Journal of Computer Applications ISSN: 0974 – 1925,
Volume-5, Issue EICA2012-5, February 10, 2012

An Approach to Threat Modeling in Web Application Security Analysis

Sreenivasa Rao B
Dept. of Computer Science & Engineering
CMJ University, Shillong, India
E-mail: basavala@gmail.com

Dr. Kumar N
ACE College of Engineering and Management,
Agra, India
E-mail: narendra.ibs@gmail.com

Abstract - In this paper, we investigate how threat modeling can be used as foundations for the specification of security requirements (SSR). It also discusses the importance of implementing web application security at design time. Threat modeling is a process to ensure that application security is implemented at design time. Threat modeling is a structured activity for identifying and evaluating application threats and vulnerabilities. This How To presents a question-driven approach to threat modeling that can help us to identifies security design problems early in the application design process. This approach allows you to quickly create a basic threat model for your web application scenario. Then you can use this threat model to help refine your application's design early and for communication among team members. The threat modeling approach is presented here focuses on identifying and addressing vulnerabilities. The security objectives, threats, and attacks that we identify in the early steps of the activity are the scoping mechanisms designed to help you find vulnerabilities in our web application. We can use the identified vulnerabilities to help shape our design and direct and scope your web application security testing.

Keywords: *Threat Modeling, Application Security, and security at design phase, Threat Mitigation, STRIDE security model, application security risk management.*

I. INTRODUCTION

The need to secure an application is imperative for use in today's internet 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. Threat modeling is a procedure for optimizing Application Internet Security by identifying objectives and vulnerabilities and then defining countermeasures to prevent or mitigate the effects of threats in the system. Threat modeling is a planned activity for identifying and assessing application threats and vulnerabilities that are actually critical in the application.

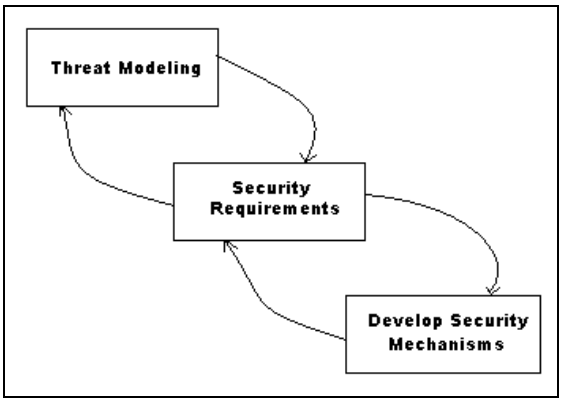

Figure 1: System Security Engineering

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.

II. THREAT MODELING PROCESS

The threat modeling is a process, varies depending on who is doing. The following process is based on research covering several different approaches. Based on my experience as an Application security Specialist at Yodlee Infotech Bangalore, I took what I believe to be best practices. The five major threat modeling steps are as shown in below figure:

A. *Identify security objectives:* Clear objectives help you to focus the threat modeling activity and determine how much effort to spend on subsequent steps.
B. *Create an application overview:* Itemizing your application's important characteristics and actors helps you to identify relevant threats during step 4.
C. *Decompose your application:* A detailed understanding of the mechanics of your application makes it easier for you to uncover more relevant and more detailed threats.

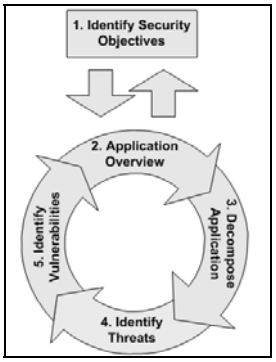

Figure 2: The iterative Threat Modeling Process.

D. *Identify threats:* Use details from steps 2 and 3 to identify threats relevant to your application scenario and context.
E. *Identify vulnerabilities.* Review the layers of your application to identify weaknesses related to your threats. Use vulnerability categories to help you focus on those areas where mistakes are most often made.

A. Security Objectives Identification

Security objectives are goals and constraints related to the confidentiality, integrity, and availability of your data and application. They include:

- *Confidentiality.* This includes protecting against unauthorized information disclosure.
- *Integrity.* This includes preventing unauthorized information changes.
- *Availability.* This includes providing the required services even while under attack.

By identifying our key security objectives, we can determine where to focus our efforts. Identifying our objectives also helps us to understand the goals of potential attackers and concentrate on those areas of an application that require closer attention. For example, if we identify customer account details as sensitive data that needs protecting, you can examine how securely the data is stored and how access to the data is controlled and audited.

