Bachelor Computer Science

From Attack Trees to Attack Graphs: Implementing Conversion and User Evaluation

Renée Gerritse

Supervisors: Nathan Daniel Schiele & Olga Gadyatskaya

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

Bachelor Project

Renée Gerritse

January 2026

Abstract

This study examines how two visual representations, attack–defense trees and attack graphs, affects users' understanding of attack scenarios. We extended the ATD-WebApp with support for SAND logic and a tree-to-graph conversion, so that the same attack-defense tree model can be visualized both as a tree and as an attack graph. Using a counterbalanced online questionnaire with 36 participants, we compare the two representations on four aspects: goal identification, path enumeration, interpretation of countermeasures, and overall preference.
In our study, attack-defense trees were more often perceived as clearer for identifying the attacker's main goal and for locating where countermeasures act. In a more complex scenario involving combined OR, AND, and SAND relations, attack graphs help participants enumerate a more complete set of valid attack paths than attack-defense trees, even though many participants still preferred trees overall. These results suggest that attack-defense trees are well suited for communicating the "big picture" of an attack and the placement of defenses, whereas attack graphs are better suited for detailed reasoning about how an attack can unfold along different paths. Tools that allow analysts to switch between both views can therefore exploit the complementary strengths of trees and graphs.

Contents

1 Introduction

Systems are increasingly exposed to complex cyberattacks, as computers and networked devices have become a major source of information [SK20]. To reason about such attacks and to plan defenses, security analysts often rely on visual models that capture possible attack steps together with countermeasures. Two important visual representations are attack trees and attack graphs. Attack trees represent how an attacker can achieve a goal using a hierarchical structure of subgoals and basic actions [MO05], whereas attack graphs represent system states and transitions between them [SHJ+02]. Attack–defense trees extend attack trees by integrating countermeasures directly into the model, making them useful for representing both attacks and defenses in a single diagram.
These models are used in both research and practice [BSCS19, CGH+19] but differ substantially in their structure and visualization [LDB18]. Prior work has shown that each model has strengths in different contexts, yet it is unclear how their visual representation affects users' clarity and understanding of attack scenarios, especially when both models are derived from the same underlying model.

In this thesis, we took a two-step approach. First, we extend the existing ADT-WebApp with support for SAND logic and a tree-to-graph conversion, enabling the same attack–defense tree model to be visualized both as a tree and as an attack graph. This ensures that any differences in user responses can be attributed to the visual representation rather than to differences in the underlying model. Second, we conducted a questionnaire study with 36 participants, in which each scenario was shown in both representations (Tree→Graph or Graph→Tree order). The questionnaire compares how well participants can identify the main goal, enumerate valid attack paths, interpret countermeasures, and which representation they prefer overall.
Together, these two steps allow us to answer the following research question:

> *How do visual representations (attack–defense trees vs. attack graphs) affect users' clarity and understanding of attack scenarios?*

The remainder of this thesis is organized as follows. First, we introduce the necessary background on graphs, attack graphs, attack trees, and attack–defense trees, followed by a discussion of related work. We then describe the implementation of SAND support and the tree-to-graph conversion in the ADT-WebApp. After that, we present the design and procedure of the questionnaire study and report its results. Finally, we discuss the limitations of the study and discuss what could be explored in future research.

2 Background

Graph-based representations are widely used to model system vulnerabilities and attack scenarios [SHJ+02, KKMS10]. Attack trees (AT) and attack graphs (AG) are two representations in threat modeling [SDP08, LMG17]. Although both approaches structure and visualize potential attacks, they differ in how they represent the relationships between steps, system states, and countermeasures.
In this section, we introduce the core concepts that form the basis of our study and the tree-to-graph conversion: graphs, attack graphs, trees, attack trees, and attack-defense trees.

2.1 Graphs

A **graph** $G = (V, E)$ consists of a set of vertices V and a set of edges E, where each edge connects two vertices [GYA18]. Graphs are widely used to represent relationships between entities and serve as the foundation for models in computer science, network analysis, and cybersecurity [SHJ+02].

In an **undirected graph**, edges have no direction, so traversal between connected vertices is possible in both directions. In contrast, a **directed graph** uses ordered pairs (u, v) to indicate a one-way connection from vertex u to vertex v; traversal is only possible from u to v unless a separate edge (v, u) also exists.

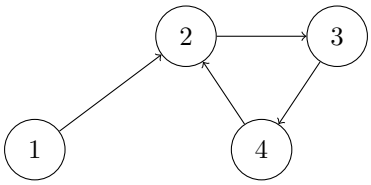


Figure 1: A graph illustrating a cycle.

A **path** is a sequence of vertices $v_0, v_1, \ldots, v_k$ such that each successive pair is connected by an edge of the graph; that is, $(v_{i-1}, v_i) \in E$ for all $i = 1, \ldots, k$. In a **directed** graph, each edge must point from v_{i-1} to v_i. A *simple* path does not have repeated vertices.
A **cycle** is a simple path of length ≥ 3 that starts and ends at the same vertex, with all edges and intermediate vertices distinct. A graph is **acyclic** if it contains no cycles.
For example, in the graph in Figure 1, the sequence $2 \to 3 \to 4 \to 2$ forms a **cycle**: it starts and ends at the same vertex, with all intermediate vertices distinct.

Directed acyclic graphs (DAGs) are particularly relevant for this research, as they also represent the structure of attack trees.

