Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 50 (2012) 10351-10353

ISSN: 2229-712X

Software Security

Paritosh Bapat[1] and Shivani Dole[2]
Department of IT, VIT University, Vellore, Tamil Nadu-632014.
Department of CSE, VIT University, Vellore, Tamil Nadu-632014.

ARTICLE INFO

Article history:
Received: 7 July 2012;
Received in revised form:
16 August 2012;
Accepted: 6 September 2012;

Keywords
Software;
Software Security;
Vulnerabilities;
Software Development Life Cycle;
Threat Modeling.

ABSTRACT
This paper untangle the idea of software security and applying sound practices to the phases of software development life cycle that design and develop software and subject all phases to risk analysis and testing. It focuses on importance of secure software and unveils why and how attackers target software. It brings into light some of the exploitable software defects and certain methods to make software less vulnerable to attacks. It also presents the idea of software testing and its goal with the perspective of considering security as an important requirement to test.

Introduction

Software security is the process of designing, building, and testing software for security. Software developers build software that can withstand attack. In this way, the development process identifies and removes problems in the software itself. Building secure software is an important responsibility of software developer. Cyber attacks are getting common these days therefore claims about software reliability and safety must include provision for built in security of the software. There is a need to avail and use the tools, knowledge and guidance that will improve security of software. This means that software at the level of an individual component, program or application will be attack-resistant. It was in 2001 that the developers, architects, and computer scientists started systematically studying how to build secure software.

Importance

Software is present everywhere. Services provided in different sectors depend on software that performs accurately. It enables and controls business operations and nation's critical infrastructure. Critical functions are dependent on software which makes it a high valued target. Because of this dependency terrorist and criminals approach malicious attackers in order to target Software controlled system. Software also protects other software. Routers, firewall, encryption system are implemented through the use of software. On a personal level, our financial transactions are exposed to the Internet and implemented using web service technologies. We shop, bank, pay taxes, buy insurance, invest online. Therefore software security matters.

How Threat Works

Hackers exploit vulnerabilities present in the software. Vulnerabilities that can be exploited by attackers to perform code injection attacks are an important kind of implementation error. They exploit errors in the software code which create abnormal condition in the software. The hackers need not know the version of the software. An exploit-kit such as Eleanor, crimepack etc is used for attacking. Malicious software also known as malware, help hackers disrupt users' computer operation, gather sensitive information, or gain unauthorized access to a computer system. It can be software, script or code. These kits work by creating a web server, a php page on some random host that just serves malware and exploits the victim's browser or his flash or downloads a pdf and exploits his pdf reader. By merely clicking a link or browsing to the affected page or downloading and running a file, the handling application is exploited and the attacker drops his executable on the host. It is not always possible to use safe languages like Java or C#. Moreover, many programmers have expertise in C-like languages and they use them in order to develop new products. In certain cases, the use of C-like languages is necessary for system software as programmers need direct access to memory, which is not possible in safe languages. Vulnerabilities are the consequences of memory management errors occurring in C-like languages. Using such vulnerabilities, an attacker can overwrite memory locations that the execution environment relies on to execute programs accurately. Virus, Trojan horse, unauthorized access, Buffer overflows come under the top vulnerabilities and threat category. The "Code Red" worm exploited a buffer overflow to be able to run arbitrary code on the vulnerable machine, allowing it to spread by copying itself to the hosts it infected. Such vulnerabilities are an important issue in C language.

Software Development Life Cycle

Software development process which is security enhanced is important to achieve secure software. It roots out vulnerabilities or exploitable defects from software. It also ensures that the defects do not occur again. Sound practices and principles are applied throughout the life cycle. Scope of development activities is expanded so that security is given the same importance as it is given to functional and non-functional requirements of the software. Risk management activities and checkpoints are integrated throughout the life cycle so that the software can resist and recover from attacks. This expansion affects the software development lifecycle in following ways.