The following are examples of some common security objectives:

- Prevent attackers from obtaining sensitive customer data, including users-Id, passwords and profile information.
- Meet service-level agreements for application availability.
- Protect the company's online business credibility.

B. Create an Application Overview

In this step, we outline what web application does. Our goal is to identify application's key functionality, characteristics, and clients. This will help you to identify relevant threats during step 4.
To create an application overview:

- Draw an end-to-end deployment scenario.
- Identify roles**.**
- Identify key usage scenarios**.**
- Identify technologies**.**
- Identify application security mechanisms**.**

The deployment diagram should generally include the following:

- *End-to-end deployment topology***.** Show the layout of the servers and indicate intranet, extranet, or Internet access. Start with logical network topologies, and then refine this to show physical topologies when you have those details. You can add or remove threats depending on your choice of specific physical topologies.
- *Logical layers.* Shows where the presentation layer, business layer, and data access layers reside. Refine this to include physical server boundaries when you know them.

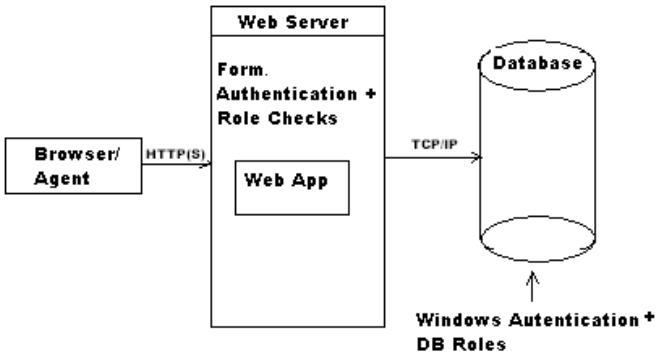

Figure 3: Common Web Application Architecture

- *Key components.* Show the important components within each logical layer. Refine this to include actual process and component boundaries when you know them.
- *Key services.* Identify any important services. Show these as processes when you know them.
- *Communication ports and protocols.* Show which servers, components, and services communicate with each other and how they do it. Show the specifics of inbound and outbound information packages when you know them.
- *Identities.* Show the main identities used for the application and any relevant service accounts if you have this information.
- *External dependencies.* Show the dependencies that your application has on external systems. Later in the modeling process, this will help you identify vulnerabilities that can arise if any assumptions you make about the external systems are false or if the external systems change in any way.*Identify Roles:* Identify application's roles: that is, identify who can do what within the application. What can end users do? What higher-privileged groups of users do you(application) have? For example, who can read data, who can update data, who can delete data? Etc.

Use role identification to determine both what is supposed to happen and what is not supposed to happen.

Identify Key Usage Scenarios: What are the important features in the application? What does it do? Use application's use cases to derive this information. Identify the dominant application functionality and usage, and capture the Create, Read, Update, and Delete aspects. Key features are often explained in the context of use cases. They can help you and others to understand how application is intended to be used and how it can be misused. Use cases help you identify data flows and provide focus when you identify threats later in the modeling process. Also identify what scenarios are out of scope, and use key scenarios to constrain the discussion. For example, you might decide that operational practices, such as backup and restore, are out of scope for the initial threat modeling exercise.

Identify Technologies: Where we can identify them, list the technologies and key features of the software and technologies that are used in the application. Identify the following:

- Operating systems
- Web server software

- Database server software
- Technologies used in the presentation, business, and data access layers
- Development languages

Identifying technologies helps us to focus on technology-specific threats later in the threat modeling activity. It also helps you determine the correct and most appropriate mitigation techniques.

Identify Application Security Mechanisms: Identify any key points that you know about the following:

- Input and data validation
- Authentication
- Authorization
- Configuration management
- Sensitive data
- Session management
- Cryptography
- Parameter manipulation
- Exception management
- Auditing and logging

The purpose of this effort is to identify interesting details and be able to add detail where ever is necessary.

C. Decompose Your Application

In this step, we break down your application to identify trust boundaries, data flows, entry points, and exit points. The more you know about the mechanics of your application, the easier it is to uncover threats and discover vulnerabilities. To decompose your application:

- Identify trust boundaries.
- Identify data flows.
- Identify entry points.
- Identify exit points.

Identify trust boundaries: Identify your application's trust boundaries to help us focus on your analysis on areas of concern. Trust boundaries indicate where trust levels change. You can think of trust from the perspective of confidentiality and integrity.

