NOTATIONS FOR FACILITATING SOFTWARE SECURITY DESIGN

Muhammad Nadeem[1*] and Adul Hussain Shah Bukhari[1]

[1] Faculty of Information and Communication Technology, Balochistan University of Information Technology, Engineering & Management Sciences, Quetta
*Corresponding author e-mail: rabbea_09@yahoo Nadeem@buitms.edu.pk

Abstract

The conventional software designing tools do not address the software security design, the security considerations are taken care of independently and there is no de facto unified mechanism•to•design•software's•functional•requirements•along•with•the•security•requirements,•it allows the applications more vulnerable to the security threats, this is especially true in client-server / web based systems. In this research designing notations are being proposed that can be integrated with the existing designing tools to address software security design. The notations have less abstraction in order to design security requirements more clearly and effectively. Security is not a feature that can be added to software or "bolted on" after other software features are codified, nor can it be "patched in" after attacks have occurred in the field. Instead, security must be built in from the very beginning (requirements specification) and included in every subsequent System Development Life Cycle phase.

Keywords: Software Design, Security, Notations, Reliability, UML

INTRODUCTION

The existing software designing tools, both classical and object oriented, have notations / symbols that are capable to design the functionality of the system but lack the notations to design the security of the system, therefore its strongly needed that there must be notations that can be used with the existing tools that cover both the functional and security design of the software, this will result in generation of artifacts representing both functional and security requirements.

The Motivational Factors

The software is the essential part of Information Technology infrastructure for an enterprise; hence the enterprise security involves the security of the software being used, the authors strongly support the view that the security requirements should be reflected in the artifacts
created for software functional requirements at the analysis or design stage.

How this Research is organized

The first part of this research investigates the de facto designing and modeling tools being used by the software designers; and explores to how much extent do they address the software security requirements? The second part proposes some notations that can be used with the existing software designing & modeling tools to address software security design along with the rest of the design. Some examples have also been presented that use the proposed notations.

Importance of Security

The security needs are evident at every stage for an enterprise, the application security lies at the core of the security paradigm as depicted by figure-1.

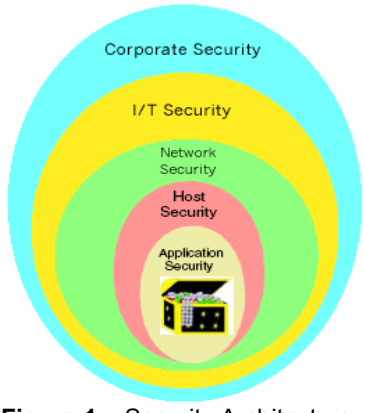

Figure 1 – Security Architecture

This• research• discusses• the• application's security to be achieved by proper design of software applications.

LITERATURE REVIEW

The Contribution of Umlsec

UMLSec has been proposed as an extension of UML, it addresses four security concepts namely• "Fair• Exchange",• "Confidentiality• or Secrecy",• "Secure• Information• Flow"• and "Secure•Communication•Link"•to•facilitate•the security aware software designing [1].

The concepts presented in UMLSec are very good initiative to secure systems development however the authors have found that abstraction level of UMLSec is very high, which leads to problems to depict the security requirements in the software design artifacts.

Integration of MAC and UML Artifacts

Researchers have been focusing on the notion that security must be give key importance in the design of software applications at all stages of the lifecycle for accurate and precise policy definition, authorization, authentication, enforcement, and assurance. UML is one of the dominant players in software design used for specifying, visualizing, constructing and documenting artifacts of software. UML provides alternate diagrams to get different perspectives for different stakeholders, e.g.: use case diagrams for the interaction of users with system components, class diagrams for the static classes and relationships among them, and sequence diagrams for the dynamic behavior of instances of the class diagram.

However,• UML's• support• for• the• definition• of security requirements for these diagrams and their constituent elements (e.g., actors, systems, use cases, classes, instances, include/ extend/ generalize relationships, methods, data, etc.) is lacking. Efforts have already been made e.g., in [2] security issue by incorporating Mandatory Access Control (MAC) into use case, class, and sequence diagrams, providing support for the definition of clearances and classifications for relevant UML elements have been addressed.

