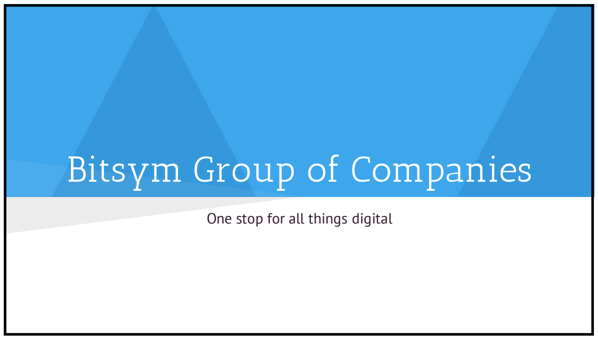


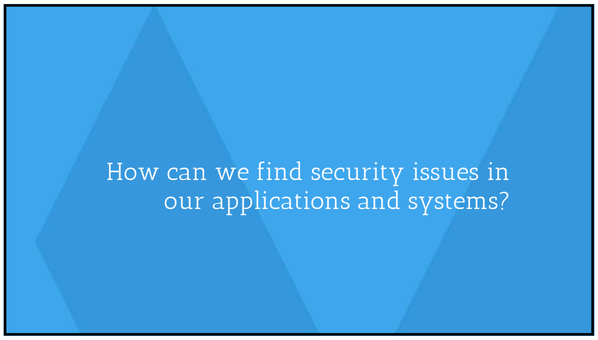
# Bitsym Group of Companies

One stop for all things digital

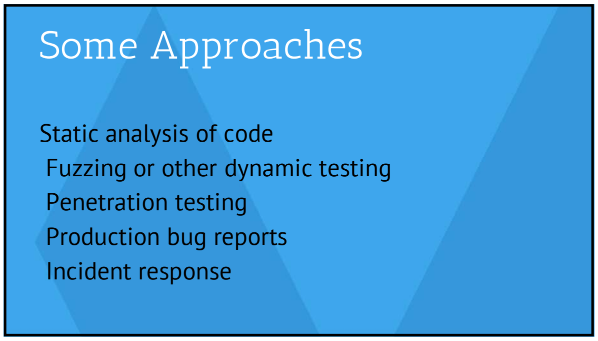
# Threats Modeling

## Agenda

**Introduction**
**Strategies for Threat Modeling**
**STRIDE**
**Attack Trees**
**Threat Modeling Tools**



How can we find security issues in our applications and systems?



## Some Approaches

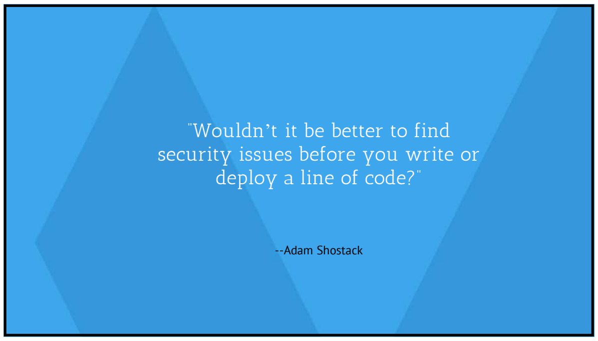
Static analysis of code

Fuzzing or other dynamic testing

Penetration testing

Production bug reports

Incident response



"Wouldn't it be better to find security issues before you write or deploy a line of code?"

--Adam Shostack

## Ways to Find Security Issues

Threat modeling! Why??

- Improve efficiency
  - Think about security issues early
  - Invest effort more wisely
- Understand requirements better
  - Bring security and development together
  - Shared, maintainable, understanding of risks
- Avoid writing bugs into the code
  - Avoid costs of rework
- Improved stakeholder confidence

## What is Threat Modeling??

Threat modeling is a structured approach of identifying and prioritizing potential threats to a system and determining the value that potential mitigations would have in reducing or neutralizing those threats.

