SECURE SOFTWARE DEVELOPMENT BASED ON A COMPREHENSIVE FRAMEWORK

MOHAIDEEN.A [1]
Department of CSE (Data Science)
Malla Reddy College of Engineering, Dhulapally, Secunderabad
Telangana,India
mohaideenamqtr@gmail.com

V.SREETHARAN[2]
Department of Computer Science and Engineering, St Martin's Engineering College, Dhulapally, Secunderabad, Telangana, India
sreetharannerist@gmail.com

E.PAVITHRA[3]
Department of CSE (Data Science)
Malla Reddy College of Engineering, Dhulapally, Secunderabad
Telangana,India
pavielango@gmail.com

Abstract: **- To design, build and deploy secure systems, we must integrate security into our application development life cycle and adapt current software engineering practices and methodologies to include specific security-related activities. Developers enforces security measures during design phase of software development processes which may end up in specifying security related architecture constraints that are not really necessary. To eliminate this problem, we propose a Framework for Security Engineering Process that involves converting security requirements and threats into design decisions to mitigate the identified security threats. The identified design attributes are prioritized and a security design template is prepared.**

Keywords- Security engineering process, Security requirement engineering, Security design engineering, Security design template, Cryptographic techniques, Cryptographic attacks

I. INTRODUCTION

Processes and techniques for implementing security in software and related systems are the main emphasis of security engineering.

A growing focus on how to create secure software has been sparked by the quick development of internet-based software systems that store sensitive data and perform essential functions [1]. The security requirements that are enforced by current software development procedures during the design phase may lead to the specification of unnecessary security-related architectural constraints. The final software system may therefore employ less effective mechanisms at higher expense.

Fire smith [2] defined security requirement as high level requirements that gives specification of system behavior and distinguish these from security related architectural constraints so that requirement engineers can discover true security requirements. A number of proposals have been made for finding and eliciting security requirements, including attack trees [7], misuse cases [8, 9], common criteria [10], and abuse cases [3, 4]. Based on well-established secure software development practice standards from organizations like BSA, OWASP, and SAFE Code, the Secure Software Development Framework (SSDF) is a collection of fundamental, solid, and secure software development principles. Software security is rarely explicitly covered in detail in software development life cycle (SDLC) models, therefore principles from the SSDF must be added to and incorporated into every SDLC implementation.

By using the SSDF techniques, software developers can lessen the amount of vulnerabilities in their products, lessen the potential effect of exploiting undiscovered or untreated flaws, and deal with the underlying causes of vulnerabilities to stop them from happening again. Software developers and buyers can improve their communication for procurement procedures and other management tasks by using the SSDF, which offers a consistent vocabulary for discussing safe software development methods. Our earlier work [19–21] proposed a framework for secure software development, where security engineering activities like (i) Security requirements elicitation, analysis & prioritization, specification and management; (ii) Appropriate design decision; and (iii) Implementation of all functionalities incorporating design decision should be carried out.

A technique for developing software that integrates security at each stage of the software development life cycle (SDLC) is known as secure software development. These design choices for security requirements call for a systems-level modeling of the software environment. We focus on design decision issue (ii) in this work. Following the elicitation and prioritization of security needs, the best design choices will take into account security threats. This is essential for today's software projects. These initiatives must find a cost-effective solution without compromising the requisite availability of necessary services because of their tight schedules.

In our method, design choices are made to mitigate the identified security dangers once security needs and threats are determined during the security requirement elicitation step. The various security services are matched to the many sorts of security requirements. A security design template (SDT) is created based on the collection of critical security attributes that affect design decisions at different tiers of the security engineering process after the identified design attributes are prioritized. By removing unneeded design constraints in a given case, we finally learn about the precise cryptographic procedures that will eventually aid in the latter stages of the design process. The repository houses a variety of cryptography methods together with their analytical properties. We use a relevant case study from the Web to demonstrate our process.

II. RELATED WORKS

Selection of inappropriate software package and security modules may be very costly. Adversely affect business processes and functioning of the organization inversely [22]. The common security goal is to keep the assets of the organization securely. The organizations are needs protect from intruder or accidental detriment by recognized actors of the system. Conventionally it is known as security of information and is expressed as confidentiality, integrity and availability of information in an organization which is called as CIA triad.