Policy Based Security Frameworks

Web-based applications are one of the important candidates that need secure development practices to be adopted; they comprise dynamic, extensible and interoperable collections of services, software components and information shared by various entities performing transactional tasks. In defining a general-purpose policy-based security framework, security policies for the confidentiality, integrity and availability of services and information need to be considered. In [3], the policies for authentication, access control, security management, identity administration and accountability have been proposed; and the implementation mechanisms and component-based generic security services for web-enabled applications are also discussed.

Security for Mobile Application Designing

Designing applications for mobile devices needs special attention as there is involvement of public medium (i.e. wireless), possibility of loss or theft due to small size, extensive mobility, and power consumption issues etc. we typically need to address the following security concerns [4]:

Table 1 – Security concerns of mobile applications

S#	Security Concern
1	User identification
2	Secure storage
3	Secure data communication
4	Secure network access
5	Secure content
6	Secure software execution environment
7	Temper resistant implementation

The security concerns discussed in Table-1 depict the need of secure system design; these factors are generally true for all embedded systems; the factors identified were a source of motivation for the researchers to work out on the notations that can be used for the design of secure systems.

RESEARCH METHODOLOGY

THE PROPOSED NOTATIONS

This section contains the detailed description of the notations for designing software security. The notations are especially useful when designing security related client-server / web based software applications.

CREATING NOTATIONS

The figure 3 gives a general structure of the notations used in this research.

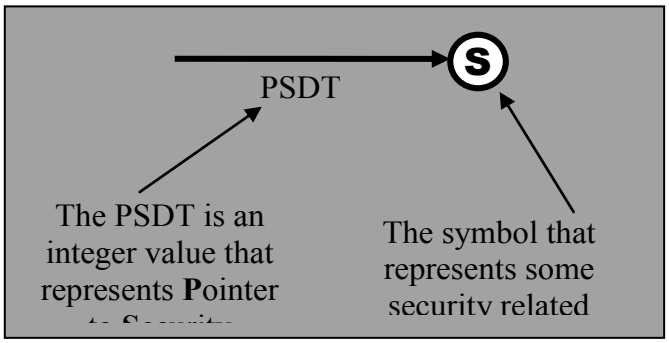

Figure 2: Guidelines for designing security notations

Each symbol used to depict some security scenario has a security descriptor table associated with it; which is a collection of key-value pairs, the general structure of the "Security•Descriptor•Table•has•been•shown•in table-2.

Table 2 – General Structure of Security Descriptor

Security Descriptor		
S#	KEY	VALUE
1	PSDT	An Integer value
2	Attribute	Value
3	Attribute	Value
4	(custom parameter)	(Description of custom parameter)

Each symbol used to depict some security scenario has a security descriptor table associated with it, which is a collection of key-value pairs. The security descriptor table has some mandatory attributes having their respective values, in order to provide flexibility the designer may add additional key-value pairs in the security descriptor table.

THE SYMBOLS USED TO MODEL NEW NOTATIONS

ENCRYPTION

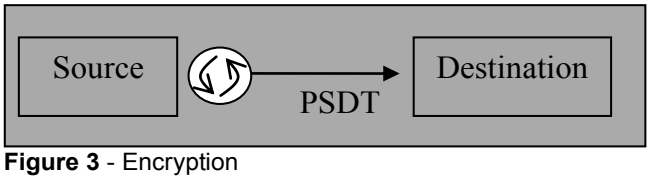

Figure 3 - Encryption

The figure 4 depicts the encryption process, A security descriptor i.e., tabular structure, is associated with each symbol; it is pointed to by the• "security• notation"• using• PSDT• which stands for **P**ointer to **S**ecurity **D**escriptor **T**able, the PSDT is an integer value that starts from "1"• in• a• given• design• artifact• and• is incremented by one, and assigned to the next notations• used.• The• "Security Descriptor Table"•holds•the•collection•of•key-value pairs that describe details about the security requirement. More rows can be added at the end of the table to hold the optional or custom "Key-Value"•pairs,•see•table-3.

