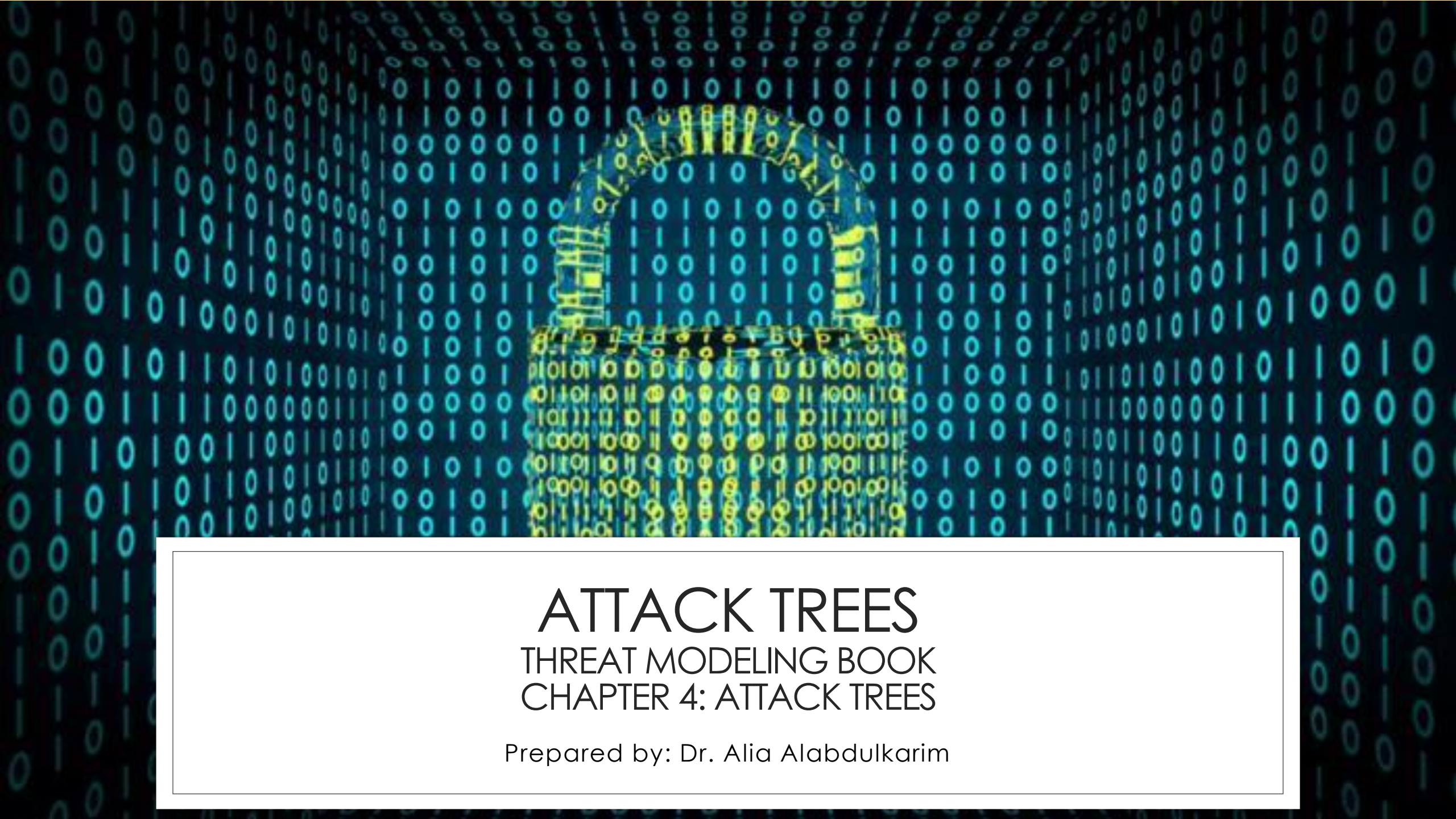


ATTACK TREES

THREAT MODELING BOOK
CHAPTER 4: ATTACK TREES

Prepared by: Dr. Alia Alabdulkarim

What is an Attack Tree

- As Bruce Schneier wrote in his introduction to the subject, "Attack trees provide a formal, methodical way of describing the security of systems, based on varying attacks. Basically, you represent attacks against a system in a tree structure, with the goal as the root node and different ways of achieving that goal as leaf nodes" (Schneier, 1999)
- Alternative to STRIDE
- Way to organize threats found

Working with Attack Trees

- There are three ways:
  - Using one someone else created
  - Create one for a project you are working on
  - Create trees intended for other's use

Using Attack Trees to Find Threats

- If you have an attack tree that is relevant to the system you're building, you can use it to find threats
- Once you've modeled your system with a DFD or other diagram, you use an attack tree to analyze it
- The attack elicitation task is to iterate over each node in the tree and consider if that issue (or a variant of that issue) impacts your system
- If there's no tree that applies to your system, you can either create one, or use a different threat enumeration building block