- ***Security engineering***

Complex security-related processes, including security engineering, are involved. Identification, management, and implementation of security requirements, security design, implementation of security mechanisms, and security testing are among the security-related activities [13]. Design decisions may be made that are most suitable when actual security needs are defined. In addition to proposing an overall structure for the software from a security standpoint, the design phase defines how the identified needs (gathered from the security requirement engineering phase) might be implemented in a specific context. The major objective of this phase is to obtain an orderly and systematic framework for security functionality and the design choices made for it.

Complex security-related processes, including security engineering, are involved. Identification, management, and implementation of security requirements, security design, implementation of security mechanisms, and security testing are among the security-related activities [13]. Design decisions may be made that are most suitable when actual security needs are defined. In addition to proposing an overall structure for the software from a security standpoint, the design phase defines how the identified needs (gathered from the security requirement engineering phase) might be implemented in a specific context. The major objective of this phase is to obtain an orderly and systematic framework for security functionality and the design choices made for it.

In 2002, an aspect oriented design technique is proposed to model and integrate security concerns into design by weaving the aspects in a primary model [12]. A design aspect can be modelled from a variety of perspective. In this paper only static and interaction aspects views are considered. This paper describes steps for weaving an aspect in primary model. But the technique fails to address the impact of the security concern on each design unit with respect to a given application environment. Also it does not focus on any well-defined framework through which developers can make design decisions to develop efficient cost effective secure system.

A design technique employing security patterns and the PICO design model using UMLsec are proposed by Apvrilla et al. in 2005 [13]. The PICO program has three separate services, including a subscription service for handling initial user registration, a presence service for maintaining a list of online users, and an instant messaging service for sending the user's instant message to the recipients. The design of the presence service using security patterns and the design of the instant messaging service using UMLsec are the main topics of this paper. These topics are only a portion of the development process and do not constitute a whole solution to the issue. "Trustworthy Computing Security creation Lifecycle (or SDL)" is a procedure that Microsoft has established for the creation of software

that must survive hostile assault, according to Lipner et al.'s [14] discussion. The high-level principles of secure by design, secure by default, and secure in deployment are described in this article, along with experiences with their use in Microsoft software. However, the software must be architected, built, and deployed in order to safeguard both the data it processes and itself, as well as to fend off assaults in a specific application context. A secure software development policy is not only advised, but in some circumstances, it is actually required. Organizations complying with SOC 2 Type 2 or ISO 27001, for instance, are required to establish a secure development policy. Your team may create a custom policy from start or draw inspiration from tools like the ISO 27001 template manual.

Data security using cryptography is a methodical defense against outsiders. To prevent unauthorized change, a sensitive item (asset) needs encryption, authentication, and access control. The term "symmetric" or "secret key cryptography" refers to any type of text encryption or decryption that uses the same key for both operations. Table 1 below lists a few symmetric algorithms along with their block and key sizes.

Where symmetric key methods of cryptography are utilized, the sharing of the secret key—which is necessary for both encryption and decryption—can be a serious security risk. This is not a problem in the same manner in asymmetric or public key encryption. The employment of two mathematically linked keys—public and private—ensures that plain text encrypted with one key can only be unlocked by using the other. The solution reduces the chance of security being compromised because the private key is not provided by the user. The most cutting-edge methods are founded on HECC [26] and ECC [25]. Table 2 below lists the timings of several operations for a few asymmetric algorithms.

III. CRYPTOGRAPHIC ATTACKS

Usually a system is designed to protect against certain threats, while other threats might not have been addressed [24]. A threat is a harm that can happen to an asset, composed of a threat agent and an attack method [17]. Thus a good authentication scheme should provide protection from different attacks relevant to that protocol. Here we have identified some of the possible attacks for password based authentication and short formed (as UA1, UA2 etc.) them. These cryptographic attacks are listed below.