Figure 2 shows an example of a directed graph $G = (V, E)$, where

$$V = \{1, 2, 3, 4\} \quad \text{and} \quad E = \{(1, 2), (2, 3), (2, 4), (3, 4)\}.$$

This graph has four vertices and four directed edges. Since it is a directed graph, traversal is only allowed in the direction indicated by each edge. It is connected since there is a path between each pair of vertices and it is acyclic because it contains no cycles.

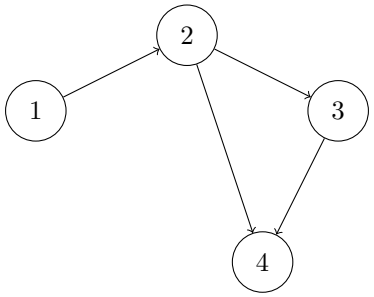


Figure 2: An example of a directed graph.

2.2 Attack graphs

Attack graphs were introduced by Sheyner et al. [SHJ+02] as a way to model and analyze all possible attack paths through a system. They represent system states as nodes and attacker actions as directed edges, allowing analysts to trace how an attacker might progress towards a goal.
Unlike attack trees, which have a hierarchical structure, attack graphs can contain cycles. A cycle can arise when an attacker revisits a previous state or retries an action under different conditions. This makes them suitable for modeling complex scenarios with multiple routes, dependencies between steps, and loops in the attack process.

An **attack graph** is a directed graph in which:

- **States (nodes)** represent system states that can be reached during an attack.
- **Edges** represent transitions between these states, triggered by attacker actions.

Each path in the attack graph represents a potential sequence of actions an attacker might follow, progressing from the initial state through intermediate states towards the target goal.

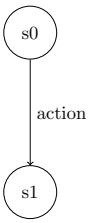


Figure 3: Minimal attack graph example showing two states and one transition.

To illustrate a basic transition, Figure 3 shows a minimal example with two states (`s0` and `s1`) connected by a single action performed by an attacker, adapted from the work of Schiele and Gadyatskaya [SG21].

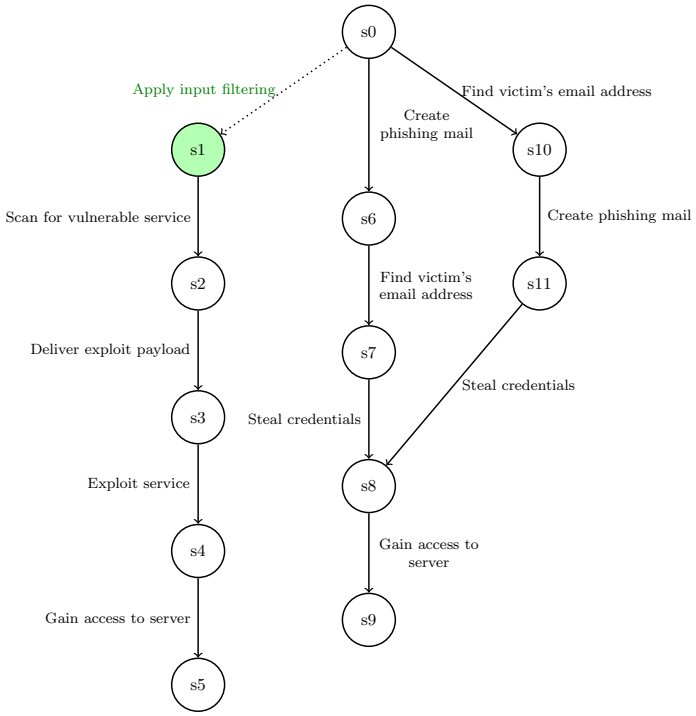


Figure 4: Attack graph illustrating attack paths to gain access to a server

In the remainder of this section, we use the same example across all representations. The attacker's goal is to *gain access to a server*.
Figure 4 shows this scenario as an attack graph. Starting from an initial state $s0$, the graph illustrates all possible paths that can lead to the final state, representing the attacker's goal. Multiple paths exist, such as exploiting a vulnerable service or stealing credentials. The green node represents a defensive action (*Ap-*

ply input filtering) that blocks one of the possible paths, forcing the attacker to pursue alternative routes.

2.3 Trees

In graph theory, a tree is a connected, acyclic, undirected graph $T = (V, E)$, where for every pair of distinct vertices $u, v \in V$, there exists exactly one path connecting u and v [Gou12].
Equivalent, a tree with n vertices always has exactly $n - 1$ edges. This is due to the fact that a tree is a minimally connected graph, it has enough edges to ensure connectivity among all vertices without creating any cycles. Adding an edge would introduce a cycle, while removing an edge would break the graph's connectivity.

Trees are commonly used to represent hierarchical relationships and are widely used in file systems, search algorithms, and network routing [GMSW06].

A **rooted tree** is a tree in which one distinguished node is designated as the root. The root serves as the starting point of the tree, inducing a hierarchical structure that flows from the root to its children. If u and v are joined by an edge and u is closer to the root than v, then u is the *parent* of v and v is its *child*. From the root node, every node is reachable via a unique path. Rooted trees are fundamental for representing structured models, such as attack trees.

- Every node except the root node has exactly one parent.
- A node with no children is a *leaf*.
- The *depth* of a node is defined as the number of edges from the root to that node.
- The *height* of the tree is the maximum depth among all the nodes (the longest path).