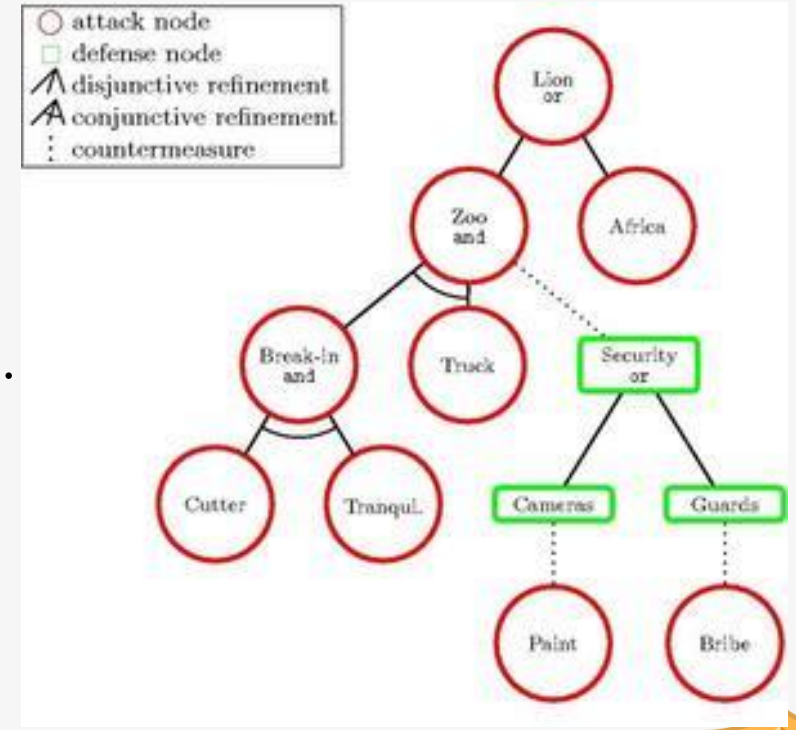
Creating New Attack Trees

- A project-specific tree is a way to organize your thinking about threats.
- **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.

1. Decide on a Representation

- **AND trees**, where the state of a node depends on all of the nodes below it being true
- **OR trees**, where a node is true if any of its subnodes are true.
- Can be presented **graphically** or as an **outline**



Source: https://wwwfr.uni.lu

2. Create a Root Node

- The root node can be the **component** that prompts the analysis, or an **attacker's goal:**
  - If the root node is a **component**, the subnodes should be labeled with **what can go wrong** for the node
  - If the root node is an **attacker goal**, subnodes will represent **ways to achieve that goal**
- It is recommended to:
  - 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

3. Create Subnodes

- Brainstorming
- Look for a structured way to find more nodes
- Some possible structures for first-level subnodes include:

- Attacking a system:
  - physical access
  - subvert software
  - subvert a person

- Attacking a system via:
  - People
  - Process
  - Technology

- Attacking a product during:
  - Design
  - Production
  - Distribution
  - Usage
  - Discard

4. Consider Completeness

- You want to determine whether your set of attack trees is complete enough
  - Consider additional components
  - Look at each node and ask "is there another way that could happen?"
  - Consider additional attackers or motivations

- 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, or
  - One of the attack libraries, or
  - A literature review to help you check the quality

5. Prune the Tree

- 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.

6. 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

Representing a Tree

- Can be represented in two ways:
  - A free-form (**human-viewable**) model: without any technical structure
  - As a **structured** representation with variable types and/or metadata to facilitate programmatic analysis.

Human-Viewable Representations

- **Graphical** vs. **Outline**
- **Graphical representations** are a bit more work to create but have more potential to focus attention
- In either case, if your nodes are not all related by the same logic (AND/OR), you'll need to decide on a way to represent the relationship and communicate that decision
- For graphical representation:
  - Use distinct shapes for terminal nodes?
  - Labels should be rich in information
  - Choosing words: "attack", "via" vs. "modify file"
- Graphics must be information-rich and communicative

