Threat Modeling

Sources:

- *Writing Secure Code* by M. Howard and P. LeBlanc, Microsoft Press
- *Threat Modeling*, by J.D. Meier et al, Microsoft, msdn.microsoft.com/en-us/library/ff648644.aspx

A *threat model* helps to reveal the highest security risks to a software product.

It also helps to indicate how attacks can manifest.

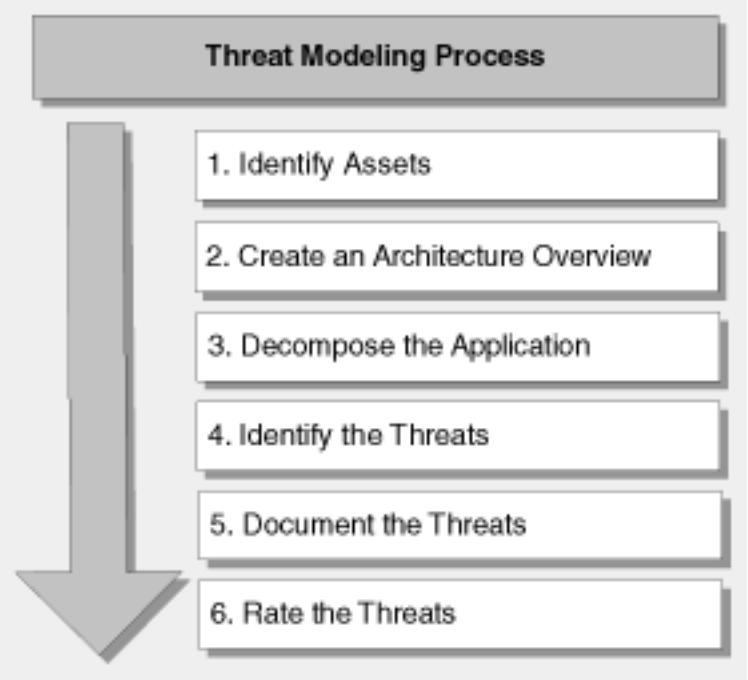
Microsoft found threat modeling to be the most important part of the design process during its 2002 security push.

Other benefits of threat modeling:

- Threat models help you understand your application better.
- They help you find bugs. (Microsoft: 50% of bugs are found during threat modeling.)
- They help new team members understand the application.
- They are helpful to other product teams that build on your product.
- They are useful for testers who should test against the threat model.

Threat modeling process:

- Assemble the threat modeling team.
- Decompose the application.
- Determine threats to the system.
- Rank threats by decreasing risk.
- Choose how to respond to the threats.
- Choose techniques to mitigate the threats.
- Choose appropriate technologies for these techniques.



[http://msdn.microsoft.com/en-us/library/ff648644.aspx]

Assembling the Threat Modeling Team

The team leader should be the most “security savvy” person on the team.

At least one member should be included from each development discipline:

- Requirements
- Design
- Coding
- Test
- Documentation

Outside members give fresh perspective.

There should be no more than 10 members.

Marketing and sales people can be helpful for their perspective and for educating users.

Team members are instructed that the goals of the meeting are:

