Lecture 4: THREAT MODELING

author: Elena Oat

Summary

This lecture is about finding the threats specific to a system, as well as identifying its most "sensitive" from the security point of view parts. The notes provide an overview of the methodology of threat analysis. In addition, some practical examples are included to clarify further the ideas.

Threat vs. vulnerability

There’s a distinction between these two terms. A threat is something that can possibly happen and impact in a negative way you and the system you are trying to protect. A vulnerability, on the other hand, is a security weakness that already exists and specific steps could be taken in order to remove it.

A threat cannot be completely removed. However, the exposure to it can be minimized.

For instance, any person using public transportation has a threat of being pickpocketed. Moreover, a person who has a backpack with an open pocket has a vulnerability. A thief can easily steal valuable items from this backpack. However, this vulnerability could be removed by closing the pocket [1].

Threat modeling: definition

Threat modeling means identifying systematically all of the things that could go wrong in a system. More precisely, we are trying to identify all of the threats from the security point of view. Even though threat modeling is not a new practice, its views have changed quite much recently. Namely, it is recommended to analyze a system from the point of view of an attacker, instead of a defender [2].

Threat analysis, or risk analysis, as it is sometimes called, could be performed on several levels. Among these are business and technical. Some of the threats of one level may or may not be a part of the other level.

Risk analysis on the business level should be done with a company business manager (e.g., owner, product manager), who knows all of the details related to information flow, assets, use and misuse cases, users of the system, etc. Analysis on this level might be useful for the overall threat modeling process, as some things might not be evident from the architectural analysis alone.
Risk analysis on the technical level is also called *architectural risk analysis*. The goal of this activity consists in identifying system components, as well as the technologies used, and listing the threats specific to them. This should also include physical deployment characteristics. All these are represented on a diagram, which shows clearly the components and the relations.

Threat modeling is sometimes represented in form of threat trees. However, these do not suit well technical risk analysis. But they are useful in some cases when drawing the model on the business level.

**Finding answers to questions**

The team working on the risk analysis aims at discovering all of the inner workings of the system. It may be necessary to know how and where is the data processed, in order to protect from XSS attacks, for instance.

Another question that has to be asked is related to the data confidentiality and integrity. In case these are important, it is essential to know the data storage and its properties. Configuration files, as well as cryptographic keys are also data and need to be properly protected.

*Attack surface* is another significant part of the risk analysis. We have to find the interfaces through which the attacks are possible. By an interface we mean the system boundary through which the data flows in and/or out.

Security analysis should remove any abstractions and reveal the details of the protocols, technologies, components, messages exchanged, etc., unless these are undoubtedly irrelevant to the system’s security. In other words, the black boxes should be opened and their contents should be unpacked. Because important details might be hidden there.

Namely, following are the details that should be known once the threat modeling is complete:

- statically linked libraries
- dynamic libraries and plugins
- frameworks used
- virtual machines
- implementation languages
- data storage and its location
- data flow
- type of data that is stored or processed (format, content, encoding)
• location of sensitive data, such as private keys.

**Sketching it all**

Optimally, the process of architectural risk analysis should be done by drawing a diagram or a graph where all of the above mentioned parts are represented. While drawing it, people participating in the process usually comment on things that are drawn. Pictures are very efficient in identifying the details, as human brain cannot keep in mind too many details simultaneously. However, sketching the details helps in analyzing one component at a time.

Among the most practical diagrams used in threat modeling are *data flow diagrams* (DFD). Using these, the team can visually represent the processing boxes and the data flowing between them.

![Figure 1: Example of a data flow diagram from lecture material](image)

Message sequence charts (MSC) are used in threat modeling too. Their role is to dissect complex message exchanges between the parties and represent each elementary one. However, UML and class diagrams are usually not so useful for this process. Breaking down the components of a system has to stop somewhere. It usually stops at the point where a component is

3
out of the team’s control. For instance, if the goal is to analyze a native app, another native app with which it interacts would be a black box.

