Volume 2, Issue 4, April 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Threat Modeling and Security Pattern used in Design Phase of Secure Software Development life Cycle

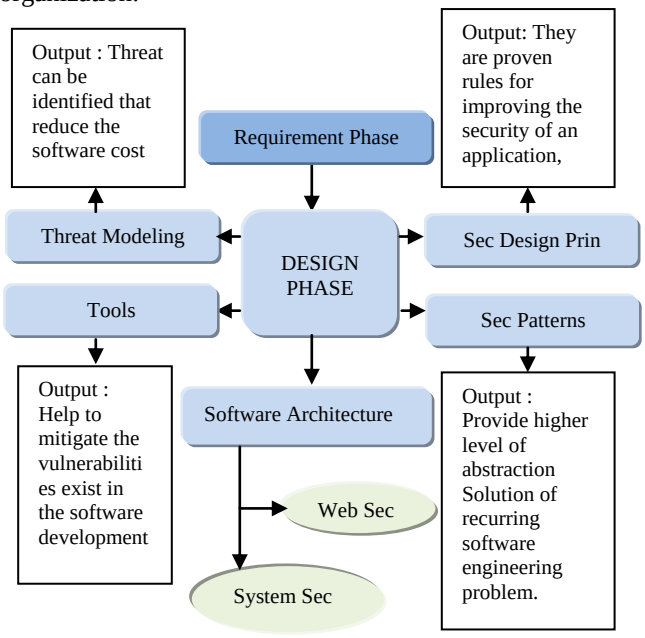
Mr. Swapnesh Taterh P.A.H.E.R.Univ, Udaipur swapnesh_t@yahoo.com
Prof (Dr.) K.P Yadav Director
Prof (Dr.) S.K Sharma Director

Abstract: Improving software-engineering practices and processes can lead to the creation of secure software, as software that realizes with justifiably high confidence, The process improvement generates software releases with extremely few defects, reasonable development costs, lower maintenance costs, and enhanced product reputation. The purpose of this paper is to explain the use of Software Architecture and Threat Modeling with respect to the security in the design phase of software development. These are one such method that, when integrated in the software development process, can help developers to prevent security problems in software. Software is malleable in the design phase, once the architectural design and threat modeling are set, and then after the cost and complexity of making changes rises tremendously.

Keywords : Software, Security, SDLC, Software Design, Secure Design.

1. Introduction

As the technologies increases there is a wide use of software which increases the threat to the security of software. Various factors that harm the organization work from inside or outside are increasing day by day. They not only affect the financial status of the organization but also adversely affect the credibility and integrity of the organization.



Fig, 1 Overview of Design Phase of Secure Software Development Life Cycle

Now a day's security is not a unique feature of the software it's a essential part of the software which has to be taken care of during the process of a software development. The security model, which has to be used, must consider these issues effectively and efficiently. If software is not secure then all its operations are exposed to attacks

2. Software Architecture

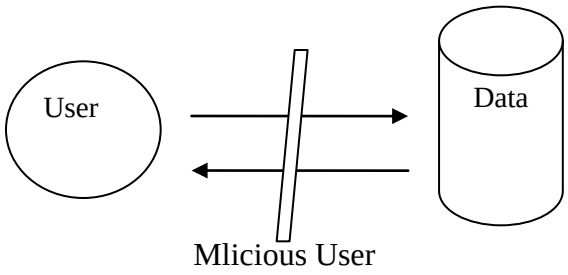
Define security architecture and design guidelines. The Software Architecture describes the essential elements in the system, their structure with their relationship. This will defined in software requirement specification and also in architecture specification of the software. Architecture specification comprised of a number of view that depict key system concern from a number of different perspectives. The architecture specification enables the designer to show all the primary characteristics in a complex system and their relationship to each other and with external system.

2.1 Application-Specific

Depending on the application being designed, the types of issues that must be addressed vary. The categories defined in each application-specific security frame were defined by security experts who have examined and analyzed the top security issues across many applications.
Define the overall structure of the software from a security perspective, and identify those components whose correct functioning is essential to security.