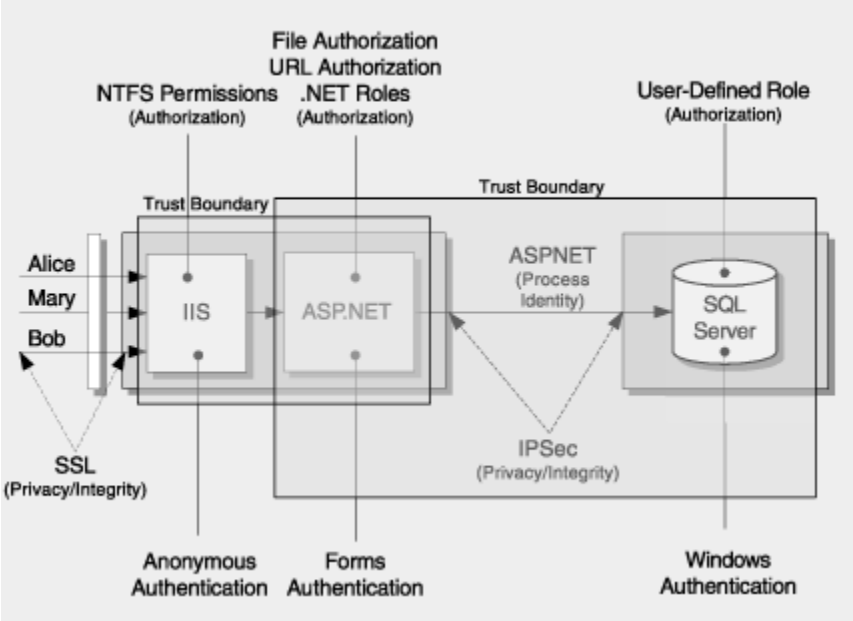
- To identify the components of the application and their interactions
- To identify security threats
- *Not* to fix problems

Create an Architecture Overview

Steps:

- Identify what the application does
- Create an architecture diagram
  - Describes composition and structure of application
  - Describes physical deployment characteristics
- Identify the technologies

Example: architecture diagram



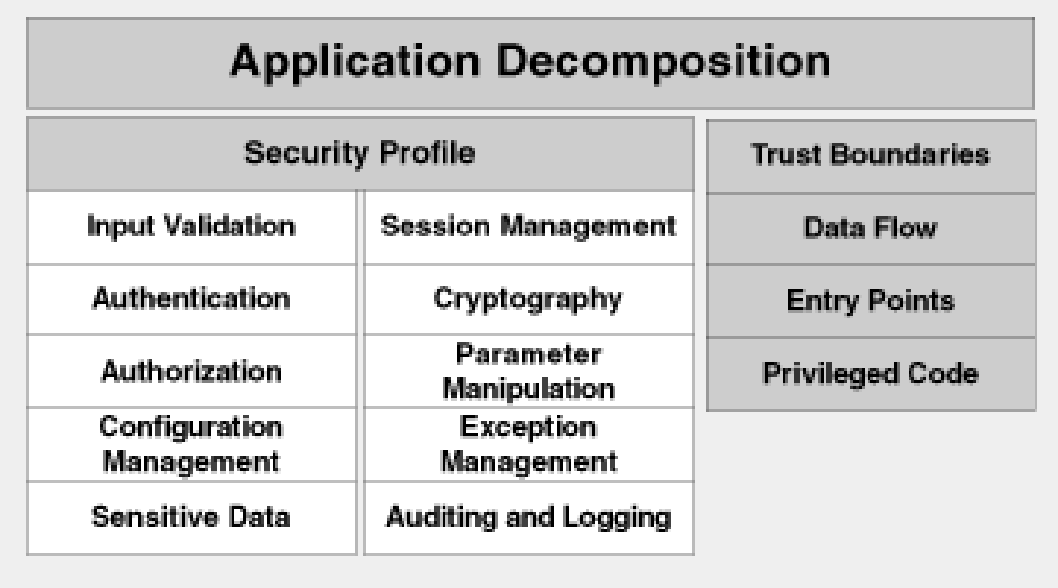
[http://msdn.microsoft.com/en-us/library/ff648644.aspx]

Decompose the Application

Data flow diagrams are used to show interactions between processes, “interactors”, and data stores.

UML diagrams (e.g., activity diagrams) can also be used.

Targets for decomposition process:



[http://msdn.microsoft.com/en-us/library/ff648644.aspx]

First phase of decomposition:

1. Determine the scope of the system.

2. Determine the boundaries between trusted and untrusted components.

DFDs define the scope of a system using a high-level *context diagram*.

This usually has one process and no data stores.

Additional DFDs are used to "drill down" to more detailed levels.

Example: context diagram for a simple, web-based payroll app

See Figures, 4-4, 4-5 of H&L.

