SANS SOFTWARE SECURITY with Frank Kim

Secure Coding.
Practical steps to defend your web apps.

Copyright SANS Institute

Author Retains Full Rights

This paper is from the SANS Software Security site. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Secure Coding: Developing Defensible Applications (DEV536)"
at http://software-security.sans.orghttp://software-security.sans.org/events/

Steven F Burns
GIAC Security Essentials Certification (GSEC) Practical Assignment
Version 1.4c

Threat Modeling: A Process To Ensure Application Security

January 5, 2005

Abstract

This paper discusses the importance of implementing application security at design time. It introduces threat modeling as a process to ensure that application security is implemented at design time. This paper also outlines and defines the process of creating a using a threat model.

1.0 Introduction

Application security has become a major concern in recent years. Hackers are using new techniques to gain access to sensitive data, disable applications and administer other malicious activities aimed at the software application. The need to secure an application is imperative for use in today's world. Until recently, application security was an afterthought; developers were typically focused on functionality and features, waiting to implement security at the end of development. This approach to application security has proven to be disastrous; many vulnerabilities have gone undetected allowing applications to be attacked and damaged. This raises the following questions: How can application security become an integral part of the development process? How can an application design team discover and avoid vulnerabilities in their application? There are three measures that can help discovering and avoiding security vulnerabilities:

- Have many experts contribute and share their knowledge
- Make expert knowledge about typical vulnerabilities available to developers and
- Make system-specific knowledge available to persons searching for vulnerabilities.

[collaborative]

One method being used to implement application security in the design process is threat modeling. The basis for threat modeling is the process of designing a security specification and then eventually testing that specification. The threat modeling process is conducted during application design and is used to identify the reasons and methods that an attacker would use to identify vulnerabilities or threats in the system. Threat modeling accomplishes the following:

- Defines the security of an application
- Identifies and investigates potential threats and vulnerabilities
- Brings justification for security features at both the hardware and software levels for identified threats
- Identifies a logical thought process in defining the security of a system
- Results in finding architecture bugs earlier and more often

© SANS Institute 2000 - 2005 Author retains full rights.

- Results in fewer vulnerabilities
- Creates a set of documents that are used to create security specifications and security testing, thus preventing duplication of security efforts

By using threat modeling to identify threats, vulnerabilities and mitigations at design time, the system development team will be able to implement application security as part of the design process. A good example of why threat modeling is needed is located at [matters].

2.0 How to Create a Threat Model

To create a threat model, a systematic approach is required. One approach is to use data flow. By using the data flow approach, the threat modeling team is able to systematically follow the flow of data throughout the system identifying the key processes and the threats to those processes. The data flow approach uses three main steps: view the system as an adversary, characterize the system and determine the threats. The threat modeling process outlined below is adapted from these references: [threat], [security].

2.1 View the system as an adversary

When an adversary views the application, they only see the exposed services. From the exposed services, the adversary formulates goals to attack the system. The steps used to understand an adversary's goals are detailed below.

2.1.1 Identify the Entry/Exit Points

Entry/exit points are the places where data enters or exits the application. When identifying entry/exit points, the following data should be identified and collected:

- Numerical ID - Each entry/exit point should have a numerical ID assigned to it for cross-referencing with threats and vulnerabilities.
- Name - Assign a name to the entry/exit point and identify its purpose.
- Description - Write a description that outlines what takes place at that entry/exit point. Identify the trust levels that exist at that point.

2.1.2 Identify the Assets

Assets are the reason threats exist; an adversary's goal is to gain access to an asset. The security team needs to identify which assets need to be protected from an unauthorized user. Assets can be either physical or abstract, i.e. employee safety, company's reputation etc. Assets can interact with other assets and, because of this, they can act as a pass-through point for an adversary. When identifying assets, the following data should be identified and

© SANS Institute 2000 - 2005 Author retains full rights.

collected:

- Numerical ID - Each asset should have a numerical ID assigned to it for cross-referencing with threats and vulnerabilities.
- Name - Assign a name to the asset.
- Description - Write a description that explains why the asset needs protection.