Threat-modeling methods are used to create

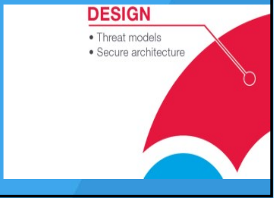
- an abstraction of the system
- profiles of potential attackers, including their goals and methods
- a catalog of potential threats that may arise

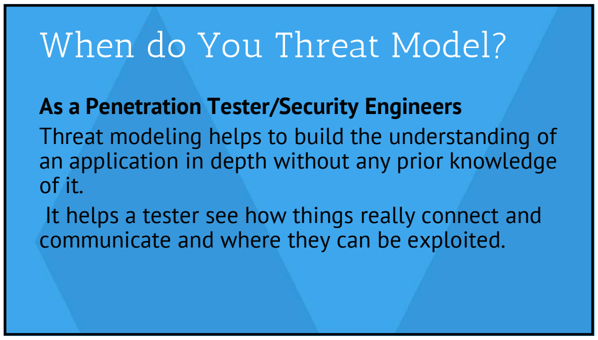
## When do You Threat Model?

**As a developer**

Threat modeling should initially be done as early as possible in the development life cycle, revisited any time there is a change to the system's architecture, and after any security incident or new vulnerabilities are introduced.

In the image you can see that it as part of the Design point of the development life cycle, but don't *only* look at it once and then never again!



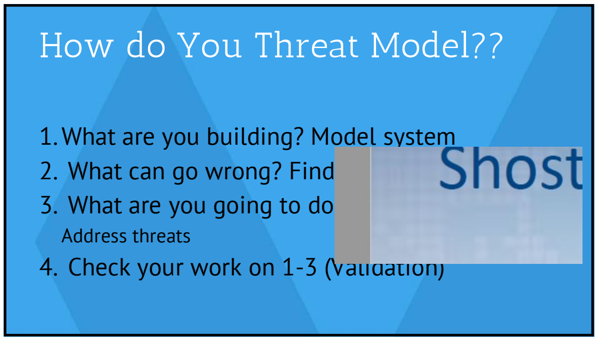


## When do You Threat Model?

**As a Penetration Tester/Security Engineers**

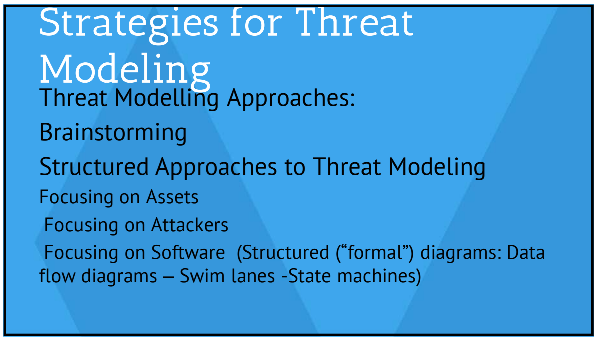
Threat modeling helps to build the understanding of an application in depth without any prior knowledge of it.

It helps a tester see how things really connect and communicate and where they can be exploited.



## How do You Threat Model??

1. What are you building? Model system
2. What can go wrong? Find
3. What are you going to do
   Address threats
4. Check your work on 1-3 (validation)



## Strategies for Threat Modeling

Threat Modelling Approaches:

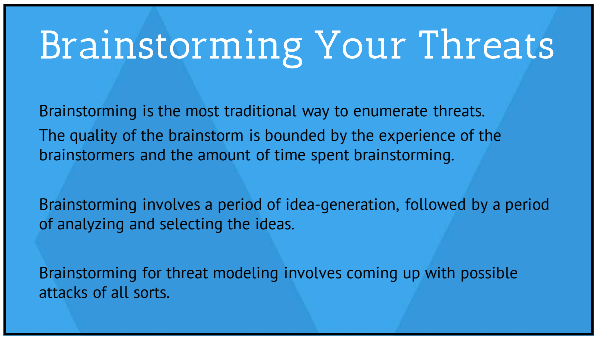
Brainstorming

Structured Approaches to Threat Modeling

Focusing on Assets

Focusing on Attackers

Focusing on Software (Structured ("formal") diagrams: Data flow diagrams – Swim lanes -State machines)



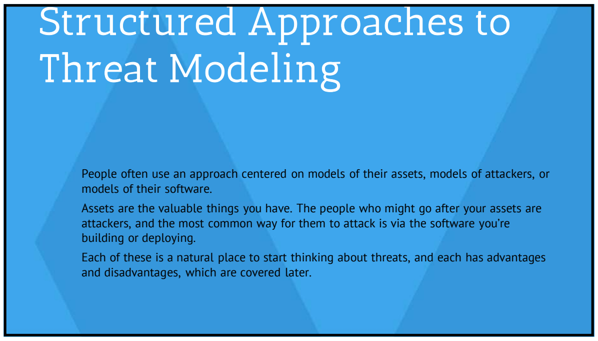
## Brainstorming Your Threats

Brainstorming is the most traditional way to enumerate threats.

