USIS: A Unified Framework for Secured Information System Lifecycle

Bei-Tseng Chu[1]
Huiming Yu[2]
Audrey Dance[2]

[1]Department of Software and Information Systems
UNC Charlotte
billchu@uncc.edu

[2]Department of Computer Science
North Carolina A&T State University
{cshmyu, asdance}@ncat.edu

Abstract

Information security ≠ information system + security features such as access control and intrusion detection. Security considerations must be an integral part of the entire lifecycle of the information system. This paper presents a framework to systematically classify all security considerations throughout the lifecycle of an information system. This framework can be seamless integrated into mainstream information system development frameworks and can serve as an effective guide for both students as well as practicing information system professional.

Keywords: Secure Software Development, Information Assurance, Information Security and Privacy, Information System Lifecycle.

Acknowledgement: This work has been supported in part by a grant from the National Science Foundation under the grant DUE 0416042.

1. Introduction

Given the importance of information in modern society and poor security track record of many commercial software products and information systems, information assurance throughout the entire information system life cycle is a topic of great interest to academic researchers, educators, as well practicing IT professionals. Securely designed information system with appropriate information security features is a much better alternative to the unsustainable penetration and patch model being practiced today. Experiences have shown time and again that poorly designed security features can be serious information security hazards.

There is a rapidly increasing body of published research and best practices, e.g., [2,3,4,5], for Information Assurance throughout information system lifecycle. IT professionals must fully consider both functional as well as security issues. Examples of security considerations for developers including checking array bounds to prevent buffer overflows, and avoiding using dynamic SQL whenever possible as they may be subject to SQL injection. Experiences have clearly shown that information system design and implementation is a complex endeavor that is prone to human errors. Seamless integration of functional and security consideration is key to any effective IS lifecycle methodology.

Many existing published best practices for secured information systems are organized based on the waterfall lifecycle model [4]. The amount of issues demanding attention can be overwhelming for practitioners. The Zachman framework [6] has shown to be an effective guide for information system designers to systematically consider important design and implementation issues. This paper proposes a Unified framework for Secured Information Systems (USIS). It extends the Zachman framework to cover important security considerations throughout the lifecycle of an information system. Due to space limitations, details of the original Zachman framework are not described. A summary of our contributions will be discussed later.

Our experiences as educators find this framework to be very effective to systematically organize this large body of knowledge and help students to develop valuable insights. Based on experiences with the original Zachman framework, we expect that USIS can also serve as an effective

guide to IT professions who work on different aspect of an information system to deliver more secure information systems and services.

2. USIS Overview

USIS is organized as a two dimensional matrix as illustrated in the appendix. The rows correspond to perspectives from key stake holders. They loosely map to a water fall model: customers (requirements), designer/architect (design), developer (coding / testing) , people involved in daily operators, people involved in maintenance and risk assessment (maintenance). The columns list key questions that must be addressed by key stake holders: what, why, how, when, where and, how good. The same person may fill multiple roles. Multiple persons may fill the same role with each answering only a subset of these questions, e.g. some implementation tasks may be outsourced.

The interpretation for the columns includes issues in traditional information systems. The "what" column focuses on data and information provided by the system. The "why" column focuses on motivation. The "how" column focuses on function and processes. The "who" column focuses on people. The "when" column focuses on timing. The "where" column focuses on network topology as well as physical facilities. The "how good" column focuses on assessment. In addition, information security considerations should be also considered associated with data, motivation, function, people, timing, network, and assessment.

Each cell in the matrix lists key answers to a specific question from the perspective of a specific stake holder. The appendix gives some examples. Traditional IS issues are depicted in regular font whereas *security issues are written in italics*; issues pertaining to both are underlined. Tools can also be organized into the cells as they are used to address related questions. The framework can be used in the context of a spiral development model [1] as each iteration in a spiral model often contains a subset of stages of a water fall process.