2.1.3 Identify the Trust Levels

Trust levels are assigned to entry/exit points to define the privileges an external entity has to access and affect the system. Trust levels are categorized according to privileges assigned or credentials supplied and cross-referenced with entry/exit points and protected resources. When identifying trust levels, the following data should be identified and collected:

- Numerical ID - Each trust level should have a numerical ID to cross-reference with entry/exit points and assets.
- Name - Assign a name to the trust level.
- Description - Write a description that explains more detail about the trust level and its purpose.

2.2 Characterize the System

In order to characterize the system, background information will need to be gathered about the system. The background information will help the security team focus and identify the specific areas that need to be addressed. There are five categories of background information:

- Use Scenarios
- External Dependencies
- External Security Notes
- Internal Security Notes
- Implementation Assumptions

2.2.1 Use Scenarios

Use scenarios describe how the system will be used or not used in terms of configuration or security goals and non-goals. A use scenario may be defined in a supported or unsupported configuration. Not addressing use scenarios may result in a vulnerability. Use scenarios help limit the scope of analysis and validate the threat model. They can also be used by the testing team to conduct security testing and identify attack paths. The architect and end users typically identify use scenarios. When defining use scenarios, the following data should be collected:

- Numerical ID - Each use scenario should have a unique identification number.
- Description - Write a description that defines the use scenario and

© SANS Institute 2000 - 2005 Author retains full rights.

whether it is supported or not supported.

2.2.2 External Dependencies

External dependencies define the system's dependence on outside resources and the security policy outside the system being modeled. If a threat from an external dependency is disregarded, it may become a valid vulnerability. "In software systems, external dependencies often describe system wide functions such as algorithm inconsistency" [threat]. When defining external dependencies, the following data should be collected:

- Numerical ID - Each external dependency should have a unique identification number.
- Description - Write a description of the dependency.
- External security note reference - External security notes from one component can be cross-referenced with external dependencies on other components within the application.

2.2.3 External Security Notes

External security notes are provided to inform users of security and integration information for the system. External security notes can be a warning against misuse or a form of a guarantee that the system makes to the user. External security notes are used to validate external dependencies and can be used as a mitigation to a threat. However, this is not a good practice as it makes the end-user responsible for security. When defining external security notes, the following data should be collected:

- Numerical ID - Each external security note should have a unique identification number.
- Description - Write a description of the note.

2.2.4 Internal Security Notes

Internal security notes further define the threat model and explain concessions made in the design or implementation of system security. When defining internal security notes, the following data should be collected:

- Numerical ID - Each internal security note should have a unique identification number.
- Description - Write a description of the security concession and justification for the concession.

2.2.5 Implementation Assumptions

Implementation assumptions are created during the design phase and contain details of features that will be developed later. When defining implementation assumptions, the following data should be collected:

- Numerical ID - Each internal implementation assumption should have a

© SANS Institute 2000 - 2005 Author retains full rights.

unique identification number.

- Description - Write a description of the method of implementation.

2.2.6 Modeling the System

To create a useful model, the team needs to look at the application through an adversary's eyes. Modeling diagrams are a visual representation of how the subsystems operate and work together. The type of diagramming is not important; for the purposes of this document, Data Flow Diagrams will be used to model the system.

2.2.6.1 Modeling using Data Flow Diagrams (DFDs)

DFDs are a high-level way of focusing on data and how it flows through the system. DFDs are iterative and should be organized in a hierarchy that accurately reflects the system. The six basic shapes used in a DFD are listed below with their definitions[threat].

1. Process – represents a task that performs some action based on the data and should be numbered. Shown in figure 2.1.

Figure 2.1

2. Multiple process – has sub processes and each sub process node number should be prefixed by the parents node number, i.e. 1, 1.1, 1.1.2. Shown in figure 2.2.

Figure 2.2

3. External entity – is located outside the system and interacts at a system entry/exit point. It is either the source or destination of data and may only interact with process or multiple process. Shown in figure 2.3.

© SANS Institute 2000 - 2005 Author retains full rights.