The quality of the brainstorm is bounded by the experience of the brainstormers and the amount of time spent brainstorming.

Brainstorming involves a period of idea-generation, followed by a period of analyzing and selecting the ideas.

Brainstorming for threat modeling involves coming up with possible attacks of all sorts.

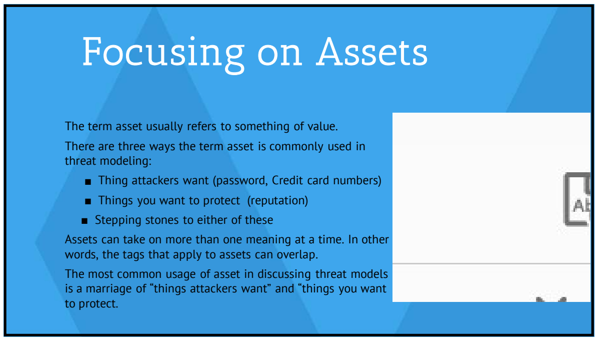


## Structured Approaches to Threat Modeling

People often use an approach centered on models of their assets, models of attackers, or models of their software.

Assets are the valuable things you have. The people who might go after your assets are attackers, and the most common way for them to attack is via the software you're building or deploying.

Each of these is a natural place to start thinking about threats, and each has advantages and disadvantages, which are covered later.



## Focusing on Assets

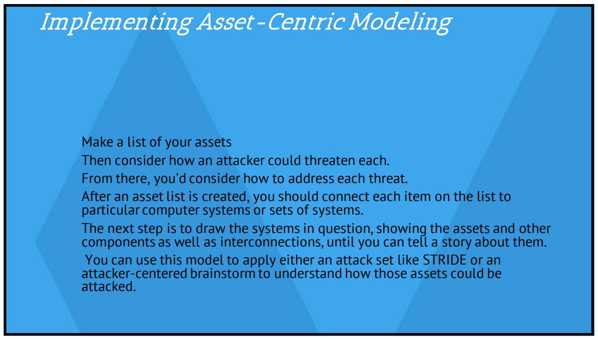
The term asset usually refers to something of value.

There are three ways the term asset is commonly used in threat modeling:

- Thing attackers want (password, Credit card numbers)
- Things you want to protect (reputation)
- Stepping stones to either of these

Assets can take on more than one meaning at a time. In other words, the tags that apply to assets can overlap.

The most common usage of asset in discussing threat models is a marriage of "things attackers want" and "things you want to protect.



## *Implementing Asset-Centric Modeling*

Make a list of your assets

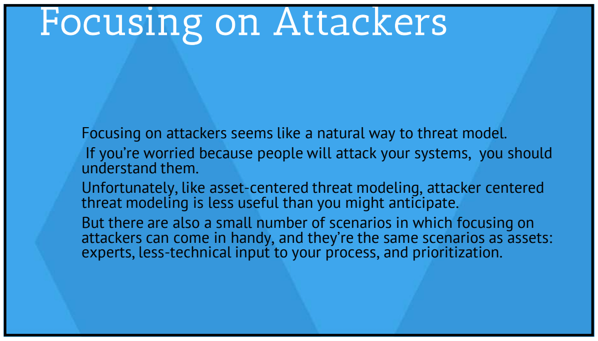
Then consider how an attacker could threaten each.

From there, you'd consider how to address each threat.

After an asset list is created, you should connect each item on the list to particular computer systems or sets of systems.

The next step is to draw the systems in question, showing the assets and other components as well as interconnections, until you can tell a story about them.

You can use this model to apply either an attack set like STRIDE or an attacker-centered brainstorm to understand how those assets could be attacked.