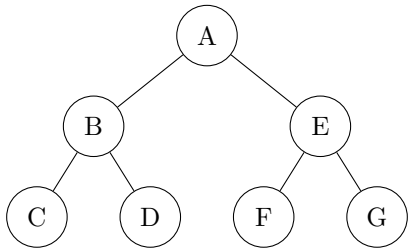


Figure 5: A simple rooted tree with a height 2.

Figure 5 illustrates a simple rooted tree with node A as the root. The root has two children, nodes B and E, which serve as roots of their subtree. The

leafs of the tree are the nodes C, D, F, and G. The height of the tree is 2, corresponding to the length of the longest path from the root to a leaf.
Rooted trees form the structural basis for attack trees, which model how specific goals can be achieved through a hierarchy of attack steps. This structure will be further discussed in the next section.

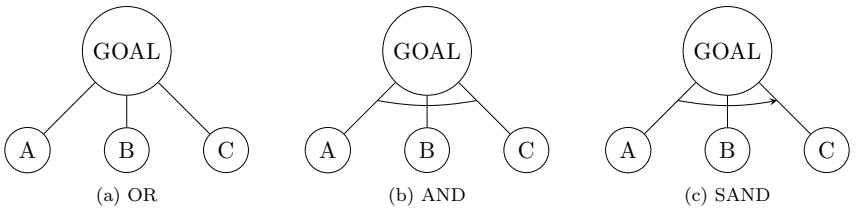
2.4 Attack trees

Attack trees were introduced by Bruce Schneier in 1999 as a structured method for graphically representing potential attack scenarios[Sch99]. They help analysts identify, visualize, and analyze strategies that an attacker might use to compromise a system.
Using a hierarchical tree structure, attack trees break down the main attack goal into subgoals and individual steps [MO05]. This allows both critical and less critical attack paths to be represented. The structure enables analysts to prioritize vulnerabilities, assess threats, and develop effective defense strategies. By offering a top-down perspective of how an attack can unfold, attack trees provide valuable insight into system vulnerabilities [BFM04].

An **attack tree** is a rooted tree model used in cybersecurity to represent different ways in which an attacker can achieve a malicious goal. It is interpreted bottom–up: an attack instance starts at a leaf node and moves up the tree toward the goal (the root).
Internal nodes in an attack tree represent subgoals or intermediate steps and define logical relationships (see Figure 6) between their child nodes using logical operators, such as `OR`, `AND` and `SAND`, where:

- An `OR` node requires at least one child node to be satisfied to proceed to the parent node.
- An `AND` node requires that all child nodes be satisfied to proceed to the parent node, but the order does not matter.
- A `SAND` (sequential AND) node requires all child nodes to be satisfied in a specific order to proceed to the parent node [JKM+15].

Each node in the tree can have child nodes, which represent steps or conditions that must be fulfilled before the parent node can be executed. This shows the flow of the attack.



(a) OR (b) AND (c) SAND

Figure 6: Logical relationships in attack trees: (a) OR, (b) AND, (c) SAND

Figure 7 revisit the same scenario as in the attack graph where the goal is to gain access to a server. The root node, *Gain access to server*, is connected via an **OR** relationship to two subgoals: *Exploit service* and *Steal credentials*. Either subgoal can lead to gaining access to a server. The subgoals and their corresponding logical structures are explained as follows:

- *Exploit service* follows a **SAND** relationship, where one must first *Scan for vulnerable service* and then *Deliver exploit payload*.
- *Steal credentials* follows a **AND** relationship, where both *Create phishing mail* and *Find victim's email address* must be completed, the order does not matter.

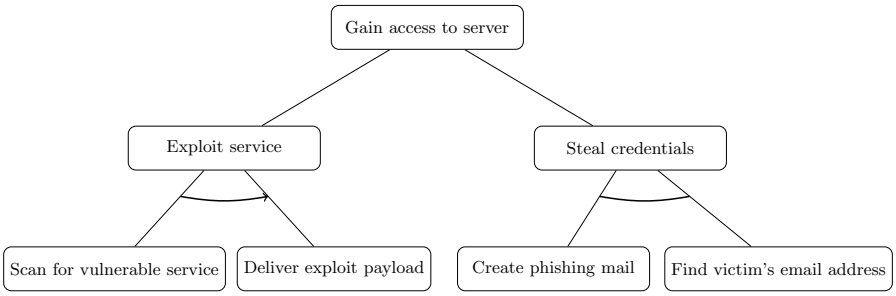


Figure 7: Attack Tree illustrating attack paths to gain access to a server

A formal definition of attack trees can be derived from the work of Fila et al. [FW20], who modeled attack-defense trees based on directed acyclic graphs (DAGs) using a tuple-based representation.

The original definition of an **attack-defense tree** of Fila et al. is:

$$T = (V, E, L, \lambda, actor, \tau)$$

where:

- (V, E) is a rooted directed acyclic graph representing the structure of the tree,
- L is a set of labels representing actions or goals,
- $\lambda : V \to L$ is a injective function that assigns labels to nodes,
- $actor : V \to \{a, d\}$ assigns each node to either the attacker (a) or defender (d),
- $\tau : V \to \{\text{AND}, \text{OR}, \text{SAND}, \text{N}\}$ defines how each node is refined: `AND`, `OR`, sequential-`AND` (`SAND`), or not at all(`N`).

To derive a formal model of an **attack tree**, we adapted the structure by removing the `actor` component, as it represents defense nodes and only attackers actions are relevant. This results in the simplified tuple:

An **attack tree** is a tuple

$$T = (V, E, L, \lambda, \tau)$$

In this formulation, all nodes represent actions performed by the attacker. A node $v \in V$ is considered a *basic action* if and only if it has no children in the graph.