Table 3 – Security Descriptor Table for Encryption

Security Descriptor		
S.#	KEY	VALUE
1	PSDT	An Integer value
2	Source	Describes source
3	Destination	Describes destination
4	Protocol / Algorithm	Defines Protocol to be used
5	Server port	Defines server port to be used
6	Client port	Defines client port to be used
7	(custom parameter)	(Description of custom parameter)

CLIENT COMPUTER AUTHENTICATION

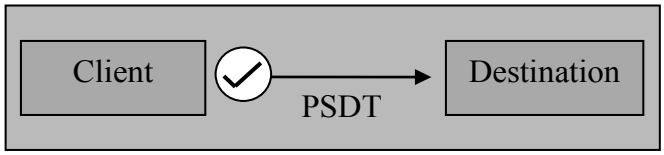

Figure 4: Client Computer Authentication

The figure 5 represents the notation proposed for the Client Computer Authentication; the associated• "Security• Descriptor• Table"• is shown in the table-4.

Table 4 – Security Descriptor Table for Client Authentication

Security Descriptor		
S.#	KEY	VALUE
1	PSDT	An Integer value
2	Client	Describes client machine
3	Destination	Describes destination machine
4	Mechanism	Describes the authentication mechanism to be used, e.g., IP based authentication, Certificate based authentication etc.
5	(custom parameter)	(Description of custom parameter)

SERVER COMPUTER AUTHENTICATION

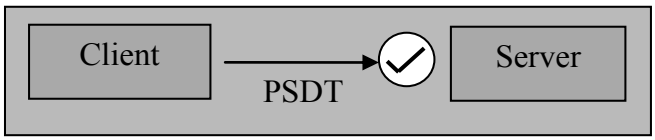

Figure 5: Server Computer Authentication

The figure 6 represents the notations proposed for Server computer authentication; the associated• "Security• Descriptor• Table"• is shown in table-5.

Table 5 – Security Descriptor Table for Server Authentication

Security Descriptor		
S#	**KEY**	**VALUE**
1	PSDT	An Integer value
2	Client	Describes client machine
3	Server	Describes the server machine
4	Mechanism	Describes the authentication mechanism to be used, e.g., IP based authentication, Certificate based authentication etc.
5	(custom parameter)	(Description of custom parameter)

SESSION MAINTAINED COMMUNICATION

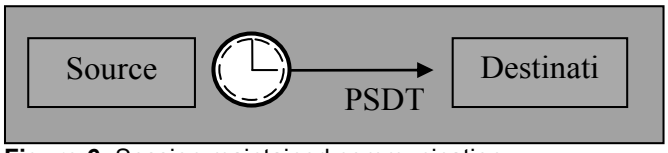

Figure 6: Session maintained communication

The figure 7 represents notation proposed for "Session Maintained Communication"; associated• "Security• Descriptor• Table"• is shown in table-6.

Table 6 – Security Descriptor Table for Sessions

Security Descriptor		
S.#	**KEY**	**VALUE**
1	PSDT	An Integer value
2	Source	Describes source
3	Destination	Describes destination
4	Protocol / Mechanism	Defines Protocol / Mechanism to be used e.g., HTTP etc
5	Session Expiry time	n – time units
6	(custom parameter)	(Description of custom parameter)

CONTENT VERIFICATION USING HASH

Figure 7: Content verification using hash value

The figure 8 represents the notation proposed for "Content verification using hash value"; associated• "Security• Descriptor• Table"• is shown in table-7.

Table 7 – Security Descriptor Table for Hashing

Security Descriptor		
S#	**KEY**	**VALUE**
1	PSDT	An Integer value
2	Source	Describes source machine
3	Destination	Describes destination machine
4	Algorithm / Mechanism	Defines Algorithm / Mechanism to be used e.g., MD5 Hash etc
5	(optional / custom parameter)	(Description of optional parameter)

AUTHENTICATION

Figure 8: Authentication

The figure 9 represents the notation proposed for "Authentication"; the associated "Security Descriptor•Table"•is•shown•in•table-8.