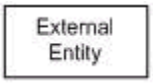

Figure 2.3

4. Data Store – place where data is saved or retrieved. May only interact with process or multiple processes. Shown in figure 2.4.

Figure 2.4

5. Data Flow – movement of data between elements. Shown in figure 2.5.

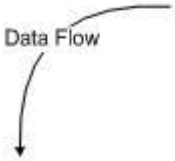

Figure 2.5

6. Trust Boundary – Boundary between trust levels or privileges. Shown in figure 2.6.

Figure 2.6

To create a DFD, first create an overall context diagram. This is the top level or root where the system is represented as a single multiple process or entry/exit point. Each node thereafter is a more detailed DFD describing other processes. Figure 2.7 shows an example of an overall context DFD.

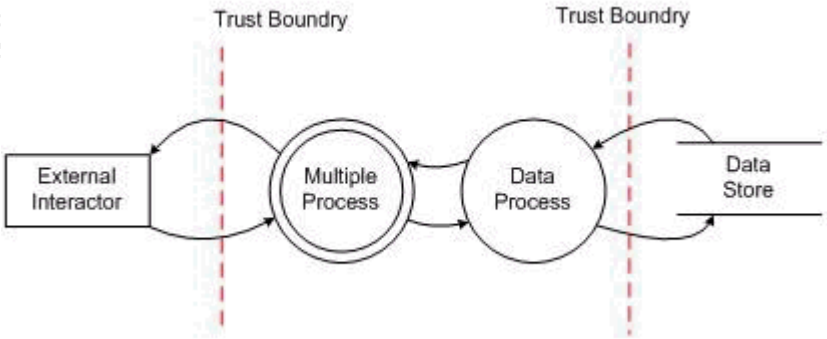

Figure 2.7

© SANS Institute 2000 - 2005 Author retains full rights.

2.3 Determining Threats

Once the DFD is created, it is used to determine what data is supplied to a node and the goals the adversary has for the application - an adversary must supply data to launch an attack. The goals are then used within the DFD to determine the threat paths, locate the entry/exit points and follow the data through the system. The threat path is defined as the sequence of any process nodes that perform security-critical processing [threat]. All areas where there is change or action on behalf of the data are susceptible to threats, i.e. process nodes.

2.3.1 Threat Profile

The threat profile is a security design specification for the system that describes the possible goals of the adversary and the vulnerabilities that exist as a result of those goals. Each threat in the profile must be prevented or mitigated. The threat profile consists of three main areas:

- Identify the threats.
- Investigate and analyzing the threats.
- Mitigate the vulnerabilities caused by the threats.

2.3.1.1 Identifying threats

Threat identification is a key to a secure system. Identifying threats consists of analyzing each entry/exit point, determining what critical security processing occurs at the entry/exit point and how it might be attacked. Threats are the goals of the adversary and for a threat to exist it must have a target asset. In the threat model document the threats are cross-referenced with the assets. To identify threats or goals, ask the following questions:

How can the adversary use or manipulate the asset to

- Modify or control the system?
- Retrieve information within the system?
- Manipulate information within the system?
- Cause the system to fail or become unusable?
- Gain additional rights?

[threat]

Can the adversary access the asset

- Without being audited?
- And skip any access control checks?
- And appear to be another user?

[threat]

© SANS Institute 2000 - 2005 Author retains full rights.

The next step in identifying threats is to classify the threat using the STRIDE model. The STRIDE model is documented in [secure].

STRIDE

- Spoofing - Allows an adversary to pose as another user, component, or other system that has an identity in the system being modeled.

- Tampering – The modification of data within the system to achieve a malicious goal.

- Repudiation - The ability of an adversary to deny performing some malicious activity because the system does not have sufficient proof otherwise.

- Information Disclosure - The exposure of protected data to a user that is not otherwise allowed access to that data.

- Denial of Service - Occurs when an adversary uses illegitimate means to assume a trust level with different privileges than he currently has.

Classifying the threat makes it easier to understand what the threat allows an attacker to do and aids in assigning priority.