Space limitations only permit us to highlight a few examples to illustrate how USIS can be helpful to practitioners. Let's first consider the perspective of a customer.

One can start by describing (what) business information a particular information system is intended to serve. At the same time, the customer must also be aware of the business risks associated with the information systems such as reputation risk if private customer information were improperly exposed. He/she then need to understand the business objectives (why) as well as compliance issues the organization must address such as the SOX regulations. To describe the how the system works, he/she needs to clearly specify the functions of the system, privacy and security policies, and business continuity plans. The customer's view also includes key stake holders of the system, relevant organizational charts and key roles and responsibility (who). The role of how the information security organization should support the system under consideration must also be specified. Key business events (when) along with the business risks posed by them must be analyzed. Geographical location of the organizations interacting with the system (where) along with the business risks imposed by such a configuration must also be identified. Finally, an assessment plan must be given with specific performance metrics as well as security auditing policies.

A designer / information architect provides the logical data design schema (what). It is important to perform threat modeling against such a design as well as identifying privacy implications associated with statistical inferences. Business rules governing the system should be specified (why) along with analyzing potential threats associated with these rules. Logic design and process design are traditional ways to provide functional specifications. At the same time, he/she should identify the system's exposed attack surface. An attack surface is a channel through which malicious may be carried out. It might be input data fields on a screen or web page, or an API as part of a library. It is much easier to identify the attack surface in conjunction with functional requirement development while all the issues are fresh in the designer's mind.

It is also important for the designer to have a clear understanding of the workflow for people who will be interacting with the system. At the same time he/she should identify potential threats to information security as a result of these interactions, including issues related to physical security. Next the distributed aspect of the information system should be specified (where) along with analysis of the associated attack surface. Finally, assessment strategies must define such as design and security reviews.

An implementer would produce the physical data model (what), and implement business rules (why). He/she must practice secure coding to guard against malicious attacks, select appropriate cryptographic algorithms and provide security annotations, e.g.

Microsoft SAL, for other analysis tools. Usability issues (who) must be addressed as well as security considerations such as at when and how security dialogs should be placed. Specific events (when) and how they are handled must be coded in a secure way. Assessment at this level includes function testing, penetration testing, and integration testing and fuzzing. Fuzzing is an effective testing technique which feeds a program being tested with variations of input data such as different sizes and character combinations. It has shown to effectively uncover both functional defects as well as buffer overflow bugs.

An operator should understand the data that he/she is responsible of collecting and monitoring including the analysis of network intrusion logs (what). He/she must also under stand policies (why) to be followed, including security policies, and follow established procedures (how) including incident response procedures. Detailed roles and responsibilities of people interacting with the system must be clearly understood, including contacts and protocols to involved law enforcement organizations in response to security incidents.

An operator / operation manager, e.g. those involved in delivering IT help desk services, will answer the "what" question by understanding his/her role and responsibilities. Appropriate security trainings must be provided at this stage as they are often part of the front line defense against social engineering attacks. The operator must also understand events he/she must monitor, including security events (when). He/she must be understood the network topology as well as physical security (where). Clear assessment methodology must be put in place including performance metrics and auditing.

A person performing system maintenance / risk assessment should clearly understand changes to data along with changes in the threat model. (what). Similarly changes to business rules and environment must be understood (why) including associated new threats. He/she then must implement these changes practicing secure coding. Changes to user interfaces (how) must also be made along with full security considerations. Security issues must be fully considered in changes to events (when) are changes to networks including patch management (where). Finally regression testing and penetration testing must be performed.

3. Contributions

USIS is based on the Zachman framework [5], which only addressed system development phases of information systems, and it did not address information assurance issues. USIS is designed to cover the entire information system lifecycle including daily operations, maintenance, and risk assessment. The original framework did not highlight assessment issues. Issues related to testing were lumped within the "how" question. USIS explicitly creates a column "how good" to highlight many issues associated with assessment and testing as they are key components of information assurance.