Table 8 – Security Descriptor Table for Authentication

Security Descriptor		
S#	**KEY**	**VALUE**
1	PSDT	An Integer value
2	User	User description who needs to be authenticated for certain resource
3	Resource	Describes resource being accessed e.g., a file, service or device etc.
4	Service / Mechanism	Defines authentication service or mechanism to be used, e.g., password based authentication, biometric technique etc.
5	(custom parameter)	(Description of custom parameter)

USER VERIFICATION USING SENSORY EXPERIENCE

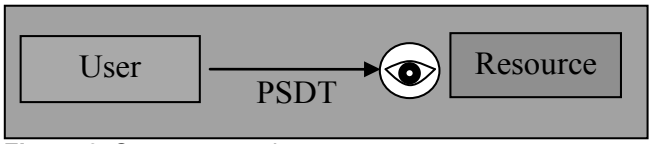

Figure 9: Sensory experience

The figure 10 represents the notation proposed for "User verification using sensory experience"; the associated "Security Descriptor Table" is shown in table-9.

Table 9 – Security Descriptor Table for Sensory Experience

Security Descriptor		
S#	**KEY**	**VALUE**
1	PSDT	An Integer value
2	User	User description who needs to be authenticated for certain resource
3	Resource	Describes resource being accessed e.g., a file, service or device etc.
4	Mechanism	Defines mechanism to be used, e.g., user is asked to recognize a pattern containing a particular phrase / letters, or he may be required to recognize an audio or visual situation.
5	(custom parameter)	(Description of custom parameter)

AUTHORIZATION

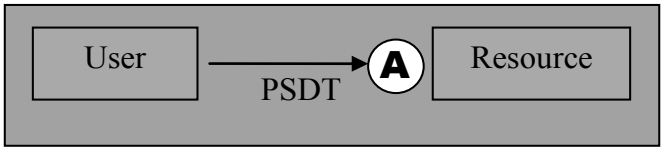

Figure 10: Authorization

The figure 11 represents the notation proposed for the "Authorization"; the associated "Security Descriptor Table" is shown in the table-10.

Table 10 – Security Descriptor Table for Authorization

Security Descriptor		
S#	**KEY**	**VALUE**
1	PSDT	An Integer value
2	User	User description who needs to be authorized for certain resource
3	Resource	Describes resource being accessed e.g., a file, operation, service or device etc.
4	Mechanism	Defines mechanism to be used for the authorization, e.g., Role Based Access Control (RBAC) etc.
5	(custom parameter)	(Description of optional parameter)

ANONYMOUS ACCESS

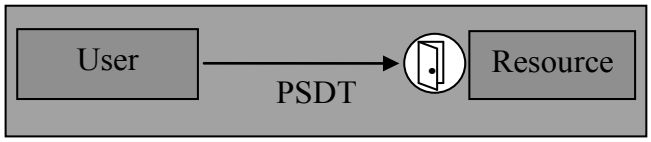

Figure 11: Anonymous access

The figure 12 represents the notation proposed for "Anonymous Access"; the associated "Security Descriptor Table" is shown in the table-11.

Table 11: Security Descriptor Table for Anonymous Access

Security Descriptor		
S#	**KEY**	**VALUE**
1	PSDT	An Integer value
2	User	User description who anonymously accesses certain resource
3	Resource	Describes resource being accessed e.g., a file, operation, service or device etc.
4	Mechanism	Defines mechanism to be used for anonymous access.
5	(custom parameter)	(Description of custom parameter)

RESULTS AND DISCUSSION

USING NOTATIONS WITH EXISTING TOOLS

The proposed notations were used with the existing design and modeling tools, some of the examples are discussed in this section.

USING NOTATIONS WITH UML DIAGRAMS

In this example a user enters the Login ID and password which is encrypted and sent to the server for the verification, on repeated login failures the user is asked to recognize a pattern in order to ensure that it is not a machine trying to guess a password. This problem is solved using UML activity diagrams with integration of proposed notations.