![Figure 2: Example of a message sequence chart from lecture material](image)

Some of the elements represented in a data flow diagram could be humans. These may be end-users, attackers, administrators and they must be represented too. In fact, some of these are among the components that the team usually forgets to draw. Other things that are usually forgotten are admin interfaces, cryptographic key storage, configuration files, load balancers, as well as the code executed in the browser.

The elements of a data flow diagram are entities, each of which have their security boundaries. Thus, each box and all its contents are inside its own security boundaries. For example, a process running in an OS could be represented as a box, as it has its security boundaries provided by this OS. That’s because modern operating systems isolate the memory of a process from other processes running on the same system. Conversely, threads that belong to a process share the memory with each other.

If a security boundary can be compromised, it should be assumed that everything inside of this boundary can be compromised. Even though this might be not true, it is safer to consider so. For example, if one can execute code on a machine, it also means they are in complete control of this machine. Once something inside of a security boundary is proven to be insecure, it may be wise to stop analyzing and securing the other elements inside the same boundary. The whole box just becomes an insecure element. Otherwise, people may spend a lot of time on correcting small issues inside this box, but this wouldn’t help in the big picture, as the box would still have weaknesses.
Some of the typical boundaries found in the systems are the boundaries of a physical or a virtual machine, the JVM, processes.

**Taint analysis**

When doing threat analysis it is important to know where each protocol communication ends. This endpoint is, in fact, an attack surface. One example of endpoint is where communication over HTTP ends. For instance, let’s suppose that we design a system which provides services to users, that need to authenticate before using them. The authentication happens at the application back-end. However, HTTP is handled by nginx. Thus, the endpoint for HTTP would be nginx.

When doing the taint analysis as a part of threat modeling, the team will identify the data flows and the components in which these data flows end. If an attack is possible on a data flow, it means the whole path is compromised. During this phase we also try to identify the places the attacks are probable and mark them for further review later. Some elements of the attack surface are more prone to be compromised than others, depending on the trustworthiness of the data source. Thus, these elements could be prioritized according to the criteria of trustworthiness.
What are we trying to protect from?

After the data flows, as well storage were identified we need to decide on what type of security services they need. When making a decision, we can consider the CIA - Confidentiality, Integrity, Availability model. Some of our components might need confidentiality and integrity, while others just availability. For instance, posts found on a forum are rarely hidden. In addition, their modification presents little incentive. However, people would like to access the forum at any time. Thus, such a system requires availability, but doesn’t usually care much about confidentiality and integrity of the posts.

STRIDE

Microsoft has also developed a model that works well for data flow analysis - STRIDE. This acronym stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege.

The team of people performing the threat analysis could use STRIDE to analyze a particular data flow or a data storage. Thus, a set of questions are asked for each of the letters of the STRIDE acronym. It is most likely that for each component several categories would apply [4].

For example, some of the relevant questions related to spoofing would be: "Do the users care that the website they are visiting is real and not a spoofed one?", or "Does the server care who are the users and what are their privileges?".

Among the questions related to repudiation could be "Should the users be able to deny their visits to the website?", or, the opposite, "Should we ensure that the users cannot deny their visits and contribution?".

Relevant question for the elevation of privilege part would be "Could the users somehow misuse the input interface and execute some code that would allow them to do unauthorized things?".

Once all of the threats to a system have been identified and mitigated we can argue that the system is secure [5]. However, this might not be always true. If the components are safe when functioning alone, it does not necessarily mean that being combined they create a secure system. In fact, some of the issues appear right then when we combine these components.

The process of threat analysis should be continuous, meaning it shouldn’t be done only once when an application is being designed. Instead, this should be repeated with time, as new features are introduced and the environment of the system has changed.
Concluding notes

Some of the findings of the threat analysis could be discarded or, conversely, emphasized. One has to estimate how much would it cost the company to fix some of the findings versus how big would be the impact if those weaknesses are exploited. If some of the risks are rather high and carry quite serious consequences, but require a lot of spendings, it would be wise to consult with the company management. In case the management rejects spending money on fixing the issues, it would be best to get a written note on this decision, so that one doesn’t end up as a scapegoat in case the things go wrong.

References