Tele:
E-mail addresses: shivanics.vit@gmail.com

Requirement specification

Assessment of risk and vulnerabilities is carried out while finding out, analyzing and documenting functional requirements specification. This helps to capture non functional security requirements that will put a check on identified risks and vulnerabilities which when addressed through the system's design should be able to reflect mitigations or reductions in risk. That is how these requirements are included in software specification which otherwise would not have been included. Some percentage of vulnerabilities injected during requirement activities are removed during requirement analysis or by developing cases that represent abnormal condition.

Software design

It defines the overall structure of the software from a security perspective and identifies those portions of the software whose correct functioning is essential to system security. It determines how the software will operate and how modules will work together in order to provide software specific functionalities. So it should be analyzed such that functions are not exposed to attackers. The selection of programming language, tools and components should be such that there is no vulnerability present in the software. At the design and architecture level, a system must be coherent and present unified security architecture. Security analysts should uncover and rank risks so that mitigation can begin. Disregarding risk analysis at this level leads to costly problems later.

Implementation

During the implementation phase, the software developers perform coding, testing and integrating the software. At the code level, implementation flaws are discovered by static analysis tools which scan source code for common vulnerabilities. The usage of safe languages such as Java and C# gives programmers less direct access to memory and can thus prevent bugs from occurring. Bugs that are present in source code should be taken into account which might result in incorrect functionalities of the software and make it behave in an unintended way. Garbage collecting languages, do not allow programmers to manually free memory, removing the problem of dangling pointer references. Safe languages will usually not allow the programmer to directly manipulate pointers or perform pointer arithmetic, preventing an important cause of buffer overflows. Applying static analysis code scanning tools and security testing tools and conducting security code reviews are common practices in order to remove certain percentage of vulnerability in software.

Software Testing

The software testing includes security reviews and tests that are driven not by verification of requirements objectives but by the need to identify any vulnerability in the software that the new threats can successfully target.Security criteria will be generated for software's unit and integration test plan. These tests checks whether software development phases have met security exit criteria. Testing security functionality with standard functional testing techniques and risk-based security testing are performed based on attack patterns and threat models.

Distribution and deployment

Any residual security issue is removed by cleaning the code before distribution of software. When the software is installed, certain configuration parameters are set to protect the software in deployment. This phase marks the end of sound practices that are applied throughout the software development lifecycle.

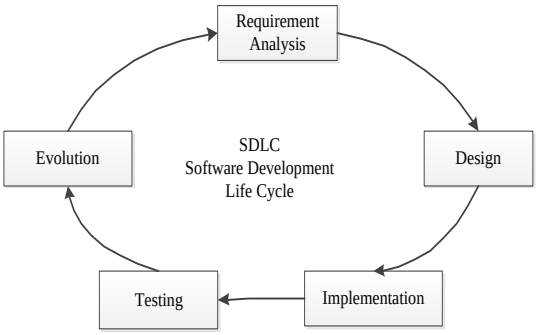

Figure 1 Software Development Life Cycle

Threat Modeling

Threat modeling is a technique that adapts security risk analysis that makes analyst to think like an attacker while systematically exploring the software. It is an iterative process that starts during the architecture or high-level design phase. It has become an important part of software development lifecycle process. It is a description of security aspects. In order to know the mitigation steps that must be taken, threats are modeled by the architecture team. Set of possible attacks and the potential harm to the software are determined which helps to take measures in advance to eradicate the threat.

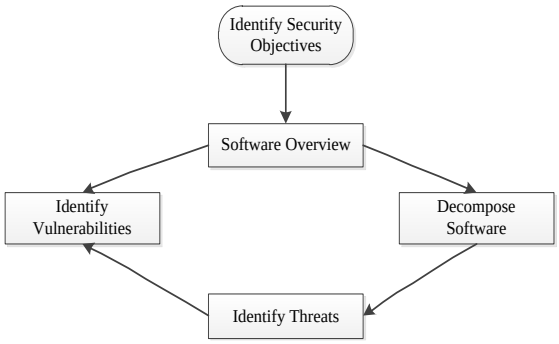

Figure 2 Threat Modeling

Enhance Software Security

Software defects can be avoided if software developers are equipped in order to recognize security implication of design and implementation that they choose. They should be able to anticipate the security implications of unexpected behavior of software functionalities when combined together or of even simple defects occurring in it. Various methods make different trade-offs in terms of performance, effectiveness, memory cost, compatibility, etc.