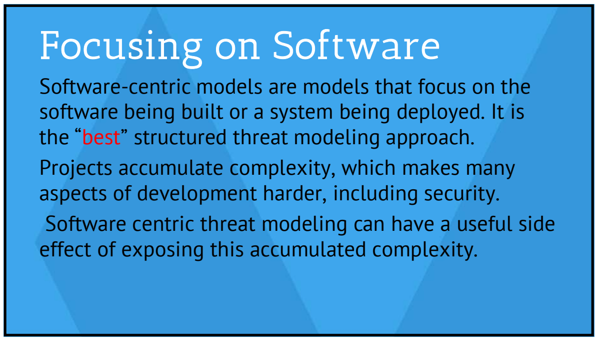
## Focusing on Attackers

Focusing on attackers seems like a natural way to threat model.

If you're worried because people will attack your systems, you should understand them.

Unfortunately, like asset-centered threat modeling, attacker centered threat modeling is less useful than you might anticipate.

But there are also a small number of scenarios in which focusing on attackers can come in handy, and they're the same scenarios as assets: experts, less-technical input to your process, and prioritization.



## Focusing on Software

Software-centric models are models that focus on the software being built or a system being deployed. It is the "best" structured threat modeling approach.

Projects accumulate complexity, which makes many aspects of development harder, including security.

Software centric threat modeling can have a useful side effect of exposing this accumulated complexity.

## Modern DFD Model

The changes made from classic DFDs to the modern are:

- ■ The processes are rounded rectangles, which contain text more efficiently than circles.
- ■ Straight lines are used, rather than curved, because straight lines are easier to follow, and you can fit more in larger diagrams.

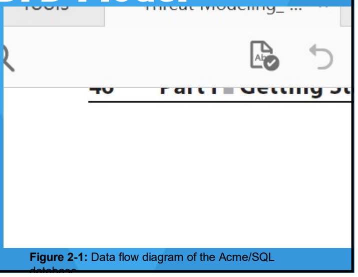


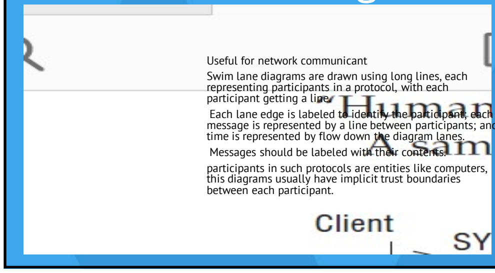
**Figure 2-1:** Data flow diagram of the Acme/SQL database

## Unified Modeling Language (UML)

If you use UML in your software development process, you can adapt UML diagrams for threat modeling, rather than redrawing them, By adding trust boundaries.

There are several types of UML diagrams, structure diagrams, behavior diagrams, and interaction diagrams

## Swim Lane Diagrams



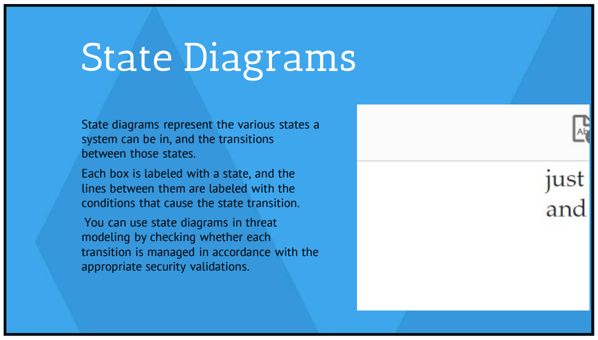
Useful for network communicant

Swim lane diagrams are drawn using long lines, each representing participants in a protocol, with each participant getting a line.

Each lane edge is labeled to identify the participant; each message is represented by a line between participants; and time is represented by flow down the diagram lanes.

Messages should be labeled with their contents.

participants in such protocols are entities like computers, this diagrams usually have implicit trust boundaries between each participant.



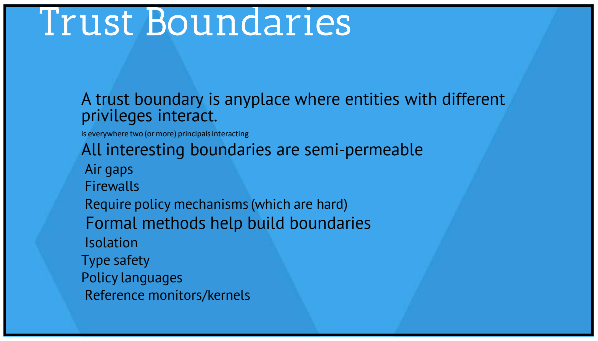
## State Diagrams

State diagrams represent the various states a system can be in, and the transitions between those states.