2.5 Attack-defense tree

Attack-defense trees capture the interaction between potential attacks on a system and the corresponding countermeasures that can be employed to counter them. They were introduced and formally defined by Kordy et al. in 2010 as an extension of the original attack tree model by Schneier[KKMS10]. While Schneier's attack trees focus solely on modeling the attacker's perspective, attack-defense trees enhance this model by introducing defense nodes, which represent countermeasures that defenders can employ against specific attack steps. As a result, attack-defense trees offer a more comprehensive and balanced view of system security, covering offensive and defensive strategies.

An **attack-defense tree** is a DAG-based model used in cybersecurity to represent different strategies that an attacker may use to achieve a malicious goal, along with possible defense responses.

Figure 8 revisits the previous attack scenario from the attack tree (see Figure 7) with the addition of a defense component. The green node labeled *Apply input filtering* acts as a countermeasure to the attack step *Deliver exploit payload*. This defensive action blocks malicious input before it can exploit a vulnerable service. Since the path *Exploit service*, the left child of *Gain access to server*, is modeled using a `SAND` (sequential AND) relationship, both steps *Scan for vulnerable service* followed by *Deliver exploit payload* must succeed for the attack to proceed. If the second step is neutralized, the entire attack path becomes ineffective, forcing the attacker to explore alternative routes.

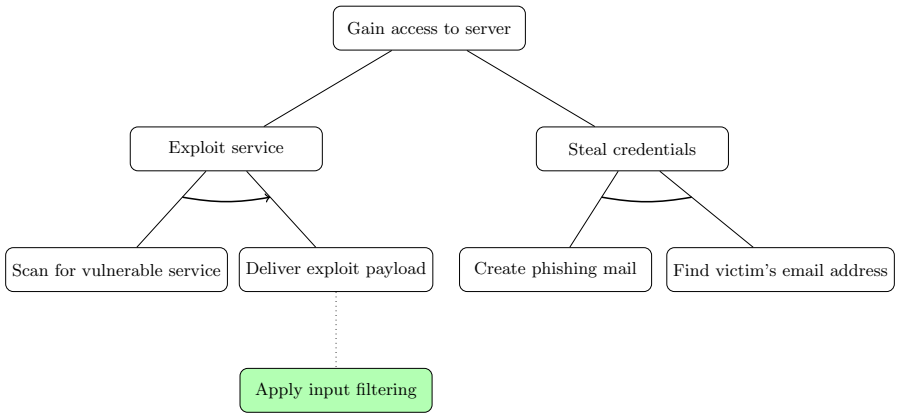


Figure 8: Attack-defense tree illustrating attack paths to gain access to a server

We previously introduced a formal definition for attack trees by adapting the model of Fila et al. [FW20] and removing the `actor` component. We now take the original definition of the **attack-defense tree** from their work, which includes the `actor` component to distinguish between attacker and defender nodes. The full definition of an **attack-defense tree** is:

$$T = (V, E, L, \lambda, actor, \tau)$$

where:

- (V, E) is a rooted directed acyclic graph representing the structure of the tree,
- L is a set of labels representing actions or goals,
- $\lambda : V \to L$ is a injective function that assigns labels to nodes,
- $actor : V \to \{a, d\}$ assigns each node to either the attacker (a) or defender (d),
- $\tau : V \to \{\text{AND}, \text{OR}, \text{SAND}, \text{N}\}$ defines how each node is refined: `AND`, `OR`, sequential-`AND` (`SAND`), or not at all(`N`).

The inclusion of the *actor* function distinguishes attack-defense trees from attack trees by enabling defensive modeling. The defense nodes represent countermeasures that aim to prevent or neutralize specific attack steps.

3 Related Work

Attack trees and attack graphs are widely used in cybersecurity to model, demonstrate, and analyze potential attack methods. Attack trees represent attacks in a hierarchical form, where the root node represent the attacker's main goal and the leafs represent the initial actions. They have been applied across various domains such as system and network security [SDP08, RGG24]. Saini et al. [SDP08] introduced attack trees in an educational context by applying them to the MyProxy component of the Globus toolkit. Their work demonstrated how an attack tree can be constructed step by step to identify vulnerabilities and assess risks, emphasizing its practical value for risk identification. Following this approach, Rana et al. [RGG24] proposed a FAIR-modified attack tree framework to support quantitative risk assessment in organizational networks by combining the attack tree structure with the FAIR methodology. Together, these studies show that attack trees are flexible tools that can be extended for a more detailed analyses, but their overall clarity still depends on how familiar users are with the model. In this study, we address this issue by examining how clearly users can understand attack models, both with and without prior knowledge.

While attack trees describe how an attack can be decomposed into subgoals, attack graphs go a step further by modeling system states and transitions, providing a more detailed view of how an attacker can move through a network or system. They capture complex relationships such as loops, multiple paths, and dependencies between actions. Attack graphs have been applied in a range of domains to identify vulnerabilities and guide defensive strategies, evolving from static network analysis toward more specialized and dynamic systems. Ingols et al. [ICL+09] applied attack graphs to enterprise networks using the NetSPA system, showing how they can reveal critical weaknesses and support defense prioritization. Building on this, Lucket et al. [LMG17] used attack graphs to model vulnerabilities in ambulatory medical devices using Bluetooth, demonstrating how attack graph can be adapted to specialized and safety-critical environments by highlighting where sensitive information could be compromised. Extending this, to an even more dynamic context, Salayma et al. [Sal24] introduced attack graphs for Internet of Things networks, where systems continuously change as devices are added or removed. They showed how attack graphs can update in real time as devices connect and disconnect, making the models both dynamic and more complex. Together, these studies demonstrate the evolution of attack graphs from static infrastructure to adaptive systems, but they also highlight a challenge that as models grow in complexity, they become more difficult for users to interpret. This evolution is directly relevant to our study, as we examine how increasing model complexity affects clarity. We address this issue by analyzing how users understand attack graphs in comparison to attack trees, both conceptually and structurally.