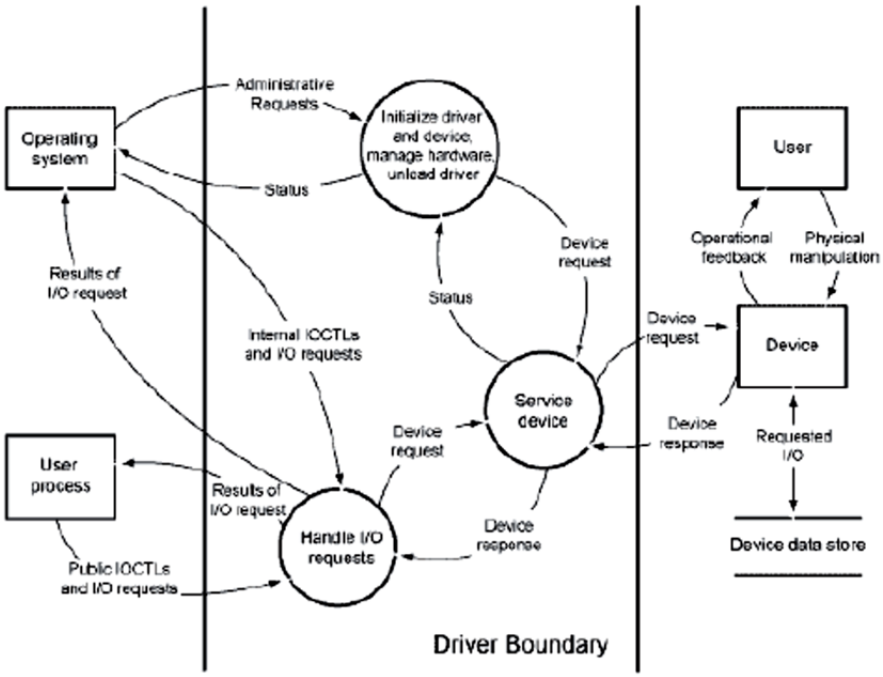


Figure 1. Sample data flow diagram for hypothetical kernel-mode driver.

[resist.isti.cnr.it/free_slides/security/williams/RiskBasedSecurityTesting.pdf]

Rules for defining scope of DFD:

- Ignore the inner workings of the system.
- Consider events and requests the system must respond to.
- Consider what responses a process generates.
- Identify the data sources for each request and response.
- Ascertain the recipient of each response.

Rules for creating DFDs:

- Each process must have at least one data flow entering and at least one exiting.
- All data flows start or stop at a process.
- Data flows may pass through a data store.
- Process names are verb phrases.
- Data flow names are noun phrases.
- External entity or interactor names are nouns.
- Data store names are nouns.

See Figure 4-5 of H&L – high-level physical view of payroll application.

See Table 4-1 of H&L – main components and users of sample payroll application.

Determine Threats to the System

The components identified during the decomposition process are the *threat targets* of the threat model.

Each component should be evaluated to identify potential threats in the following categories (STRIDE):

- Spoofing identity
- Tampering with data
- Repudiation
- Information disclosure
- Denial of service
- Elevation of privilege

Threat Trees

These describe the decision process an attacker would use to compromise a component.

See Figure 4-8 of H&L – threat tree for payroll application.

The top box represents the ultimate threat.

