality checkpoints where the outputs of a phase are compared to the requirements to ensure completeness before proceeding forward. Unfortunately, as existing SDMs do not formally address security during the requirements elicitation process, projects following these methods do not have objective means available to evaluate the level of security in the system throughout the design, development, and testing phases of the project.

To rectify these problems, we believe existing SDMs must be updated to incorporate formal processes which address security throughout the entire systems development process. First, the requirements elicitation process must be expanded to specifically examine the environment the system will operate in and determine the system's potential for encountering security threats. The appropriate responses to those threats must then be recorded as security requirements for the system. This expansion of the requirements elicitation process must further contain processes such as those proposed by Fabian et. al. (2010) to rectify conflicts between the state security and functional requirements of the system.

Second, the solution design processes within existing SDMs must be expanded to include process such as those proposed by Mouratidis et. al. (2006), Siponen et. al. (2006), and Tryfonas (2007) to fully integrate the security requirements gathered into the overall design of the developed system. Finally, during the system and acceptance testing phases of the project, security requirements must be translated into specific test conditions that can be used to objectively evaluate the level of security in the system. Our proposed holistic Secure Systems Development framework is presented below in Figure 3.

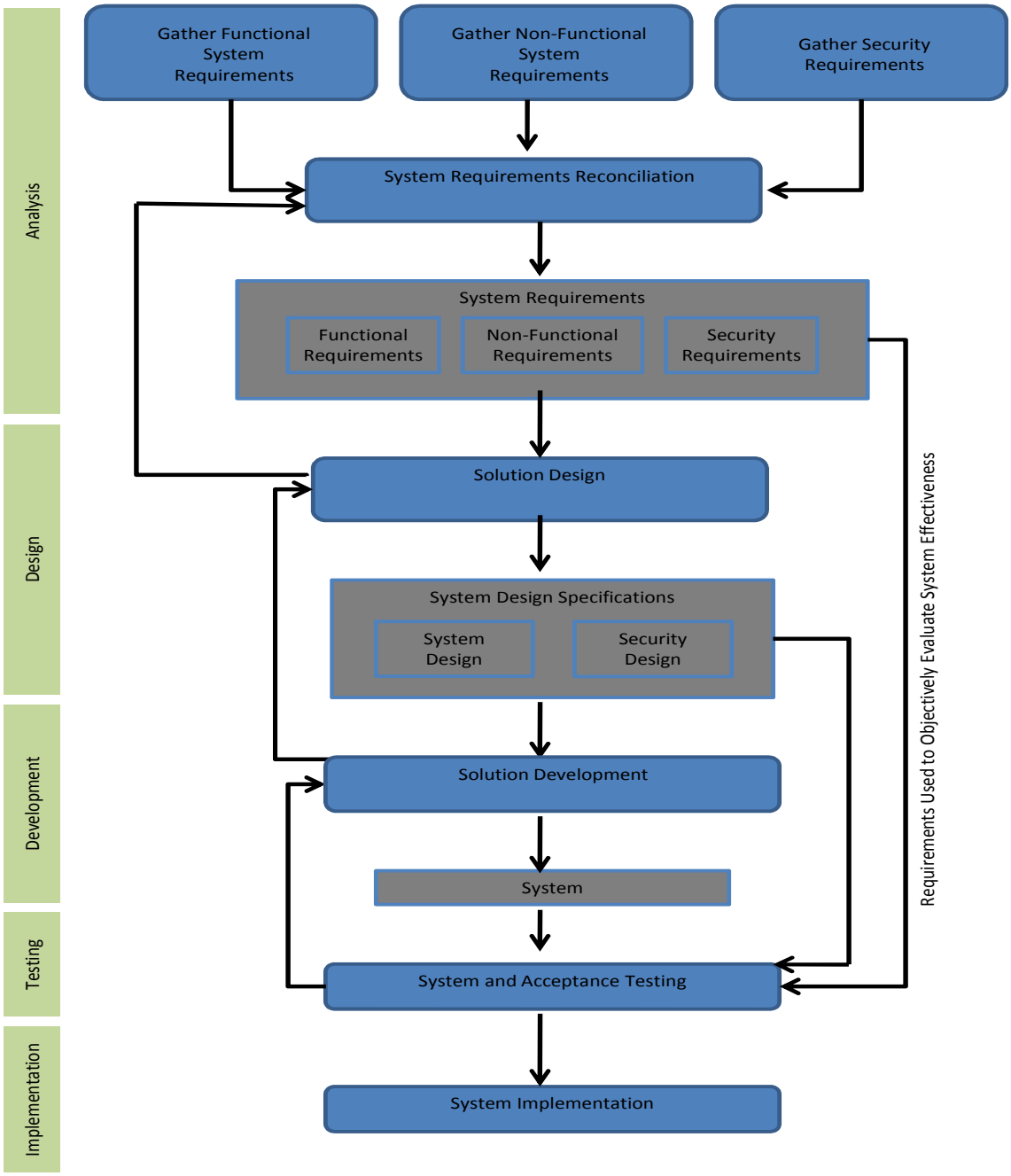

Figure 3. Secure Systems Development Framework

FUTURE RESEARCH DIRECTIONS

Several avenues of research are needed to move this significant issue forward. First, more precise empirical research is necessary to determine if organization are aware of this problem and how exactly they are addressing security concerns in application systems. While the high number of application level security flaws reported seems to indicate that organization

are not addressing security during the systems development process, we could not find empirical evidence for this supposition. Thus, more in-depth organizational level empirical research is needed.

Second, both theoretical and empirical research is needed to determine the optimal methods of incorporating security related processes into existing sequential and iterative software development methods. While several processes for relating security to specific software development tasks have been proposed in the literature, research is needed to measure the impact of their use on the overall systems development process and on the security level of the resulting systems. Further, the costs, benefits, and consequences of incorporating these processes into the overall systems development process need to be explored and measured.