While the previous works applied each model independently, other research has

directly compared attack trees and attack graphs to examine how each representation supports the understanding of cyberattacks. Alhebaishi et al. [AWJS16] applied both models to threats in cloud data center infrastructures. By constructing models of two representative cloud architectures, they showed that attack trees provide a higher-level overview of threats, while attack graphs reveal detailed attack paths within the network. This combination helped cloud providers understand security risks from different perspectives and guided improvements to their defensive strategies. Similarly, Lallie et al. [LDB17] conducted an empirical evaluation comparing attack graphs and fault trees to assess which method better supports the understanding of cyberattack scenarios by participants. Their results indicated that attack graphs were generally more effective in aiding perception and comprehension. In a follow-up study, Lallie et al. [LDB18] examined how participants interpret the visual syntax of attack trees and attack graphs, finding that analysts and security professionals preferred attack graphs due to their top-down flow, which was considered more intuitive than the bottom-up structure of attack trees. Building on these findings, the same authors [LDB20] conducted a broader survey analyzing over 180 attack trees and attack graphs, highlighting the lack of a standardized visual syntax for either models and the inconsistencies across representations. Together, these studies reflect a continued effort to understand how different visualization influence users' clarity about attack scenarios. This line of work directly relates to our study, where we present attack trees and attack graphs side by side to evaluate how visual representation affects user's clarity in identifying goals, paths, and countermeasure, while maintaining a consistent visual style across both models.

In addition to comparing the two representations, other research has focused on transforming one model into the other to highlight their complementary strengths. Pinchinat et al. [PAV14] proposed a synthesis approach in which attack scenarios are first described as an attack graph and then transformed into an attack tree. Their goal was to raise the level of abstraction and make risk analysis easier to perform, since attack trees provide a more structured and readable overview for experts. Haque and Atkinson[HKA17] took a different approach by presenting an evolutionary method for converting attack graphs into attack trees. While Pinchinat et al. focused on a model-based approach, Hague and Atkinson relied on data-driven techniques, analyzing large volumes of data from intrusion detection systems and network configurations to automatically generate attack trees that reorganize complex attack paths into a clearer and more manageable structure. Together, these studies highlight different approaches for converting attack graphs into trees to improve readability. In our work, we examine both directions, tree$\rightarrow$ graph and graph $\rightarrow$ tree, to asses how each transformation effect users' understanding of attack scenarios.

Conversely, some research has focused on converting attack trees into attack graphs. Schiele and Gadyatskaya [SG21] introduced a method that maps the hierarchical structure of attack trees into the state-based representation of at-

tack graphs. Their motivation is that attack trees are easier for humans to construct and reason about, while attack graphs enable deeper analysis of dependencies, cycles, and possible attack paths. We extend their approach in two ways: (1) by supporting countermeasures within attack-defense trees, and (2) by conducting a user study. This shifts the focus from transformation to an evaluation of its effect on human comprehension.

Finally, recent work has used games and gamification to raise awareness of cybersecurity risks. Ng and Hasan [NH25] analyzed 53 cybersecurity games and found that they can effectively improve user's cybersecurity awareness. Scholefield and Shepherd [SS19] took a more focused approach, showing that interactive learning through gamification can enhance understanding of security threats. Both studies demonstrate that engaging, visual, and interactive methods can improve comprehension of complex cybersecurity concepts. Our work takes a different direction: instead of using games, we compare two visual models to examine how their representation influences clarity and comprehension.

4 Implementation

Our first contribution was the implementation of the conversion of attack-defense trees to attack graphs. We used the ADT-WebApp as a foundation [Moh23] and extended it with a transformation algorithm. The extended application was later used to generate both the visualization and the questionnaire material (see Section 5).

4.1 ADT-WebApp

The ADT-WebApp[1] is a browser-based tool for constructing, visualizing, and editing attack-defense trees. Users start by defining a root goal and then incrementally add attacker actions and optional countermeasures, connected via `OR` and `AND` relations. Models can be saved and loaded in XML format for reuse and automated processing. We chose the application as a foundation because it already provides a graphical interface for modeling attack scenarios, making it suitable for extension. As shown in Figure 9, the interface of the ADT-WebApp allows users to define a root goal and iteratively add attack and defense actions to construct an attack-defense tree.