To help identify trust boundaries:

- Start by identifying your outer system boundaries. For example, your application can write to files on server X, it can make calls to the database on server Y, and it can call Web service Z. This defines your system boundary.
- Identify access control points or the key places where access requires additional privileges or role membership. For example, a particular page might be restricted to managers. The page requires authenticated access and also requires that the caller is a member of a particular role.
- Identify trust boundaries from a data flow perspective. For each subsystem, consider whether the upstream data flow or user input is trusted, and if it is not, consider how the

data flow and input can be authenticated and authorized. Knowing which entry points exist between trust boundaries allows you to focus your threat identification on these key entry points. For example, you are likely to have to perform more validation on data passed through an entry point at a trust boundary.

Identify Data Flows: Trace your application's data input through the application from entry to exit. You do this to understand how your application interacts with external systems and clients and how internal components interact. Pay particular attention to data flow across trust boundaries and how that data is validated at the trust boundary entry point. Also pay close attention to sensitive data items and how these flow through your system, where they are passed over a network, and where they are persisted.

A good approach is to start at the highest level and then deconstruct the application by analyzing the data flow between individual subsystems. For example, start by analyzing the data flow between your Web application, middle tier servers, and database server. Then consider page-to-page and component-to-component data flows.

Identify Entry Points: The entry points of an application also serve as entry points for attacks. Entry points can include the front-end or user interface of web application listening for HTTP requests. This entry point is intended to be exposed to clients. Other entry points, such as internal entry points exposed by subcomponents across the layers of your application, may exist only to support internal communication with other components. However, you should know where these are and what types of input they receive in case an attacker manages to bypass the front door of the application and directly attack an internal entry point. Additional levels of checking provides defense in depth but may be costly in terms of money and performance.

Consider the trust levels required to access an entry point and the type of functionality exposed by the entry point. In the threat modeling activity, focus your attention on entry points that expose privileged functionality, such as administration interfaces.

Identify Exit Points: Identify the points where your application sends data to the client or to external systems. Prioritize the exit points where your application writes data that includes client input or includes data from un-trusted sources, such as shared databases.

D. Identify Threats

In this step, we identify threats and attacks that might affect of an application and compromise security objectives. These threats are the bad effects that could happen in the application. To conduct this identification process, bring members of the development and test teams together to conduct an informed brainstorming session. Use a whiteboard to identify potential threats. Ideally, the team consists of application architects, security professionals, developers, testers, and system administrators.

You can use two basic approaches:

- *Start with common threats and attacks:* With this approach, we start with a list of common threats grouped by application vulnerability categories. Next, apply the threat list in the application architecture. While doing this, use the data you gathered. For example, use the identified scenarios to review data flows, paying particular attention to entry points and where trust boundaries are crossed. You will be able to eliminate some

threats immediately because they do not apply to your application and its use cases. Use the "Cheat Sheet: Web Application Security Frame" as a starting point.

- *Use a question-driven approach:* A question-driven approach can help us to identify relevant threats and attacks. The STRIDE categorization includes broad categories of threats, such as spoofing, tampering, repudiation, information disclosure, and denial of service. You can use the STRIDE model to ask questions related to each aspect of the architecture and design of the application. This is a goal-based approach, where you consider the goals of an attacker. For example, could an attacker spoof an identity to access your server or Web application? Could someone tamper with data over the network or in a data store? Is sensitive information disclosed when you report an error message or log an event? Could someone deny service etc.?

While identifying threats, examine the application tier by tier, layer by layer, and feature by feature. By focusing on vulnerability categories, you focus on the areas where security mistakes are most frequently made. The threats identified at this stage do not necessarily indicate vulnerabilities. Identify potential threats and the actions that an attacker might try to use to exploit the application. During this step, we perform the following tasks:

- Identify common threats and attacks.
- Identify threats along use cases.
- Identify threats along data flows.

Identify Common Threats and Attacks: There are a number of common threats and attacks that rely on common vulnerabilities. As a starting point, use the companion document, "Cheat Sheet: Web Application Security Frame." The cheat sheet will help you identify threats and attacks relevant to your application.

Web Application Security Frame:
The following vulnerability categories were developed by security experts who have examined and analysed the top security issues across many Web applications. They have been refined with input from security consultants, product support engineers, customers, and business partners. This section identifies a set of key questions to ask for each category.