2.3.1.2 Investigating and analyzing the threats using a threat tree

To identify vulnerable areas in the system and determine valid attacks paths, the threats identified in the previous step must be analyzed to discover where the system is susceptible to the threat. One method that works well for the investigation process is to build a threat tree.

Threat trees can be expressed graphically or as text in a threat modeling document. A threat tree consists of a root node or threat and child node(s). Each child node represents conditions needed for the adversary to find and identify the threat. Threat trees are used to determine the vulnerabilities associated with a threat. To identify a threat's vulnerabilities, begin at a node without any children and traverse it up to the root threat.

Another step in analyzing the threats is to determine the risk of the threat and the threat's conditions or child nodes by using the DREAD model. The DREAD model is documented in [secure]. When using the DREAD model, a threat modeling team calculates security risks as an average of numeric values assigned to each of five categories.

© SANS Institute 2000 - 2005 Author retains full rights.

- Damage potential – Ranks the extent of damage that occurs if a vulnerability is exploited.

- Reproducibility – Ranks how often an attempt at exploiting a vulnerability really works.

- Exploitability – Assigns a number to the effort required to exploit the vulnerability. In addition, exploitability considers preconditions such as whether the user must be authenticated.

- Affected users – A value characterizing the number of installed instances of the system that would be affected if an exploit became widely available.

- Discoverability – Measures the likelihood that, if unpatched, a vulnerability will be found by external security researchers, hackers, and the like.

Use a scale of 1-10 to rate each category, where 1 is the least probability of occurrence and the least damage potential. Add the rating of each category and divide the total by five to get an overall risk rating for each threat. The result can further be divided into three sections to generate a High, Medium or Low risk rating.

2.3.1.4 Vulnerability resolution and Mitigation

Up to this point, the threats have been identified and analyzed. If a threat is left unresolved, it will become a vulnerability. A vulnerability is present when a threat exists and the steps to mitigate it have not been implemented. To reduce the risk caused by threats, the team must analyze the conditions of each threat, using DREAD to assign a risk level, and identify a mitigation strategy to each condition.

Once the threat tree is completed, it can be used to identify attack paths, routes from a condition to a threat. Any attack path that is not mitigated will become a vulnerability. Figure 2.8 shows an example of a threat tree [threat].

© SANS Institute 2000 - 2005 Author retains full rights.

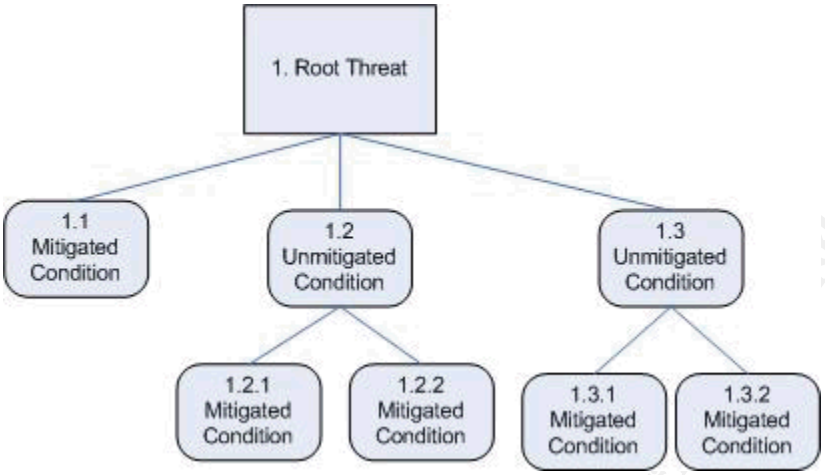

Figure 2.8

The threats, threat tree, vulnerabilities and mitigations are compiled into a threat modeling document that describes the threat profile of the system. The threat modeling document can be used in the design process as a security design specification and in the testing process to identify the vulnerable areas of the system.

Frank Swiderski, co-author of Threat Modeling [threat] , has developed a tool, which is currently free for download on the web [tool], to help design and implement a threat model.

More resources for threat modeling are listed at [resources] .