Each box is labeled with a state, and the lines between them are labeled with the conditions that cause the state transition.

You can use state diagrams in threat modeling by checking whether each transition is managed in accordance with the appropriate security validations.

just
and



## Trust Boundaries

A trust boundary is anyplace where entities with different privileges interact.

is everywhere two (or more) principals interacting

All interesting boundaries are semi-permeable

Air gaps

Firewalls

Require policy mechanisms (which are hard)

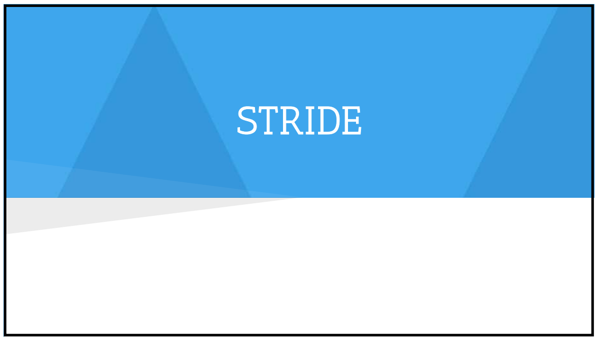
Formal methods help build boundaries

Isolation

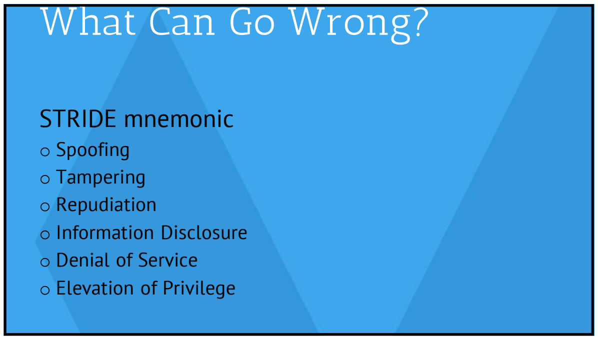
Type safety

Policy languages

Reference monitors/kernels



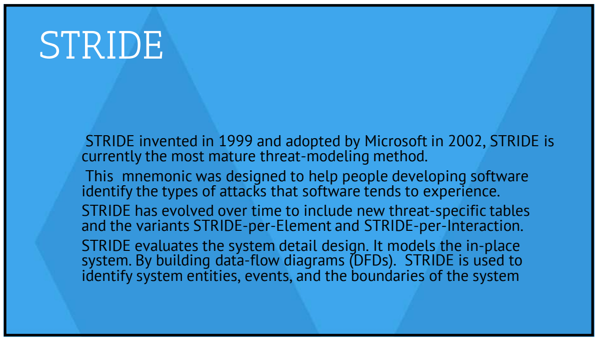
## STRIDE



# What Can Go Wrong?

STRIDE mnemonic

○ Spoofing
○ Tampering
○ Repudiation
○ Information Disclosure
○ Denial of Service
○ Elevation of Privilege



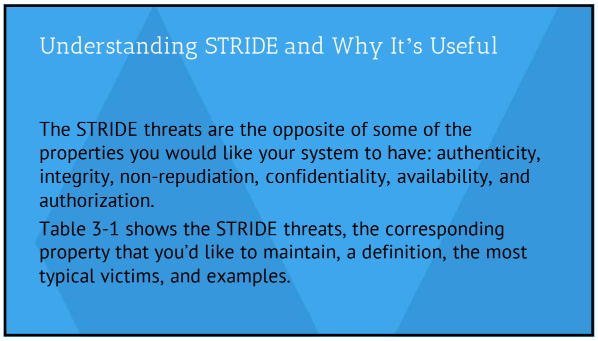
# STRIDE

STRIDE invented in 1999 and adopted by Microsoft in 2002, STRIDE is currently the most mature threat-modeling method.

This mnemonic was designed to help people developing software identify the types of attacks that software tends to experience.

STRIDE has evolved over time to include new threat-specific tables and the variants STRIDE-per-Element and STRIDE-per-Interaction.