Finally, education related research is necessary. Traditionally, security and software development have been treated as completely separate fields in the IS discipline. The findings discussed above demonstrate that there is overlap between the fields. Software development methodologies and practices need to be incorporated into security training and conversely, security issues need to be incorporated into software development training. Finally, the academic community needs to assess current graduate and undergraduate curriculums and make recommendations as to how this important topic should be incorporated into higher educational programs.

CONCLUSIONS

The 2010 announcement that Google and several other technology related organizations had been hacked and their source code penetrated brought to the forefront a simultaneously interesting and frightening fact. Cyber attackers' interest has shifted away from traditional organizational level network and operating systems vulnerabilities and toward application level security flaws in end-users systems. These events represent a very strong signal that it is time to re-examine the way that information systems are designed and built.

Systems theory has long stressed the impact that environmental variables can have on a system's performance. In today's highly interconnected computing environment, non-legitimate use of applications is a standard part of that operating environment and by all indications is linked to the high number of application level security vulnerabilities being reported. To address this problem we believe security processes need to be formally incorporated into existing software development processes. While formative steps have been made at relating security to specific software development tasks, our review of the existing literature found no research that focused on holistically addressing security throughout the entire software development process. To fill this gap, we proposed a secure systems development framework which formally incorporates security into all phases of the systems development process.

REFERENCES

1. Andriole, S.J. "Boards of Directors and Technology Governance: The Surprising State of the Practice," *Communications of AIS* (24) 2009, pp 373-394.
2. Baskerville, R. "Information systems security design methods: implications for information systems development," *ACM Comput. Surv.* (25:4) 1993, pp 375-414.
3. Bertalanffy, L.v. "An Outline of General Systems Theory," *British Journal for the Philosopy of Science* (I:2) 1950, pp 134-165.
4. Carugati, A. "Information systems development activities and inquiring systems: an integrating framework," *European Journal of Information Systems* (17:2) 2008, pp 143-155.
5. Churchman, C.W. *The Systems Approach* Dell Publishing, New York, NY, 1968.
6. Conklin, A.and Dietrich, G. "Secure Software Design Principles: A Systems Approach," Eleventh Americas Conference on Information Systems, Association of Information Systems, Omaha, NE, USA, 2005, pp. 3171 - 3180.
7. Fabian, B., Gurses, S., Heisel, M., Santen, T., and Schmidt, H. "A comparison of security requirements engineering methods," *Requirements Engineering* (15) 2010, pp 7-40.
8. Futcher, L.and Solms, R.v. "Guidelines for Secure Software Development " South African Institute of Computer Scientist and Information Technologists, ACM, Wilderness Beach Hotel, Wilderness, South Africa, 2008, pp. 56 - 64.