Human-Viewable Representations

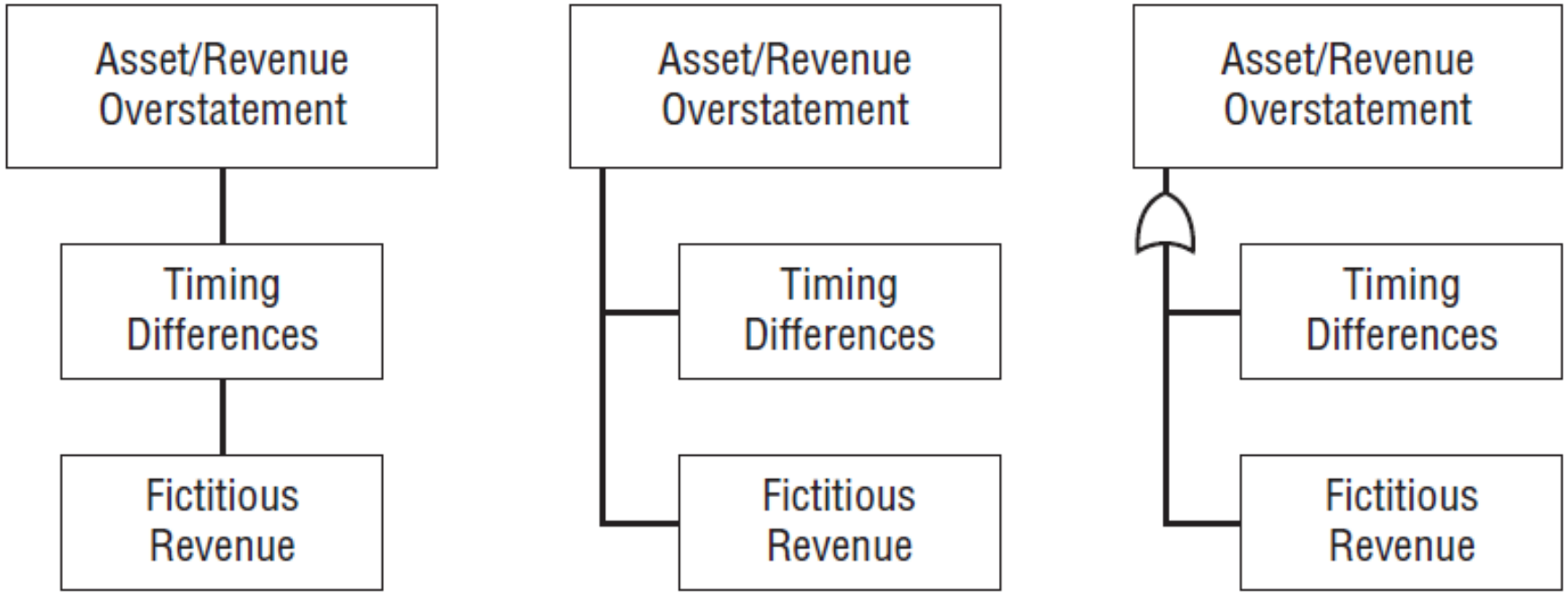
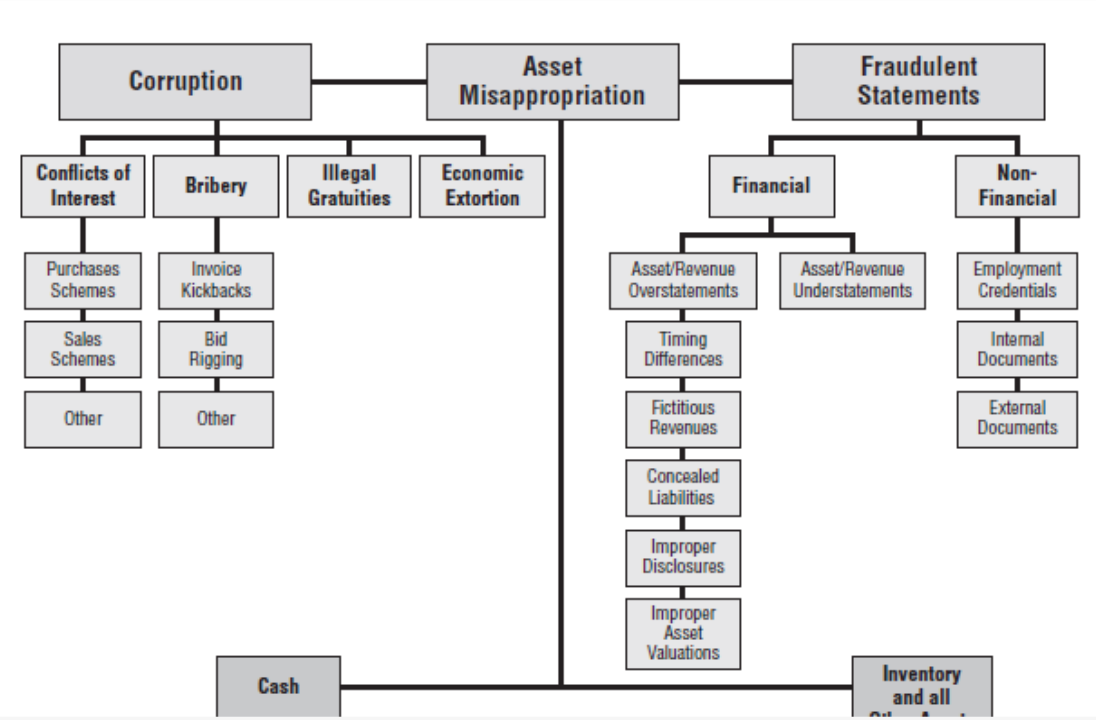
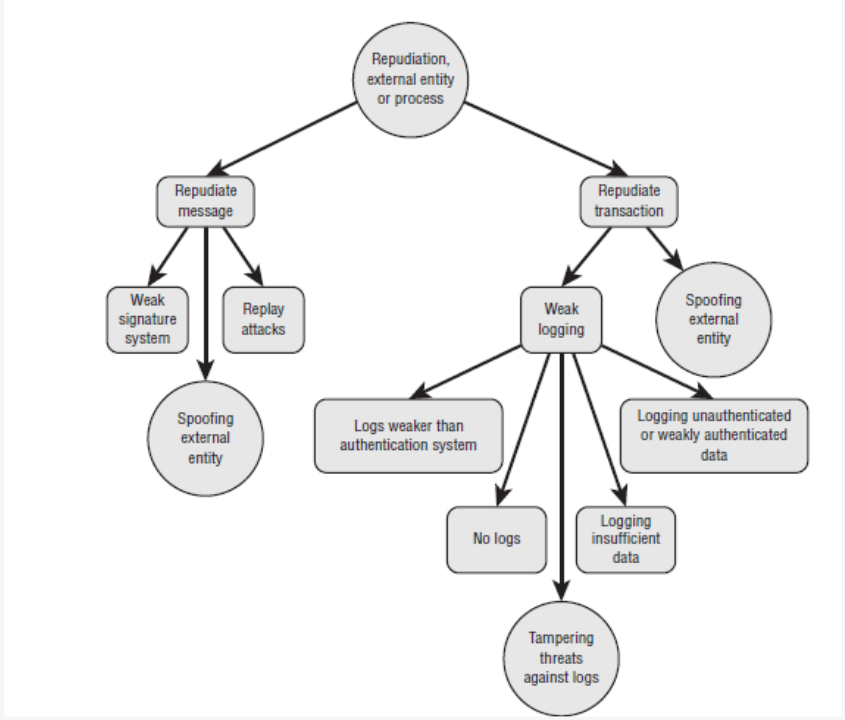