STRIDE evaluates the system detail design. It models the in-place system. By building data-flow diagrams (DFDs). STRIDE is used to identify system entities, events, and the boundaries of the system

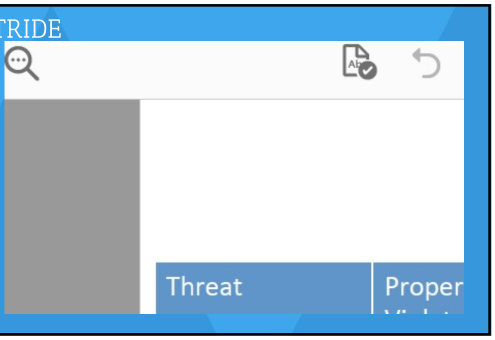


# Understanding STRIDE and Why It's Useful

The STRIDE threats are the opposite of some of the properties you would like your system to have: authenticity, integrity, non-repudiation, confidentiality, availability, and authorization.

Table 3-1 shows the STRIDE threats, the corresponding property that you'd like to maintain, a definition, the most typical victims, and examples.

## STRIDE



## When to Find Threats

Start at the beginning of your project

Create a model of what you're building

Do a first pass for threats

Dig deep as you work through features

Think about how threats apply to your mitigations

Check your design & model matches as you get close to shipping

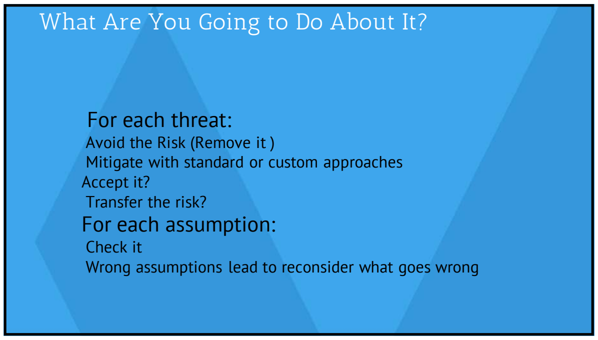
## Attackers Respond to Your Defenses

The ideal attacker will follow the road you defend

Ideal attackers are like spherical cows — they're a useful model for some things

Real attackers will go around your defenses

Your defenses need to be broad and deep



## What Are You Going to Do About It?

For each threat:

Avoid the Risk (Remove it )

Mitigate with standard or custom approaches

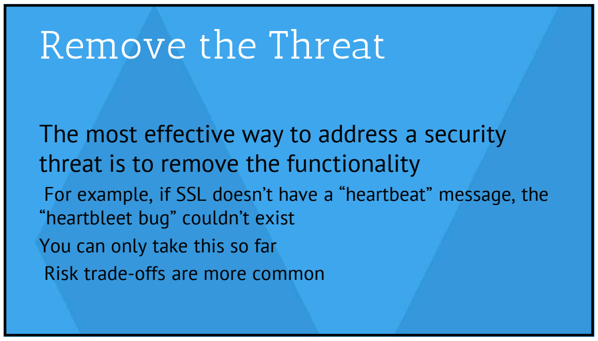
Accept it?

Transfer the risk?

For each assumption:

Check it

Wrong assumptions lead to reconsider what goes wrong



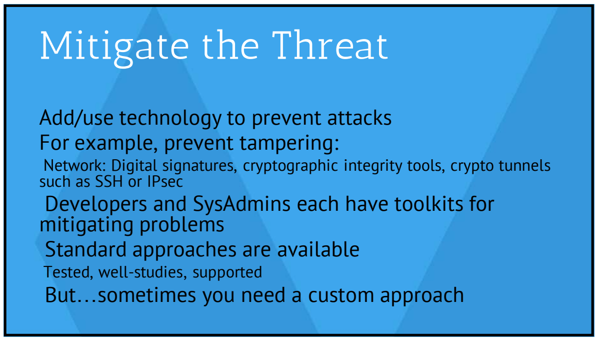
## Remove the Threat

The most effective way to address a security threat is to remove the functionality

For example, if SSL doesn't have a "heartbeat" message, the "heartbleet bug" couldn't exist

You can only take this so far

Risk trade-offs are more common



## Mitigate the Threat

Add/use technology to prevent attacks

For example, prevent tampering:

Network: Digital signatures, cryptographic integrity tools, crypto tunnels such as SSH or IPsec