9. Hackathorn, R.D.and Karimi, J. "A Framework for Comparing Information Engineering Methods," *MIS Quarterly* (12:2) 1988, pp 203-221.
10. Haley, C.B., Moffett, J.D., Laney, R., and Nuseibeh, B. "A framework for security requirements engineering," SESS '06, ACM, Shanghai, China, 2006, pp. 35-41.
11. Higgins, K.J. "Mega-Breaches Employed Familiar, Preventable Attacks," in: *Information Week Analytics*, Informationweek.com, 2009.
12. Howard, M.and Lipner, S. *The Security Development Livecycle* Microsoft Press, Redmond, Washington, 2006.
13. Karlsson, F.and Agerfalk, P. "Exploring agile values in method configuration," *European Journal of Information Systems* (18) 2009, pp 300-316.
14. MITRE "2010 CWE/SANS Top 25 Most Dangerous Programming Errors," The MITRE Corporation, 2010a.
15. MITRE, C. "Common Vulnerabilites and Exposures ", C.E. Board (ed.), 2010b.
16. Mouratidis, H.and Giorgini, P. "SECURE TROPOS::: A SECURITY-ORIENTED EXTENSION OF THE TROPOS METHODOLOGY," *International Journal of Software Engineering & Knowledge Engineering* (17:2) 2007, pp 285-309.
17. Mouratidis, H., Weiss, M., and Giorgini, P. "MODELING SECURE SYSTEMS USING AN AGENT-ORIENTED APPROACH AND SECURITY PATTERNS," *International Journal of Software Engineering & Knowledge Engineering* (16:3) 2006, pp 471-498.
18. OWASP "OWASP Top 10 - 2010 rcl: The Ten Most Critical Web Application Security Risks," in: *The Open Web Application Security Project*, 2010.
19. Raghavan, V.and Zhang, X. "Building Security in during Information Systems Development," Americas Conference on Information Systems San Francisco, CA, USA, 2009, pp. 1 - 8.
20. Richardson, R. "2008 CSI Computer Crime Security Survey,") 2008.
21. SANS "SANS Top-20 2007 Security Risks," SANS Institute
22. SANS "The Top Cyber Security Risks," SANS Institute, 2009a.
23. SANS "Twenty Critical Controls for Effective Cyber Defense: Consensus Audit Guidelines," 2009b.
24. Siponen, M., Baskerville, R., and Heikka, J. "A Design Theory for Secure Information Systems Design Methods," *Journal of the Association for Information Systems* (7:11) 2006, pp 725-770.
25. Spence, I.and Bittner, K. "What is iterative development?," IBM developerWorks, 2005.
26. Stefanou, C.J. "System Development Life Cycle," in: *Encyclopedia of Information Systems*, Elsevier Science, 2003, pp. 329-344.
27. Tryfonas, T. "On Security Metaphors and how they Shape the Emerging Practice of Secure Information Systems Development," *Journal of Information Systems Security* (3:3) 2007, pp 21 - 50
28. Wernick, P.and Hall, T. "Can Thomas Kuhn's paradigms help us understand software engineering?," *European Journal of Information Systems* (13) 2004, pp 235-243.
29. Whitten, J.L.and Bentley, L.D. *Systems Analysis & Design Methods 7th Edition* McGraw-Hill, 2007.
30. Wynekoop, J.L.and Russo, N.L. "Systems development methodologies: unanswered questions," *Journal of Information Technology (Routledge, Ltd.)* (10:2) 1995, p 65.
31. Wynekoop, J.L.and Russo, N.L. "Studying system development methodologies: an examination of research methods," *Information Systems Journal* (7:1) 1997, pp 47-65.
32. Xu, L.and Brinkkemper, S. "Concept of product software," *European Journal of Information Systems* (16) 2007, pp 531-541.
33. Zafar, H.and Clark, J.G. "Current State of Information Security Research in IS," *Communications of AIS* (24), June 2009 2009, pp 557-596.

34. Zorz, Z. "End users are the main target of online attacks," in: *Help Net Security*, 2009.

35. Zou, J.and Pavlovski, C.J. "Control case approach to record and model non-functional requirements," *Information Systems & e-Business Management* (6:1) 2008, pp 49-67.