Table 12 – Security Descriptor Table for Figure 13

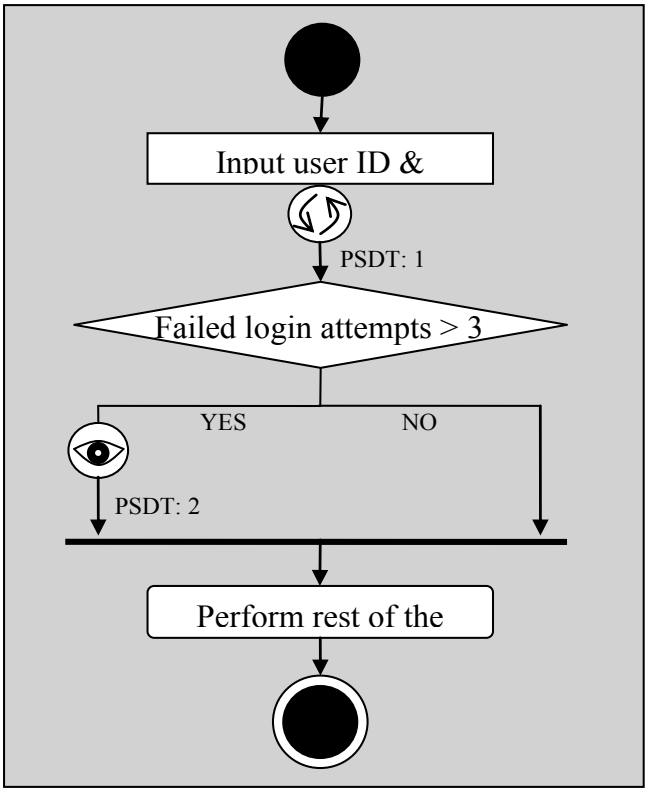

Figure 12: The Enhanced Activity Diagram

Following are the Security Descriptor tables (table 12 & 13) associated with figure 13.

Table 13 – Security Descriptor Table for Figure 13

Security Descriptor		
S#	KEY	VALUE
1	PSDT	1
2	Source	User's•machine
3	Destination	Authentication Server
4	Protocol / Algorithm	Public Key Encryption

CONCLUSIONS

With the passage of time the importance of security is increasing and this growth shall be exponential in the future. The security of software applications in general and web based / client server applications in particular lie at the center of the security focus. Security designing tools and frameworks will become very common in the next few years, therefore as a theoretical work to support such future frameworks and CASE tools there is need of extensive research in security design. This research contributes by proposing notations to facilitate software security design.

Security Descriptor		
S#	KEY	VALUE
1	PSDT	2
2	User	System user
3	Resource	Account
4	Mechanism	Recognize a fuzzy word being displayed in a bitmap image

FUTURE WORK / OPEN ISSUES

Currently this research has focused the security concerns related to web applications / client server computing, but security encompasses several other areas as well the common examples may be the security of code being executed in a machine, security of passwords and other credentials stored somewhere etc. also need to be modeled and the design notations are not yet proposed. The proposed security techniques shall be evaluated by carrying out a project by two different groups first using the proposed security techniques for designing the application and the second group using the conventional methods for the same, the software designed using security aware designing tools should be less vulnerable to the threats.

The software developers / designers feel handicapped unless they have proper CASE tools to facilitate them and make the software engineering process more and more efficient. Developing the CASE (Computer Aided Software Engineering) tools to facilitate software designers has got vital importance; it is being left for the future work.

There are still some security concepts that may be too abstract to be represented as a notation that may be used to model software security, therefore when ever designing notations for any security concept / issue care must be taken on the fact weather it can represent the concept precisely or not. Designing confusing notations may lead to ambiguity to the designing process.

ACKNOWLEDGMENT

The central and critical intellectual acknowledgement belongs to my supervisor Dr. A. H. S. Bukhari for his guidelines and continuous support throughout this work both in technical and research paper writing areas.

REFERENCES

- **Jan• Jurjens,• "UMLSec**:• Extending• UML• for• Secure• Systems• Development",• Software• & Systems Engineering, Department of Informatics, Munich University of Technology, Germany, 2002.
- **Thuong Doan and Steven Demurjian**,•"MAC•and•UML•for•Secure•Software•Design",•FMSE'•04, October 29, 2004, Washington, DC, USA, copyright ACM 1-58113-971-3/04/0010.
- **Marian Ventuneac, Tom Coffey, Ioan Salomie**, "A policy-based security framework for web-enabled applications", marian.ventuneac@ul.ie, University of Limerick, Department of Electronic and Computer Engineering, Proceedings of the 1st international symposium on Information and communication technologies, 2003
- **Anand Raghunathan, Srivaths Ravi, Sunil Hattangady, and Jean-Jacques Quisquater,** "Securing Mobile Appliances: New Challenges for the System Designer"; NEC Laboratories America, Princeton, NJ, USA; Proceedings of the "Design, Automation and Test in Europe (DATE)"•Conference•and•Exhibition,•©•2003•IEEE