Developers and SysAdmins each have toolkits for mitigating problems

Standard approaches are available

Tested, well-studies, supported

But…sometimes you need a custom approach

## Accept the Risk

Works best when it's your risk

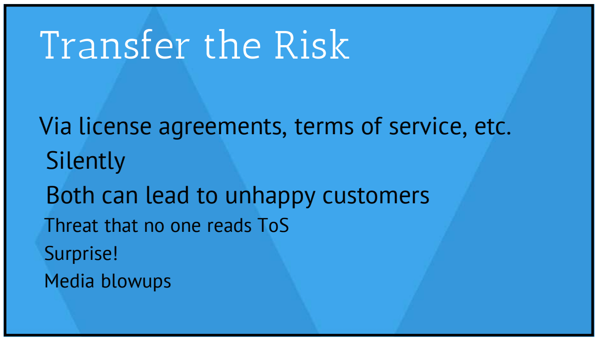
Your organization can accept risk

Be careful about "accepting" risk for your customers.

Customer risk acceptance

Via user interface

Sometimes the customer has details you can't have (is this network your work or a coffee shop?)



## Transfer the Risk

Via license agreements, terms of service, etc.

Silently

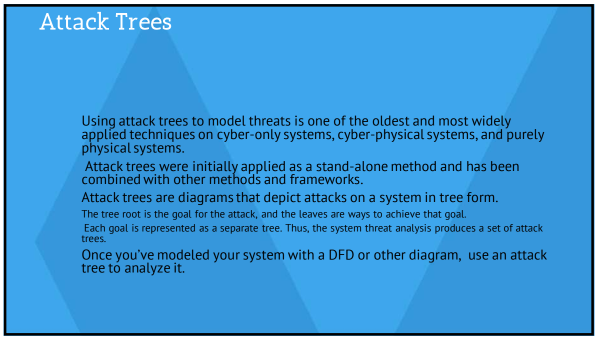
Both can lead to unhappy customers

Threat that no one reads ToS

Surprise!

Media blowups

## Attack Tree



## Attack Trees

Using attack trees to model threats is one of the oldest and most widely applied techniques on cyber-only systems, cyber-physical systems, and purely physical systems.

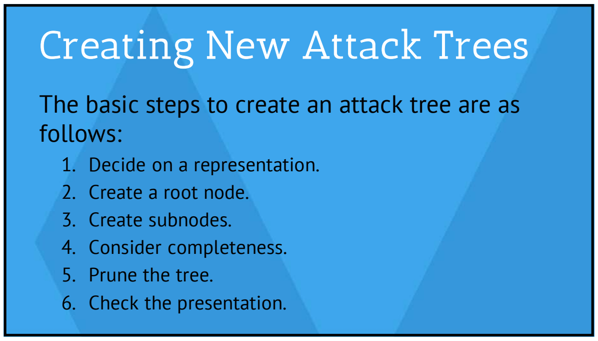
Attack trees were initially applied as a stand-alone method and has been combined with other methods and frameworks.

Attack trees are diagrams that depict attacks on a system in tree form.

The tree root is the goal for the attack, and the leaves are ways to achieve that goal.

Each goal is represented as a separate tree. Thus, the system threat analysis produces a set of attack trees.

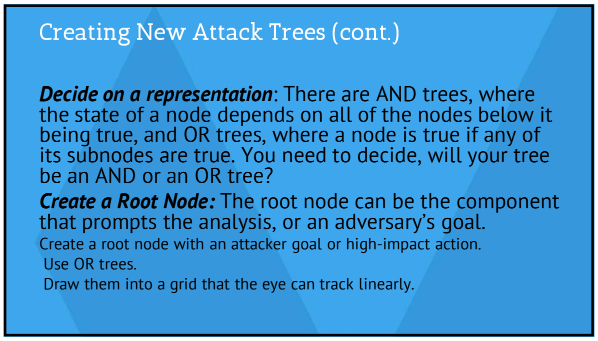
Once you've modeled your system with a DFD or other diagram, use an attack tree to analyze it.



## Creating New Attack Trees

The basic steps to create an attack tree are as follows:

1. Decide on a representation.
2. Create a root node.
3. Create subnodes.
4. Consider completeness.
5. Prune the tree.
6. Check the presentation.



## Creating New Attack Trees (cont.)

***Decide on a representation***: There are AND trees, where the state of a node depends on all of the nodes below it being true, and OR trees, where a node is true if any of its subnodes are true. You need to decide, will your tree be an AND or an OR tree?