To the best knowledge of the authors this is the first attempt to cover, in a uniform way, many key information assurance issues over the entire lifecycle of an information system. Because USIS is based on a well respected traditional IS framework, it also easily covers traditional software engineering issues. Our experiences as educators suggest that the columns are very valuable for students to organize and integrate the large amount of knowledge covered in Software Engineering and Information Assurance. For example all assessment related issues are grouped together in one column and discussed from perspectives of different stake holders. This helps students to understand subtle relationships between different techniques, e.g., penetration testing, functional testing, and fuzzing. As another example, by explicitly asking "what", "why", and "how" questions, USIS can help students develop deeper appreciation of the relationships between business objectives, business risks, government regulation, and organizational policies.

4. Future work

We plan to continue this work by putting key concepts throughout information system lifecycle and information assurance into USIS. Popular tools can also be put in the context of USIS. We believe such an exercise have significant values in education as well as providing guidance for practitioners.

From an education perspective, USIS has shown promise of being able to help students organize a large body of knowledge as well help develop deeper appreciation of important relationships between different perspectives. USIS can also be a very effective guide for practitioners in their effort to design, operation, and manage a large information system in a way that meet high information security standards.

References

[1] Boehm, B. "A spiral model of software development and enhancement" *IEEE Computer,* Vol 21, No. 5, pp. 61-72, 1988.
[2] Howard, M. Lablanc, D., and Viega, J. *19 Deadly Sins of Software Security* McGraw Hill, 2006
[3] Howard, M. and Lablanc, D. *Writing Secure Code* Second Edition, Microsoft Press, Redmond, WA 2002
[4] Howard, M. and Lipner, S. "The Trustworthy Computing Security Development Lifecycle" in MSDN library, (msdn.microsoft.com), March 2005
[5] Swiderski, F., and Snyder, W., *Threat Modeling* Microsoft Press, Redmond, WA, 2004
[6] Zachman, J. A. "A framework for information systems architecture" in *IBM System Journal* Vol. 26, No. 3, pp. 276-292, 1987

Appendix

	What (data / information)	Why (motivation)	How (function / process)	Who (people)	When (time)	Where (network / facility)	How good (assessment)
Customer	Business information, *Business Risk*	Business goals, *Compliance Regulation*	Functions provided, *Privacy/ Security policies,* business continuity	Stake holders, Org chart, roles and responsibilities, *Info sect org.*	Business events, *Business risk*	Locations business operates, *Business risk*	Performance metrics, *Security auditing policies*
Designer/ Architect	Logical data model, *Threat model, statistical DB security issues*	Business rules, *threat modeling*	Logic design, process model, *threat model attack surface, defense in depth*	Workflow model, *Threat model*	System events, processing structure, *Threat model*	Distributed system architecture *Attack surface, physical security*	Design reviews, *Security reviews*
Implementer	Physical data model, *Secure coding*	Rule design, *secure coding, SAL annotation*	Coding, *secure coding, SAL annotation, propagation of fixes, crypto algorithm selection*	Human interface designs, *security and privacy considerations in HCI*	Timing definitions, *secure coding, SAL annotation*	Network architecture *System patching, firewalls, IDS, physical security*	functional testing, fuzzing, *penetration testing*
Operator	Logs, data collection, *security logs*	Policies, *Security policies*	Procedures, *incident response procedure*	Detailed org charts, *relationship with law enforcement*	Detailed events, *recognize security events*	Network management, *,Physical security*	Performance metrics, auditing, *penetration testing*
Maintenance and risk assessment	Data changes, *Threat modeling*	Business rule changes, *changes in environment, threat modeling, secure coding*	Implement changes, change management, *Propagation of fixes*	Workflow, HCI, *threat model*	Changes in events, *threat modeling, secure coding*	Network changes, *corresponding security changes, patch*	Regression testing, functional testing, *penetration testing*