2.2 System Security Architecture

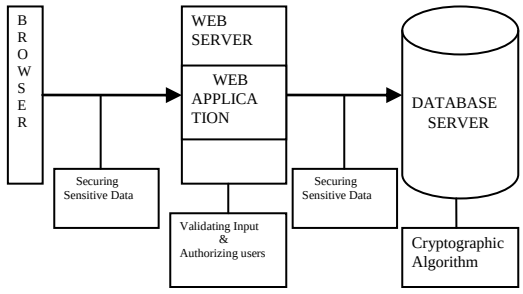
We know from the past that many security vulnerabilities on a system take place either by sending data from the system (e.g., buffer overflow exploitation) or by receiving data from the external system (e.g., symlink). In both of these types of security vulnerabilities, the malicious user connects to a system and invokes the system's methods (e.g., API), and sends (receives) data items (e.g., input strings) into (from) the system. The malicious user can also send (receive) data indirectly into (from) a system by using persistent data items (e.g., files); the malicious user can send data into a system by writing to a file that the system later reads. Hence, the malicious user uses a system's methods and data items to harm the system.



Sending and receiving data

2.3 Web Security Architecture

When we design a secure Web application, it is important that we follow guidelines to ensure effective user authentication and authorization, to protect sensitive data as it is transmitted over public networks, and to prevent attacks such as session hijacking. Some of the important Web application issues that must be addressed with secure design practices are shown in Figure.



WEB APPLICATION DESIGN ISSUE

Identify design techniques, such as layering, use of strongly typed language, application of least privilege, and minimization of attack surface, that apply to the software globally. (*Layering* refers to the organization of software into well-defined components that are structured so as to avoid circular dependencies among components—components are organized into layers and a higher layer may depend on the services of lower layers, while lower layers are forbidden from depending on higher layers.) Specifics of individual elements of the architecture will be detailed in individual design specifications, but the security architecture identifies an overall perspective on security design.

3. Security design principles

Security design principles are a specific type of guidelines and practices. They are proven rules for improving the security of an application, and in order to be useful, the principles must be applied to specific problems. This is the great advantage with them since they can be identified during the requirements phase doing threat modeling. There exist
a large number of such principles, and even though just reading through them once in a while will improve security consciousness, the real value is added when they are directly used to identify weaknesses and argue for architecture and implementation decisions.
The security design principles in Table are built upon the idea of simplicity and restriction. They were described by Saltzer and Schroeder but a more up to date description and examples are given by Bishop

Examples of design principles.

1.Principle of Least Privilege :The principle of least privilege states that a subject should be given only those privileges that it needs.
2.Principle of Fail : Safe Defaults The principle of fail-safe defaults states that, unless a subject is given explicit access to an object, it should be denied access.
3.Principle of Economy of Mechanism : The principle of economy of mechanism states that security mechanisms should be as simple as possible.
4.Principle of Complete Mediation : The principle of complete mediation requires that all accesses to objects be checked to ensure that they are allowed.
5.Principle of Open Design :The principle of open design states that the security of a mechanism should not depend on the secrecy of its design.
6.Principle of Separation of Privilege : The principle of separation of privilege states that a system should not grant permission based on a single condition.
7.Principle of Least Common Mechanism : The principle of least common
mechanism states that mechanisms used to access resources should not be shared.
8.Principle of Psychological Acceptability : The principle of psychological acceptability states that security mechanisms should not make the resource more difficult to access than if the security mechanisms were not present.

4. Security patterns

A security pattern is a well-understood solution to a recurring security problem, and encourages effective re-use for building in robustness. They are patterns in the sense originally defined by Christopher Alexander1, applied to the domain of information security. Software design patterns have become widely accepted after the

Gang of Four published their very influential book on this topic in the middle of the nineties, and there exists a vast number of patterns for software development, see for example Hillside2 for an extensive online library.
However, while some security patterns take the form of traditional design patterns, not all of them are design patterns. There are many sources containing and related to security patterns, which all have very much in common, but some of these tend to resemble general guidelines and rules-of-thumb more than traditional patterns, and this can be a bit confusing.

Security patterns are usually divided into different types and categories, typically:
1.Structural, behavioral and creational security patterns encompass design patterns, such as those used by the Gang of Four. They include diagrams on relationships between entities and descriptions of interaction and object creation.
2.Available System patterns are a sub-type of structural patterns and they facilitate construction of systems which provide predictable uninterrupted access to the services and resources they offer to users.
3.Protected System patterns is another sub-type of structural patterns that facilitate construction of systems which protect valuable resources against unauthorized use, disclosure, or modification.
4.Antipatterns are ways of not doing things based on things that have failed in the past or invalid assumptions. An antipattern should also include a solution, e.g. reference to a working pattern.
5.A mini-pattern is a shorter, less formal discussion of security expertise in terms of just a problem and its solution. Programming language specific