***Create a Root Node:*** The root node can be the component that prompts the analysis, or an adversary's goal.

Create a root node with an attacker goal or high-impact action.

Use OR trees.

Draw them into a grid that the eye can track linearly.

# Creating New Attack Trees (cont.)

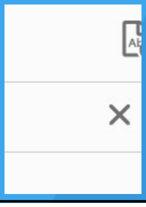
***Create Subnodes:***

You can create subnodes by brainstorming, or you can look for a structured way to find more nodes. The relation between your nodes can be AND or OR.

You can use these structures as a starting point, and make them more specific to your system.

Iterate on the trees, adding subnodes as appropriate.

Some possible structures for first-level subnodes include:



# Creating New Attack Trees (cont.)

***Consider Completeness:*** For this step, you want to determine whether your set of attack trees is complete enough.

An attack tree can be checked for quality by iterating over the nodes, looking for additional ways to reach the goal.

It may be helpful to use STRIDE to help you check the quality.

***Prune the Tree:*** In this step, go through each node in the tree and consider whether the action in each subnode is prevented or duplicative.

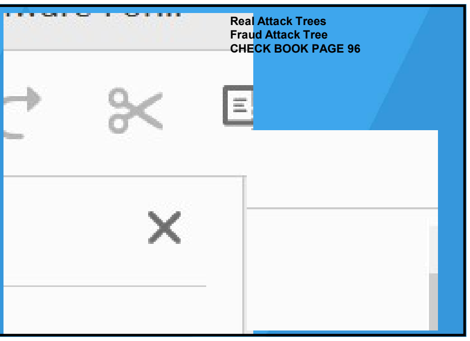
If an attack is prevented, by some mitigation you can mark those nodes to indicate that they don't need to be analyzed.

Marking the nodes (rather than deleting them) helps people see that the attacks were considered.

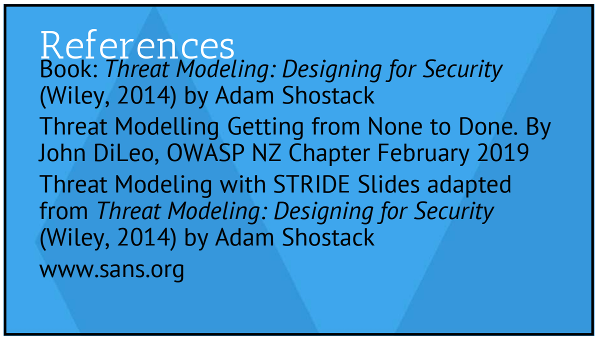
***Check the Presentation:***

you should aim to present each tree or subtree in no more than a page. If your tree is hard to see on a page, it may be helpful to break it into smaller trees.

The node labels should be of the same form, focusing on active terms. Finally, draw the tree on a grid to make it easy to track



**Real Attack Trees**
**Fraud Attack Tree**
**CHECK BOOK PAGE 96**



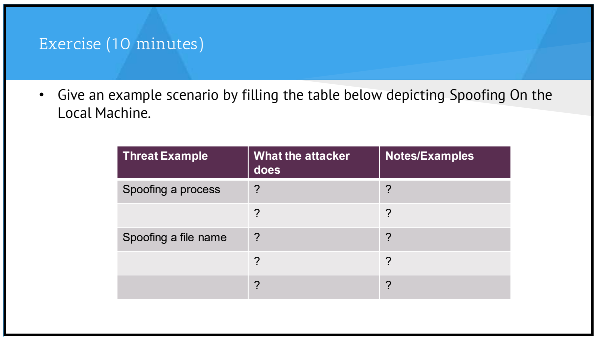
# References

Book: *Threat Modeling: Designing for Security* (Wiley, 2014) by Adam Shostack

Threat Modelling Getting from None to Done. By John DiLeo, OWASP NZ Chapter February 2019

Threat Modeling with STRIDE Slides adapted from *Threat Modeling: Designing for Security* (Wiley, 2014) by Adam Shostack

www.sans.org



## Exercise (10 minutes)

- Give an example scenario by filling the table below depicting Spoofing On the Local Machine.

| Threat Example | What the attacker does | Notes/Examples |
|---|---|---|
| Spoofing a process | ? | ? |
| | ? | ? |
| Spoofing a file name | ? | ? |
| | ? | ? |
| | ? | ? |

# Questions/Comments?

*Thank You*