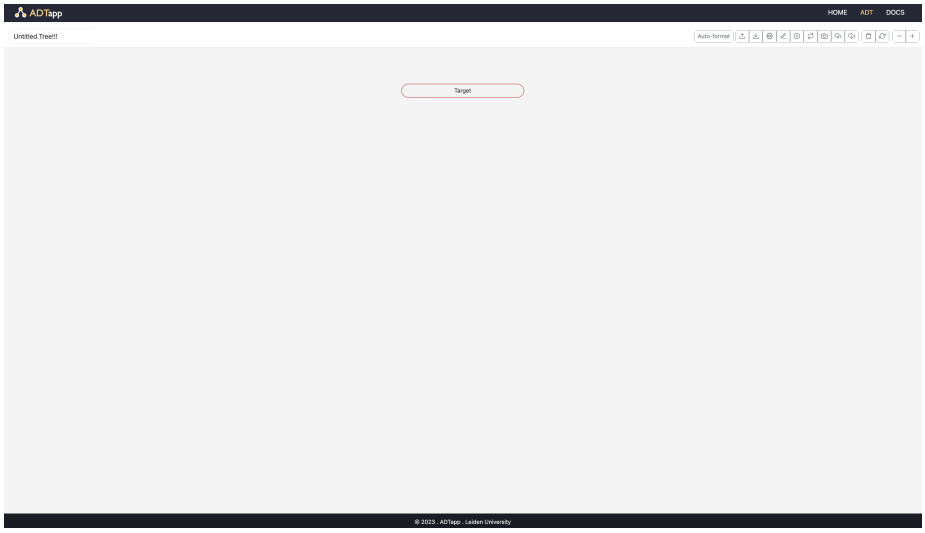


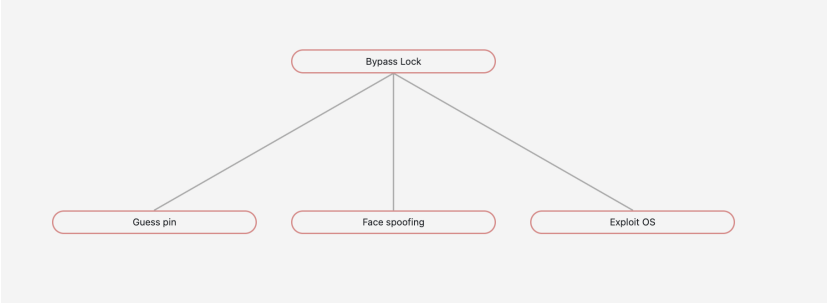
Figure 9: The ADT-WebApp interface used for modeling attack-defense trees.

4.2 Extension: Tree-to-Graph Conversion

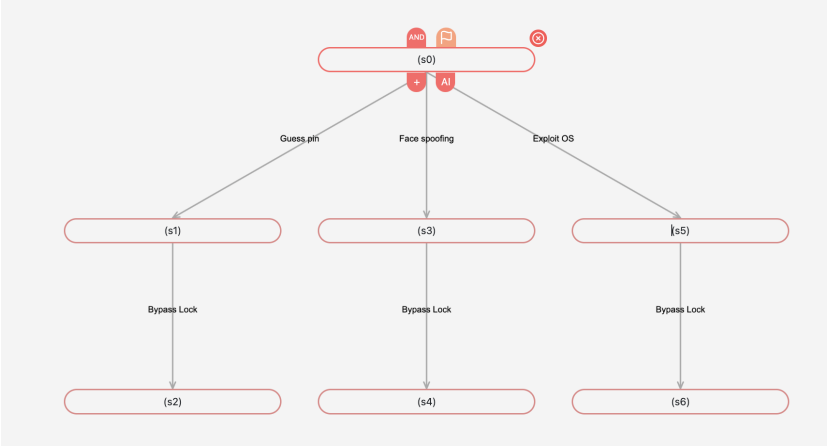
We extended the ADT-WebApp with the functionality to transform attack-defense trees into attack graphs. In the resulting graphs, system states are represented as nodes and attacker actions appear as labeled transitions. Countermeasure nodes act as blockers, preventing further progress along certain paths.

[1] `https://adtweb.app/home.html`

Figure 10 shows an example of an attack-defense tree (left) and its corresponding converted attack graph (right), both generated within the extended ADT-WebApp.



(a) Attack tree interface showing the root node (target) and child nodes.



(b) Attack graph visualization generated from the tree model.

Figure 10: Interface views of the ADT-WebApp: (a) attack tree, (b) corresponding attack graph.

4.2.1 Path construction

The central part of the conversion process is enumerating all valid root-to-leaf paths in the tree. The function `get_paths()` traverses the tree recursively and processes the logical operators.

Algorithm 1 get_paths(node)

```
if node.switchRole = "yes" then return empty list
end if
label ← node.label or "(no label)"
children ← child nodes with switchRole ≠ "yes"
if children is empty then return [[label]]
end if
refinement ← node.refinement or "disjunctive"
child_paths ← GET_PATHS(child) for each child
if refinement = "disjunctive" then                                  // OR
    result ← [ ]
    for paths in child_paths do
        for path in paths do
            Append [label] + path to result
        end for
    end forreturn result
else if refinement = "conjunctive" then                             // AND
    result ← [ ]
    for combo in product(child_paths) do
        Append [label] + flattened combo to result
    end forreturn result
elsereturn [[label]]
end if
```

The ADT-Webapp only supports only the `OR` and `AND` relationships, but not the sequential AND (`SAND`) operator. To address this, a Python implementation was developed based on the same conversion logic as described in Algorithm 1. This extension enabled trees containing `SAND` nodes to be correctly converted into attack graphs. The resulting converted graphs were then used as visualizations for the questionnaire, ensuring that scenarios containing Sequential AND dependencies were fully supported during the user study.