3.0 Conclusion

As the world increases its dependency on computers for critical information, the chances of applications being attacked are also increasing. Network security is no longer sufficient to secure an application. Security needs to be a part of the application design process. Implementing security during the design phase using the threat modeling process ensures that security is being designed into the application, thus decreasing the risk of an attack. Threat modeling will help you acknowledge, manage and communicate security risks across your application, ensuring that security has been designed into the system. Threat modeling is an iterative process and your threat model should evolve over time, changing to adapt to new threats and adjusting to changing business requirements.

© SANS Institute 2000 - 2005 Author retains full rights.

References

[threat] Frank Swiderski, Window Snyder, Threat Modeling, 2004, Microsoft Press

[secure] Michael Howard and David LeBlanc, Writing Secure Code, Second Edition, 2003, Microsoft Press

[collaborative] Jan Steffan, Markus Schumaker, Collaborative Attack Modeling, 2002, http://www.ito.tu-darmstadt.de/publs/pdf/sac2002.pdf

[model] Pete Lindstrom, Model Making Without Glue, August 2004, http://infosecuritymag.techtarget.com/ss/0,295796,sid6_iss446_art927,00.html

[security] J.D. Meier, Alex Mackman, Michael Dunner, Srinath Vasireddy, Ray Escamilla and Anandha Murukan, June 2003, http://www.msdn.microsoft.com/security/securecode/threatmodeling/default.aspx?pull=/library/en-us/dnnetsec/html/thcmch03.asp

[tool] Threat Modeling Tool, http://www.microsoft.com/downloads/details.aspx?familyid=62830f95-0e61-4f87-88a6-e7c663444ac1&displaylang=en

[resources] Threat Modeling Resources, http://cyberforge.com/weblog/aniltj/archive/0001/01/01/550.aspx

[matters] Dana Epp's Weblog, Why Threat Modeling Matters http://silverstr.ufies.org/blog/archives/000700.html

© SANS Institute 2000 - 2005, Author retains full rights.

Last Updated: April 25th, 2015

Upcoming SANS App Sec Training

SANS Security West 2015	**San Diego, CA**	**May 03, 2015 - May 12, 2015**	**Live Event**
Community SANS New Jersey DEV522	**New Jersey, NJ**	**May 04, 2015 - May 09, 2015**	**Community SANS**
SANS Secure Europe 2015	**Amsterdam, Netherlands**	**May 05, 2015 - May 23, 2015**	**Live Event**
Community SANS Seattle DEV522	**Seattle, WA**	**Jun 08, 2015 - Jun 13, 2015**	**Community SANS**
SANSFIRE 2015	**Baltimore, MD**	**Jun 13, 2015 - Jun 20, 2015**	**Live Event**
Community SANS Salt Lake City DEV541	**Salt Lake City, UT**	**Jun 15, 2015 - Jun 18, 2015**	**Community SANS**
SANS Capital City 2015	**Washington, DC**	**Jul 06, 2015 - Jul 11, 2015**	**Live Event**
SANS London in the Summer	**London, United Kingdom**	**Jul 13, 2015 - Jul 18, 2015**	**Live Event**
SANS vLive - DEV522: Defending Web Applications Security Essentials	**DEV522 - 201508,**	**Aug 03, 2015 - Sep 09, 2015**	**vLive**
SANS Chicago 2015	**Chicago, IL**	**Aug 30, 2015 - Sep 04, 2015**	**Live Event**
SANS Network Security 2015	**Las Vegas, NV**	**Sep 12, 2015 - Sep 21, 2015**	**Live Event**
SANS Gulf Region 2015	**Dubai, United Arab Emirates**	**Oct 17, 2015 - Oct 29, 2015**	**Live Event**
SANS vLive - DEV541: Secure Coding in Java/JEE: Developing Defensible Applications	**DEV541 - 201510,**	**Oct 26, 2015 - Nov 18, 2015**	**vLive**
SANS vLive - DEV544: Secure Coding in .NET: Developing Defensible Applications	**DEV544 - 201510,**	**Oct 27, 2015 - Nov 19, 2015**	**vLive**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**