- Authentication
- Authorization
- Input and Data Validation
- Configuration Management
- Sensitive Data
- Session Management
- Cryptography
- Parameter Manipulation
- Exception Management
- Auditing and Logging

Identify Threats along Use Cases:
Examine each of your application's key use cases that you identified earlier, and examine ways in which a user could maliciously or unintentionally coerce the application into performing an unauthorized operation or into disclosing sensitive or private data.
Ask questions and try to think as an attacker would. Examples of the types of question you should ask include the following:

- How can a client inject malicious input here?
- Is data being written out based on user input or on un-validated user input?
- How could an attacker manipulate session data?
- How could an attacker obtain sensitive data as it is passed over the network?
- How could an attacker bypass your authorization checks?

Identify Threats along Data Flows:
Review the key use cases and scenarios, and analyze the data flows. Analyze the data flow between individual components in your architecture. Data flow across trust boundaries is particularly important. A piece of code should assume that any data from outside the code's trust boundary is malicious. The code should perform thorough validation of the data. When identifying threats associated with data flows, ask the following questions:

- How does data flow from the front end to the back end of your application?
- Which components call which components?
- What does valid data look like?
- Where is validation performed?
- How is the data constrained?
- How is data validated against expected length, range, format, and type?
- What sensitive data is passed between components and across networks, and how is that data secured while in transit?

E. Identify Vulnerabilities

In this step, we review the Web application security frame and explicitly look for vulnerabilities. Focus on the vulnerability categories as you did while identifying threats in the previous step. A useful way of proceeding is to examine in the application layer by layer, considering each of the vulnerability categories in each layer as follows.

- Authentication
- Authorization
- Input and Data Validation
- Configuration Management
- Sensitive Data
- Session Management
- Cryptography
- Parameter Manipulation
- Exception Management
- Auditing and Logging

III. CONCLUSION

In this paper we propose threat modeling as an essential foundation for defining security requirements of computer systems. Without identifying threats, it is impossible to provide assurance for the system and justify security measures taken. We have presented our view on best practices for the threat modeling for both software applications and complex systems. We have discussed potential threats to the application and requirement for the threat modeling process. Threat modeling process provides a security framework to secure the web application.

Based on the study, it can be concluded that modeling the application for present and future threats and vulnerabilities can provide great level of security to an organization. Security policies can be a very helpful practice in protecting applications from the threats vulnerabilities and maintains Confidentiality, Integrity and Availability of the system.

Finally, being ever cautious and watchful will keep the attackers at holiday. So, it is always better to hide yourself from Hacker, Cracker and Script Kiddies to survive in the today's technological environment.

REFERENCES

1. Understanding and Developing a Threat Assessment Model, Stilianos Vidalis and Andrew Blyth, University of Glamorgan.
2. J.D.Nosworthy, A Practical Risk Analysis Approach: Managing BCM Risk Computers & Security, 2000. Pg. 596-614
3. Analyzing Threat Agents & Their Attributes, Dr. Stilianos Vidalis, Dr. Andrew Jones, University of Glamorgan.
4. Electronic Warfare Association – Australia (URL:www.ewa-australia.com/infosec-stream2.htm)
5. http://searchsecurity.techtarget.com/ sDefinition/0,290660,sid14_gci1166533,
6. An Introduction to FAIR: The Factor Analysis of Information Risk (FAIR) Framework.
7. http://www.itsecurity.com/features/it- security-audit-010407/
8. http://msdn2.microsoft.com/en-us/library/ ms978516.aspx
9. Frank Swiderski, Window Snyder, Threat Modeling, 2004, Microsoft Press
10. Michael Howard and David LeBlanc, Writing Secure Code, Second Edition, 2003, Microsoft Press
11. Jan Steffan, Markus Schumaker, Collaborative Attack Modeling, 2002, http:// www. ito.tu-darmstadt.de/publs/pdf/sac2002.pdf
12. Pete Lindstrom, Model Making Without Glue, August 2004, http:// infosecuritymag. techtarget.com/ss/0295796,sid6_iss446_art927,00.html
13. 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-s/dnnetsec/html/thcmch03.asp
14. Threat Modeling Tool, http://www.microsoft.com/downloads/details.aspxfamilyid =62830f95-0e61-4f87-88a6-e7c663444ac1&displaylang=enThreatModeling Resources http://cyberforge.com/weblog/aniltj/archive/0001/01/01/550.aspx .
15. Dana Epp's Weblog, Why Threat Modeling Matters http:// silverstr .ufies .org /blog/archives/000700.html.