Algorithm 2 Sequential AND refinement handling

```
if refinement = "sequential" then                    ▷ SEQUENTIAL AND node
    result ← [ ]
    for combo in product(reverse(child_paths)) do
        flat ← flattened list of steps in combo
        Append [label] + flat to result
    end for
    return result
end if
```

As shown in Algorithm 2, the Sequential AND refinement is handled by reversing the order of child paths to ensure that the resulting attack paths preserve the intended sequence of actions. The function returns a set of reversed attack paths, starting at the attacker's goal and ending at the initial actions.

4.2.2 Graph construction

The paths returned by `get_paths` (see Algorithm 1), from the goal to the leaf, are passed to createDisjunctiveXMLFromPaths(paths), which generates an XML structure compatible with the ADT-WebApp (see Algorithm 3). Because a graph expects execution to flow from initial action to final goal, each path is reversed during XML construction to restore the correct order.

Algorithm 3 `createDisjunctiveXMLFromPaths(paths)`

```
root ← new adtree element
initialNode ← new node element with refinement = "disjunctive"
initialLabel ← new label element with text "(s0)"
Append initialLabel to initialNode
Append initialNode to root
stateCounter ← 0
for path in paths do
    reversedPath ← reversed copy of path
    currentNode ← initialNode
    for label in reversedPath do
        stateCounter ← stateCounter + 1
        labelText ← label
        stateText ← "(s" + stateCounter + ")"
        childNode ← new node element with refinement = "disjunctive"
        labelElement ← new label element with text = stateText
        Set edgeLabel attribute of childNode to labelText
        Append labelElement to childNode
        Append childNode to currentNode
        currentNode ← childNode
    end for
end for
return serialized XML string of root
```

As shown in Algorithm 3, this function constructs a new `<adtree>` with an initial node `(s0)` that serves as the common start state. A counter generates unique state identifiers `(s1)`, `(s2)`, ... For each path, we reverse it to obtain the forward execution order (initial action → goal). Each step becomes a `<node>` with disjunctive refinement; the action label is stored as an `edgeLabel` on the child node as the transition from the parent state. Assigning disjunctive refinement to all nodes preserves compatibility with the ADT-WebApp structure, while edge labels indicate the state-to-state transitions.

5 Methods

In this part of the study, we outline the methodology we use to examine how users interpret and understand cybersecurity attack scenarios when presented in two different visual formats: attack-defense trees and attack graphs. Our goal is to evaluate which representation provides greater clarity and interpretability for understanding attack logic, countermeasures, and dependencies through a structured online questionnaire. Here, we want to answer the following research question:

> *How do visual representations (attack–defense trees vs. attack graphs) affect users' clarity and understanding of attack scenarios?*

To make this question more concrete, we focus on three aspects where the two representations may differ:

- **RQ1 (Goal identification):** Does one representation make it easier for users to identify the attacker's main goal?
- **RQ2 (Countermeasures):** Does one representation make the placement and impact of countermeasures clearer?
- **RQ3 (Attack paths):** Does one representation make it easier for users to enumerate valid attack paths?

For RQ1, in both representations the attacker's main goal is highlighted (at the root of the attack tree and at the terminal state in the attack graph). We therefore expect goal identification accuracy to be high and similar for both representations. For RQ2, attack-defense trees attach countermeasures directly to the actions they defend, so we expect them to provide a clearer view of where and how defenses apply than attack graphs. For RQ3, attack graphs lay out all possible attack paths in the graph, whereas in attack-defense trees users need to reconstruct all valid paths from the logical structure themselves. We therefore expect attack graphs to lead to more complete path enumeration.

Based on these sub-questions and prior work on evaluating security diagrams from Lallie et al. [LDB17], we formulate the following hypotheses :

H1: For simple scenarios, there is no substantial difference between attack–defense trees and attack graphs in correct identification of the attacker's main goal (RQ1).

H2: Attack–defense trees will be perceived as clearer for understanding the placement and effect of countermeasures than attack graphs (RQ2).

H3: In more complex scenarios, attack graphs will lead to a higher number of correctly enumerated distinct attack paths than attack–defense trees (RQ3).

Together, these sub-questions and hypotheses guide our analysis of the questionnaire data and address the main research question.

5.1 Study Design

We designed a comparative user study using a structured online questionnaire. The questionnaire contained 26 questions divided across three scenarios and a final evaluation section. Each scenario was presented in both visual formats to enable direct comparison between attack-defense trees and attack graphs for the same underlying model. Our study design follows a similar approach to that of Lallie et al. [LDB17], who compared attack graphs and fault trees using a structured, scenario-based questionnaire. As in their study, we used multiple scenarios and collected both multiple-choice and open-ended responses to evaluate user comprehension and perception. We adapted this design to the context of attack-defense trees and attack graphs, focusing on how users interpret models before and after seeing the other representation.

To ensure participants did not all view the scenarios in the same sequence, we created two versions of the questionnaire: one presenting each scenario in Tree→Graph order and the other in Graph→Tree order. Participants were assigned randomly based on the order of participants (i.e., the first respondent received one version, the next respondent the other, and so on). This ensured an even split between the two groups. In both version, participants evaluated each representation independently by answering multiple choice and open ended questions about attack logic, visibility of goals, valid paths, countermeasures, and overall clarity. After viewing both representations, they were asked to reflect on which format better supported their understanding of the scenario.

A full copy of one questionnaire version, including a short introduction, all the attack scenarios, and all questions, is provided via zenodo [2].