MDA

Model Driven Architecture (MDA) is a software design approach for the development of software systems. It is based on generating code from Unified Modeling Diagrams. Main aim is to separate design from architecture. The design addresses the functional requirement architecture and provides infrastructure by which non functional requirements are realized. By using MDA, developers need to write small as well as less complex code. Thus the software contains fewer design and coding error.

Object Oriented Model

Unified Modeling Language is the industry-standard language for specifying, visualizing, constructing and documenting the developmental phases of software. Security is a non functional requirement in object oriented modeling. It is possible to specify security aspect in UML model by a set of UML security stereotype. It guides developers by annotating vulnerable model parts or places where a flaw can occur. This also helps to generate test cases.

Agile Methods

It promotes development of high quality software. Certain key practices in agile methods reduce the vulnerable defects that are present in the software like simple design, continuous testing and pair programming. Pair programming ensures higher quality code, enhanced trust and learning. Programmers working in pairs produce shorter programs with better designs and fewer bugs than programmers working alone. Continuous Testing is achieved when uninterrupted tests run twenty four hours a day, seven days a week, and the results of this testing can be efficiently processed. It increases product quality by detecting more defects in a shorter period of time. One objective of agile development is to put minimal software into production as quickly as possible and then enhance it. Use of this technique for security has several benefits and drawbacks. The nature of an evolving design leaves the product vulnerable to the problems of an add-on product. Leaving requirements open does not ensure that security requirements will be properly implemented into the system. However, if threats were analyzed and appropriate security requirements were developed before the software was designed, secure or trusted software could result.

Software Testing

It verifies that the software is secure. It does this by unit as well as integration testing. It detects security defects, coding error and other vulnerabilities that are present in the software. It demonstrates the secure behavior of software when subjected to threats or attack. In addition, it verifies that software exhibits its required security property in any condition.

The goal of security in software is to withstand or resist attacks and recover rapidly from attacks that cannot be resisted or withstood. The test plan for software security should include test cases that are not included in requirement based testing. The test cases should demonstrate as follows:

- The software behaves consistently under any condition.
- Areas of code are not exploited by attackers.
- Integration of modules of software is consistently secure.
- Even if the software fails it is not exposed to attackers.
- Exception and error handling are able to resolve all faults and error and does not leave the software exposed to attackers.

Conclusion

Software security is about building secure software or making sure that software is secure and educating software developers, architects and users about how to build secure software. Software is critical and increasingly exposed to threats posed by malicious, recreational hackers, cyber terrorists or cyber criminals. In addition to this, demand for better and improved functionalities along with time constraint does not allow careful specification of design, implementation, coding and testing. As a result there are number of areas developed in the software with defects and flaws. All these reasons make the software vulnerable. Most technologists acknowledge this undertaking's importance, but they need some help in understanding how to tackle it. Security holes in software are common. Exploring software security best practices helps in this direction. Security enhanced software development life cycle ensures mitigation strategies against threat attacking the vulnerabilities right from the beginning of software development. Software defects can also be avoided by methodologies like Model Driven Architecture, Object Oriented Model and Agile Methods. Such expansion of ideas in this field and increasing demand of software everywhere makes it mandatory to consider software security a high value discipline in software engineering.

References

[1] Software Security, http://www.microsoft.com/security/sdl/default
[2] Joe Jarzombek, Karen Mercedes Goertzel. "Security in the Software Life Cycle".
[3] Karine P. Peralta, Alex M. Orozco, Avelino F. Zorzo, Flavio M. Oliveira. "Specifying Security Aspects in UML Models".
[4] Prof. Dr. W. JOOSEN, Prof. Dr.F. PIESSENS. "Efficient countermeasures for software vulnerabilities due to memory management errors".
[5] MDA, http://en.wikipedia.org/wiki/Model-driven_architecture.
[6] Bugs, http://en.wikipedia.org/wiki/Software_bug
[7] Threat Modeling, http://en.wikipedia.org/wiki/Threat_model.
[8] Gary McGraw, "Software Security".