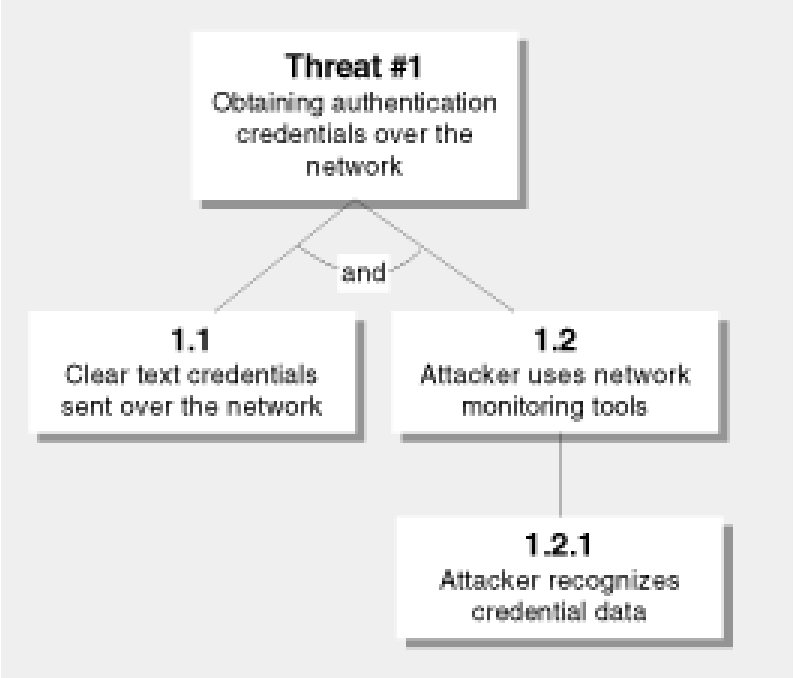
The boxes below it are the steps in realizing it.

They may be combined with AND and OR (default).

Dotted lines can be used to show the least likely attack points.

Solid lines can be used to show the most likely ones.

A circle can be placed below a node to show why the threat it represents is mitigated.



[http://msdn.microsoft.com/en-us/library/ff648644.aspx]

See Figure 4-9 of H&L.

The following items should be determined and recorded when modeling a threat:

- Title
- Threat target
- Threat types (STRIDE)
- Risk
- Attack tree
- Mitigation techniques and status
- Bug number (optional)

Ranking Threats

After threat trees are created, threats should be ranked by decreasing risks.

A simple way to calculate risk is by multiplying the criticality (damage potential) of the vulnerability by the likelihood of occurrence:

Risk = Criticality * Likelihood

1 is low criticality and 10 is high criticality.

Dread

Another way to determine the risk is to rank bugs using the DREAD method:

- Damage potential
- Reproducibility
- Exploitability
- Affected percentage of users
- Discoverability (by attackers)

The DREAD rating is determined by averaging these numbers.

	Rating	High (3)	Medium (2)	Low (1)
D	Damage potential	The attacker can subvert the security system; get full trust authorization; run as administrator; upload content.	Leaking sensitive information	Leaking trivial information
R	Reproducibility	The attack can be reproduced every time and does not require a timing window.	The attack can be reproduced, but only with a timing window and a particular race situation.	The attack is very difficult to reproduce, even with knowledge of the security hole.
E	Exploitability	A novice programmer could make the attack in a short time.	A skilled programmer could make the attack, then repeat the steps.	The attack requires an extremely skilled person and in-depth knowledge every time to exploit.
A	Affected users	All users, default configuration, key customers	Some users, non-default configuration	Very small percentage of users, obscure feature; affects anonymous users
D	Discoverability	Published information explains the attack. The vulnerability is found in the most	The vulnerability is in a seldom-used part of the product, and only a few users should come across it. It	The bug is obscure, and it is unlikely that users will work out damage

		commonly used feature and is very noticeable.	would take some thinking to see malicious use.	potential.

[msdn.microsoft.com/en-us/library/ff648644.aspx#c03618429_010]

Example Threat Descriptions

Threat Description	**Attacker obtains authentication credentials by monitoring the network**
Threat target	Web application user authentication process
Risk	
Attack techniques	Use of network monitoring software
Countermeasures	Use SSL to provide encrypted channel

[msdn.microsoft.com/en-us/library/ff648644.aspx#c03618429_010]

See also:

- Table 4-4 of H&L (p. 98).
- Table 4-5 of H&L (p. 99).
- Table 4-6 of H&L (p. 100).
- Figure 4-10 of H&L (p. 100).
- Figure 4-11 of H&L (p. 101).

To obtain an overall risk rating, you must consider the most likely path of attack (path of least resistance).

Example: Figure 4-10 of H&L.

Mitigating Threats

The next phase is mitigating threats that have been identified.

This is a two-step process:

1. Determine which techniques can help.

2. Choose the appropriate technologies.

See Table 4-10 of H&L – techniques to mitigate STRIDE threats.