Name of algorithm	**Block size (bits)**	**Key size (bits)**	**Encryption speed (on 33 MHZ 486SX) (Kb/s)**
DES	64	56	35
Blowfish	64	128	182
3DES (Triple DES)	64	168	12
IDEA	64	128	70
AES	128	128	60
CAST	64	128	53
RC5	64	128	86
RC4 (Stream cipher)	One byte at a time	256	164
SEAL (Stream cipher)	One byte at a time	160	381
PIKE (Stream cipher)	One byte at a time	160	62

Table 1: Some popular asymmetric ciphers [32]

Name of algorithm	Encryption (ms)	Decryption (ms)	Decryption (ms)	Verify (ms)
RSA (512 bits)	30	160	160	20
RSA (768 bits)	50	480	520	70
RSA (1024 bits)	80	930	970	80
ECDSA (160 bits)	797	281	150	230
ECDSA (233 bits)	882	385	250	521
ECDSA (283 bits)	928	400	25	580
HECDSA (81 bits)	668	191	60	31
HECDSA (83 bits)	893	224	56	32
ElGamal (512 bits)	330	240	250	1370

Table 2: Some popular hash functions [32]

•Man-in-the-middle attack (UA1)

The attacker intercepts the message sent between the client and the server and replay these intercepted messages within a valid time with recorded messages.

•Denial of service attack (UA2)

The attacker updates password verification information on smart card to some arbitrary value so that legal user cannot login successfully in subsequent login request to server.

•Replay attack (UA3)

The passive capture of data and its subsequent retransmission to show unauthorized effect. Any unauthorized malicious user can send duplicate data repeatedly to the receiver which is already sent.

•Perfect forward secrecy (UA4)

The user's password is compromised, it never allows the adversary to determine the session key for past sessions and decrypt them.

•Impersonation attack (UA5)

The attacker impersonates legitimate client and forges the authentication messages using the information obtained from the authentication scheme.

•Dictionary attack (UA7)

There are two types of dictionary attacks named as Offline dictionary attack and online dictionary attack. In Offline dictionary attack, the attacker can record messages and attempts to guess user's identity and password from recorded messages. In Online dictionary attack, the attacker pretends to be legitimate client and attempt to login on to the server by guessing different words as password from a dictionary.

•Stolen verifier attack (UA8)

An attacker who steals the password-verifier (e.g., hashed passwords) from the server can use the stolen-verifier to impersonate a legal user to log in the system.

•Smart card loss attack (UA9)

When the smart card is lost or stolen, unauthorized users can easily change the password of the smart card, can guess the password of the user by using password guessing attacks.

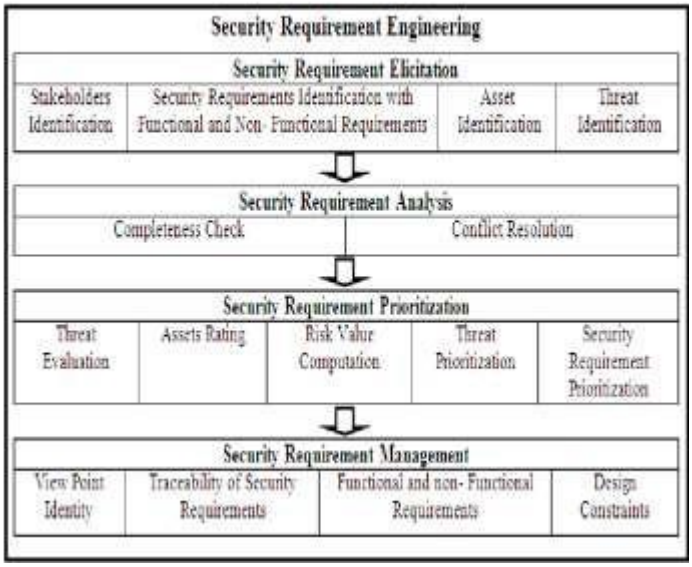
•Insertion attack (UA10)

In this type of attack, the attacker modifies or inserts some messages on the communication channel with the hope of discovering client's password or gaining unauthorized access.

IV. PROPOSED FRAMEWORK FOR SECURITY ENGINEERING

Security engineering deals with security-related activities which include identifying security requirements, prioritizing and management of security requirements, security design, implementing security mechanisms, security testing. The proposed framework for overall security engineering process (SEP) is shown in Fig. 1.

direct actor.