Figure 4–1: Three representations of a tree

Human-Viewable Representations





Human-Viewable Representations

- **Outline representations** are easier to create, but they tend to be less attention-grabbing
- Representing AND/OR is not simple:
  - Some representations leave them out
  - Others include an indicator

  either before or after a line

1. Attack voting equipment
   - 1.1 Gather knowledge
     - 1.1.1 From insider
     - 1.1.2 From components
   - 1.2 Gain insider access
     - 1.2.1 At voting system vendor
     - 1.2.2 By illegal insider entry

Human-Viewable Representations

1. Attack voting equipment
   - 1.1 Gather knowledge (and)
     - 1.1.1 From insider (or)
     - 1.1.2 From components
   - 1.2 Gain insider access (and)
     - 1.2.1 At voting system vendor (or)
     - 1.2.2 By illegal insider entry

Human-Viewable Representations

- Outline representation drawbacks:
  - Less simple than you might expect
  - When using someone else's tree, be sure you understand their intent
  - When you are creating a tree, be sure you are clear on your intent, and clear in your communication of your intent

Structured Representations

- A tree is also a data structure
- A structured representation of a tree makes it possible to apply logic to the tree and in turn, the system you're modeling
- Several software packages enable you to create and manage complex trees
- Allows the modeler to add costs to each node, and then assess what attacks an attacker with a given budget can execute
- More in Chapter 11: Threat Modeling Tools

Example Attack Tree

Open PDF file: **ExampleAttackTree.pdf**

Real Attack Trees

- A variety of real attack trees have been published
  - Use directly if they model systems like the one you're modeling, OR
  - As an example of how to build an attack tree

- Three real trees examples:
  - Fraud attack tree [grid]: **FraudAttackTree.pdf**
  - Election operations assessment tree [AND/OR nodes]: **ElecOpTree.pdf**
  - Attacks against SSL [mind map]: **MindMap.pdf**

Perspective on Attack Trees

- They are a useful way to convey information about threats
- Quickly consider possible attack types
- However, despite their surface appeal, it is very hard to create attack trees
- There are also a set of issues that can make trees hard to use:
  - **Completeness:** Without the right set of root nodes, you could miss entire attack groupings
  - **Scoping:** The nature of attack trees means many of the issues discovered will fall under the category of "there's no way for us to fix that."
  - **Meaning:** There is no consistency around AND/OR, or around sequence, which means that understanding a new tree takes longer.

References

- Threat Modeling
  - Chapter 4: Attack Trees