5. Threat modeling

Threat modeling is a security-analysis methodology that can be used to identify risks and guide subsequent design, coding, and testing decisions. The methodology is mainly used in a project's earliest phases, using specifications, architectural views, dataflow diagrams, activity diagrams, and so on. But it can also be used with detailed design documents and code. The goal is to address those threats with the potential to cause maximum damage to an application.
Overall, threat modeling involves decomposing an application to identify its key assets, and then identifying and categorizing the threats to each asset or component. The threats should be rated according to a risk ranking, which should guide the development of threat mitigation strategies in designs, code, and test cases.
Threat modeling is also the process of identifying what functionality and which assets an attacker can take advantage. The software design should be evaluated from an attacker's point of view. This process will result in a threat model document that can be used by developers to identify which threats are present, and which steps should be taken to mitigate the associated risks. Swiderski and Snyder list the following purposes of threat modeling:

• Understand the threat profile of a system.
• Provide a basis for secure design and implementation.
• Discover vulnerabilities.
• Provide feedback for the application security life cycle.

Threat modeling is not solely connected to the design phase, threat modeling also considered an important part of the requirements phase, as well as an iterative process, continuously revisited throughout the software lifecycle. Typical artifacts in threat modeling are: attack trees, threat trees, misuse cases and cause and effect diagrams.
Results from threat modeling should feed directly into product design: no design is complete until it accommodates the potential threats against it. Revisit this stage regularly through the development life cycle—remember, attacks only get better.

6. Overview of Tools

Several types of tools are available to support secure software production. These range from static code analyzers and checkers to automated tools for verifying and validating formal specifications and design. Tools such as Prefast, Prefix and Model checking are helping to reduce overall defects
The Threat Modeling Tool also combines with issue-tracking system, making the threat modeling process a part of the standard development process.
The Threat Modeling Tool enables the developer to:

- Communicate about the security design of their systems
- Analyze those designs for potential security issues using a methodology
- Suggest and manage mitigations for security issues

7. Conclusion

The Software Process may consider the seemingly unconnected facts on the requirements for and capabilities of processes for producing secure software, a recommended path emerged. The author has given the System Architecture and Web Architecture in terms of secure software design. The author also has provided a way that following the guidelines and principles will lead to software that is secure and reliable. We have also shown the concept of thread modeling and the tools that are used with threat modeling. We had a concluded that without the use of threat modeling one can not enabled the security features in the software.

8. References

[1] Martin Gilje Jaatun and Inger Anne Tondel , "Covering Assets in Software Engineering," in Availability, Reliability and Security (ARES'08), Barcelona, Spain, Mar. 2008, pp.1172–1179.

[2] Anurag Agarwal, "Threat modeling enhanced with misuse cases,"searchsoftwarequalitytechtarget.com http://searchsoftwarequality.techtarget.com/t.html. Aug.2,2008.

[3] Chandramohan Muniraman and Meledath Damodaran, "A practical approach to include security in software development," Issues in Information Systems, Vol 2, No.2, pp 193-199, 2007.

[4] Sridhar and K. W. Hamlen, "ActionScript in-lined reference monitoring in Prolog," in the Proceeding 11th International Conference on Verification, Model Checking, and Abstract Interpretation, January 2010.

[5] M. Jones and K.W. Hamlen, "Disambiguating aspect-oriented security policies," in Proceeding 9th International Conference on Aspect-Oriented Software Development, March 2010.

[6] A.Apvrille and M. Pourzandi, "Secure Software Development by Example," in the IEEE Security & Privacy, vol. 3, 2005, pp. 10-17.

[7] N. Davis, "Developing Secure Software," The DoD Software Tech News, vol. 8, pp. 3-7, 2005. Proc.3rd International Conference on the

[8] M. Dowd, J. McDonald, and J. Schuh, The Art of Software Security Assessment: Addison-Wesley, 2007.

[9] S. Lipner and M. Howard, "The Trustworthy Computing Security Development Lifecycle," 2005.

[10] The Ballista Project, http://www.cs.cmu.edu /afs/cs.cmu.edu/project/edrcballista on 20 April 2007.

[11] JTest, http://www.parasoft.com/jsp/products/home jsp?product=Jtest, April 2007.

[12] OWASP CLASP Project,http://www.owasp.org/ index.php/Category:OWASP_CLASP_Project. May 2009.

[13] I. Flechais, C. Mascolo, and M.A.Sasse, "Integrating Security and Usability into the Requirements and Design Process, "International Journal of Electronic Security and Digital Forensics, Inderscience Publishers, Geneva, Switzerland, 2007, vol. 1, no. 1, pp. 12-26.

[14] A.S. Sodiya, S.A. Onashoga, and O.B. Ajayi, "Towards Building Secure Software Systems," Issues in Informing Science and Information Technology, Informing Science Institute, California, USA, 2006, vol. 3, pp. 635-646.