The concept of this framework for security engineering process is an attempt to propose a design framework taking the view of stakeholders as well as environmental constraints in the earlier software development phases [21]. We now discuss in detail each activity of this proposed framework.

A. Security requirement engineering

This phase involves security discovery requirements, eliciting, analyzing and managing them. It consists of four different stages: Security Requirement Elicitation, Security Requirement Analysis, Security Requirement Prioritization and Security Requirement Management. All the different activities perform in each stage of this process are explained below:

i. ***Step 1:*** *security requirement elicitation*

In Security Requirement Elicitation phase different tasks are performed as explain below:

•Identifiy the various abstract classes of actors as direct and indirect actors. Direct actors are those who directly interact with the system such as human, software system and hardware devices. Indirect actors refer to developers who develop software and people who regulate application domain.

•Identify the functionalities of each actor conceptualized in the previous step and also determine associated non-functional requirements.

•Identify the threats associated with each of the functional requirements or data which is used by the functionality.

•Define the security requirements such as authentication, integrity, non-repudiation etc. to mitigate these threats.

ii. ***Step 2:*** *security requirement analysis*

The various tasks perform in analyzing the security requirements are as follows:-

•Checking for completeness we make a check list to check that the elicited security requirements have mitigated all the threats to the functionality of the system.

In the first step, we will evaluate the threats based on the estimated risk value. For this we have to perform the following tasks:

•Threat assembling after identifying the threats, a repository of the threats will be developed as in common criteria based approach [6, 30]. Actor profiles will be maintained also in this repository. Thus predefined threats can be retrieved from the repository according to the profile of the actor.

•Threat rating after threat assembling, we have assigned a value of each threat according to CRAMM [11].

•Vulnerability measurement Assigning value to corresponding vulnerability.

•Asset rating identify the concerned affected asset and give them a value.

•Estimate the value of risk we can measure risk as Risk = value based on measure of (Threat, Vulnerability, Asset). After threat rating, assigning vulnerability value and asset value, we will use the table given by the CRAMM [11].

In the second step, we will prioritize the security requirement after identifying threats. Initially we have identified the measures of risk to all the threats and prioritize them based on value of risk. After finding out the high prioritized assets that are involved with the particular security requirements, we calculate the priority of security requirement just from the value of threat priority.

iii. ***Step 4:*** *security requirement management*

As security requirement also evolve along with functional and nonfunctional requirements, it is necessary to maintain the information about traces of each security requirements and its associated attributes in this phase. The techniques for requirement management presented in [31] can be used for this activity. There are different types of traceability information that must be maintained for the management of security requirements.

iv. ***Step 3:*** *security requirement prioritization*

As security requirements are to mitigate threats and avoid vulnerability and risk, they will be prioritized on the measure of threat, vulnerability and risk. So prioritization is done in following two steps:

Evaluation of threats Prioritization of security requirement. Cryptographic services, design structuring and finally design decisions. The steps of this process are explained below.

B. Security design engineering

This phase deals with designing a software structure that realizes the specification. So depending upon the identified security requirements, we identify the cryptographic services to mitigate the identified security threat of the system. Bad decisions made during the design phase can lead to design flaws that can leave the system vulnerable to security threats, so we focus on the design phase through a set of systematic design activities mainly identification of

•Conflict resolution we resolve the contradictions that may exist in the security requirements elicited from different viewpoints.

•Grouping of requirements this step consists of identifying the security requirements that can be grouped together.

i. Step 1: *mapping of security requirements with security services*

After the security requirements have been identified, we proceed to the design phase of the security engineering process i.e. prioritized security requirements are mapped with security services like confidentiality, integrity, authentication and non-repudiation services. The different types of security requirements proposed by Fire smith [2] are mapped to the different security services provided by cryptography. This would eventually help in the later stages of the design process, by specifying which crypto- graphic techniques would be suitable in a particular scenario. After the security services have been identified for the particular security requirement, we proceed to the next activity i.e. security design analysis.

ii. Step 2: *security design analysis*

This step will define what the prioritized threats are and which assets are affected by these threats. This step consists of two sub steps as explained below:

(2a) Mapping of the Prioritized threats with related attacks. In security requirement prioritizing process we identify different threats and prioritized the security requirements according to threat analysis. Now in this step, we first identify the threats which affect the assets with high value. Then we identify what type of attacks can be caused by these threats.

(2b) Mapping of attacks to security mechanism (cryptographic techniques). In this step we map the security attacks with the available techniques of cryptography and calculate the impact of these attacks. From our literature survey, the security analysis result is shown in Table 5, for a password based authentication in wireless network.

Applicability of the attacks on an algorithm shows that whether the algorithm resists the attack (if resists then applicability 'N' else 'Y') or not. For example, AES does not resist the attack UA1 (Man-in-the-middle attack), thus in Table 5, we have marked it as 'Y' that means this attack can be applicable for AES when it is used in a password based authentication in wireless network. In this mapping table we also calculate the total impact which will help us to find out

a set of algorithms which can protect the assets. In this step the total impact shows the consequence of the attacks that means if an attack is not applicable on an algorithm then the impact of the attack is zero. Accordingly we calculate the impact of all ten identified attacks for an authentication scheme which can be based on any one of the above specified algorithm. On the next step we will consider the design

iii. Step 3: *identifying security design constraints*

A very common cause of protocol failure is that the environment changes, so that assumptions that were originally true no longer hold and the security protocols cannot cope with new environment. A security environment describes the context in which the software is expected to evolve. The environment affects the kind of threats the application is likely to encounter. This is only because each environment has some design constraints. So before design struck-turning, first we have to find design constraints.

The Environmental Constraints of the target deployment system is considered here depending on whether the system would be implemented on a wireless/mobile/mobile ad hoc environment. For a web based system in wireless network, there are many communicational constraints (like channel capacity, bandwidth, power, through put etc.) and computational constraints (like memory, encryption speed, energy etc.). Suppose some important information is transmitted from a mobile phone in a mobile network. One of the design constraint (encryption speed) here plays an important role for selection of cryptographic technique. While comparing the encryption speed of symmetric/asymmetric ciphers (shown in Tables 1, 2) on that device, some of the suitable crypto techniques are DES, 3DES, CAST, RSA etc. Now the design attributes will help us to select the best suitable technique.

iv. Step 4: *security design structuring*

In this activity, different design attributes are identified which affects the selection of cryptographic protocols. The sub steps are explained below:

(4a) Identifying design attributes and prioritizing them. While identifying them we have to first look whether the system would be implemented on a wireless/mobile/mobile ad hoc or any other environment. The design attributes like cost, implementation platform etc. greatly affect our design choices because only a subset of cryptographic algorithms can work efficiently on constrained environments. Further, cryptographic algorithm would differ depending upon the service requirement. For example, symmetric key algorithms like AES, 3DES would be more suitable for confidentiality service requirement as they are 1000 times faster than asymmetric key algorithms like RSA which are less constraints to choose a particular security algorithm which is best for a particular environment. Efficient for large plain text encryption. But asymmetric key algorithm like ECC is more suitable in constrained devices with limited memory/processing power/energy etc. for its short key size. Mainly we separate the design attributes on the basis of the devices used because their priorities are different for High-end and Low-end devices. 4b) Preparation of security design template (SDT). After the security requirement and threats have been identified in the requirement phase and security services and design attributes identified in the first (phase of the design process, we proceed with the next step in which a security design template (SDT) is prepared to take care of each security requirement. A design template is shown in next section. This template will store each specification of the design constraints and design attributes of a particular environment for further processing.

V. Conclusion

With a significant emphasis on security design engineering, we have created a Framework for Security Engineering Process. Every stage of the life cycle has security considerations. The many security services in our architecture are linked to the various sorts of security requirements. To identify the specific cryptographic algorithms in a given situation, the discovered design criteria are prioritized, and a security design template is created. The developers may effectively discover and apply the right cryptographic approach in a given context with the aid of this framework. Eliciting security requirements and security design become an inherent component of system engineering and security engineering since this approach is cohesive with the traditional software engineering process.

References

1. Anderson, R., Security Engineering: A Guide to Building Dependable Distributed Systems. John Wiley & Sons, New York, 2001.

2. Basin, D., Doser, J., Lodderstedt, T., Model Driven Security for Process Oriented Systems. In Proceedings of the 8th ACM symposium on Access Control Models and Technologies, Como, Italy, 2003

3. Bresciani, P. Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A., TROPOS: An Agent Oriented Software Development Methodology. In Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers Volume 8, Issue 3, Pages 203-236, 2004 62 H. Mouratidis, J. Jürjens, and J. Fox

4. CEPSCO, Common Electronic Purse Specifications, Business Requirements ver. 7, Functional Requirements ver. 6.3, Technical Specification ver. 2.2. Available from http://www.cepsco.com [2000].

5. Crook, R., Ince, D., Lin, L., Nuseibeh, B., Security Requirements Engineering: When Anti-requirements Hit the Fan, In Proceedings of the 10th International Requirements Engineering Conference, pp. 203-205, IEEE Press, 2002

6. Cysneiros, L.M. Sampaio do Prado Leite, J.P., Nonfunctional Requirements: From Elicitation to Conceptual Models. IEEE Trans. Software Eng. 30(5): 328-350 (2004)

7. Devanbu, P., Stubblebine, S., Software Engineering for Security: a Roadmap. In Proceedings of ICSE 2000 ("the conference of the future of Software engineering"), 2000.

8. Giorgini, P., Massacci, F., Mylopoulos, J., Requirements Engineering meets Security: A Case Study on Modelling Secure Electronic Transactions by VISA and Mastercard, in Proceedings of the International Conference on Conceptual Modelling (ER), LNCS 2813, pp. 263-276, Springer-Verlag, 2003.

9. Hermann, G. Pernul, G., Viewing business-process security from different perspectives. International Journal of electronic Commence 3:89-103, 1999

10. Jürjens, J., Shabalin, P., Tools for Critical Systems Development with UML (Tool Demo), UML 2004 Satellite Events, Nuno Jardim Nunes, Bran Selic, Alberto Silva, Ambrosio Toval (eds.), LNCS, Springer-Verlag 2004E. [Accessible at http://www.UMLsec.org. Protected content can be accessed as user: Reader, with password: Ihavethebook]. Available as open-source.

11. Jürjens, J., Secure Systems Development with UML, Springer, March-Verlag, 2004

12. McDermott, J., Fox, C., Using Abuse Case Models for Security Requirements Analysis. In Proceedings of the 15th Annual Computer Security Applications Conference, December 1999.

13. Mouratidis, H., A Security Oriented Approach in the Development of Multiagent Systems: Applied to the Management of the Health and Social Care Needs of Older People in England. PhD thesis, University of Sheffield, U.K., 2004

14. Mouratidis, H., Giorgini, P., Manson, G., Integrating Security and Systems Engineering: towards the modelling of secure information systems. In Proceedings of the 15th Conference on Advanced Information Systems (CaiSE 2003), Velden –Austria, 2003

15. Object Management Group, OMG Unified Modeling Language Specification v1.5, March 2003. Version 1.5. OMG Document formal/03-03-01.

16. Saltzer, J., Schroeder, M., The protection of information in computer systems. Proceedings of the IEEE, 63(9):1278–1308, September 1975.

17. Schneider, F., editor. Trust in Cyberspace. National Academy Press, Washington, DC, 1999. Available as http://www.nap.edu/readingroom/books/trust/.

18. Schneier, B., Secrets & Lies: Digital Security in a Networked World, John Wiley & Sons, 2000

19. Schumacher, M., Roedig, U., Security Engineering with Patterns. In Proceedings of the 8th Conference on Pattern Languages for Programs (PLoP 2001), Illinois-USA, September 2001

20. Schumacher, M., Security Engineering with patterns. In LNCS 2754, Springer-Verlag, 2003

21. Shamir, A., Crypto Predictions. In 3rd International Conference on Financial Cryptography (FC 1999), 1999.

22. The Economist, Digital rights and wrongs, July 17, 1999

23. van Lamsweerde, A., Letier, E., Handling Obstacles in Goal-Oriented Requirements Engineering, Transactions of Software Engineering, 26 (10): 978-1005, 2000

24. Viega, J., McGraw, G., Building a Secure Software. Addison-Wesley, Reading, MA, 2002.