5.2 Participants

We collected responses from 36 participants. Participants were recruited primarily among computer science students at Leiden University, friends and family members, and their colleagues. All participation was voluntary.
The questionnaire asked for the participant's age, general technical background (including IT or cybersecurity experience), and familiarity with tree and graph theory. This information is later used to explore whether technical background or prior familiarity influence comprehension and preferences of one of the representation.

5.3 Ethical Considerations

Participation was voluntary and based on informed consent. No direct identifiers (e.g., name, email address) were collected. The questionnaire stored only task responses plus two non identifying items: age bracket and prior familiarity with attack trees/attack graphs. The data was used solely to assess how the

[2] `https://zenodo.org/records/17905142`

two representations (attack-defense tree vs attack graph) affect clarity and understanding of attack scenarios. All responses were stored securely with access limited to the research team, reported only in a grouped summary form, not shared with third parties.

5.4 Procedure

Data was collected between 17 June and 11 July 2025. Before starting the questionnaire, participants were asked to provide background information, including their age, technical experience in IT or cybersecurity, and familiarity with tree or graph theory. Afterwards, they received a short recap to the three modeling approaches used in this study (attack trees, attack-defense trees, and attack graphs), including a brief explanation of how to interpret logical relations (OR, AND, SAND) and edges.
This short introduction ensured that all participants, regardless of their prior knowledge, shared a common baseline understanding of how to interpret the models. In this way, we aimed to minimize confusion due to unfamiliar notation and to ensure that differences in responses were more likely to arise from the visual representation themselves.

The questionnaire then presented three structured scenarios, each focused on different aspects of model comprehension and gradually increasing in complexity.

- **Scenario 1:** introducing a simple attack structure to assess basic comprehension. Participants were asked to identify the main goal of the attacker, recognize the logical relationship used (OR), and determine possible attack paths.
- **Scenario 2:** introduced countermeasures and combined multiple logical relations. Participants evaluated how clearly the defenses were represented and how they influenced the attack paths.
- **Scenario 3:** integrated all logical relationships (OR, AND, SAND) and increased structural complexity. Participants were asked to identify all valid attack paths, recognized reused nodes, and interpreted more complex dependencies.

After completing all three scenarios, participants filled in a final evaluation section in which they compared the two representations overall and rated how understandable, useful, and informative they found each model.

5.5 Data Analysis

For each scenario and each representation, we derive measures of comprehension and perception from the questionnaire responses that correspond to RQ1, RQ2, and RQ3 and hypotheses H1, H2, and H3. Each scenario focuses on different aspects:

- Scenario 1 mainly addresses RQ1 and H1 (goal identification).
- Scenario 2 mainly addresses RQ2 and H2 (countermeasures).
- Scenario 3 mainly addresses RQ3 and H3 (attack paths).
- The final evaluation section contributes directly to answering the main research question by comparing overall perceived clarity and preference between the two representations.

Goal identification (RQ1, H1). In Scenario 1, participants were asked to identify the attacker's main goal. Responses are marked as correct or incorrect. For each representation, we compute the proportion of correct answers. Because the main goal is explicitly highlighted in both models, we expect accuracy to be high and similar for trees and graphs, in line with H1.

Countermeasures (RQ2, H2). In Scenario 2, we analyze how clearly countermeasures and their effects are understood. This is measured using:

- self-reported ratings of how clear the countermeasures are in each representation (from "very unclear" to "very clear");
- preference questions asking in which representation the role of countermeasures was easier to understand and which representation provided better overall understanding of the scenario.

We summarize clarity ratings as distributions of responses per representation and report counts and percentages for the preference questions. Higher clarity ratings and stronger preferences for attack-defense trees on countermeasure-related questions provide evidence for H2.

Attack paths (RQ3, H3). In Scenario 3, participants were asked to enumerate all valid attack paths in multiple choice form. Each response is evaluated for:

- the number of distinct valid paths listed, and
- whether any invalid paths were included.

For each representation and presentation order, we summarize the distribution of participants who listed all paths, only some paths, or none correctly. We also compute the average number of correctly identified distinct paths per representation. If participants list more complete sets of paths when working with attack graphs than with attack-defense trees, this provides evidence for H3.

Perceived clarity and overall preference (main research question). Across scenarios, and in the final evaluation section, participants rated each representation on their understandability, usefulness, and informativeness, and indicated an overall preference. We analyze these ratings and preferences to compare the two representations on overall clarity and perceived usefulness. Together with the hypothesis-driven analyses for RQ1, RQ2, and RQ3, these results allow us to answer the main research question.

6 Results

In this section, we analyze how well participants understood and interpreted the scenarios with attack-defense trees and attack graphs, focusing on: goal identification (how easily they found the main target), path comprehension (how accurately they recognized all valid attack paths), countermeasure understanding (how well they located and interpreted defenses), and overall preference between the two representations.

6.1 Participant Background

Participants varied in age, technical background, and familiarity with tree and graph theory.

Table 1: Age distribution by experimental order (n=18 per group)

	Tree → Graph	Graph → Tree
Age	**n**	**n**
18–24 years	6	3
25–34 years	2	2
35–44 years	1	4
45–54 years	3	6
55–64 years	5	3
65+ years	1	0
Total	18	18

Table 1 shows the age distribution per presentation order (Tree→Graph vs. Graph→Tree). All predefined age ranges were represented. The Tree→Graph group included one participant aged 65+, whereas the Graph→Tree group did not. The Tree→Graph group contained more participants aged 18–24, and the Graph→Tree group had more participants aged 45–54.

Table 2: IT background by experimental order

Group	**Yes (n)**	**No (n)**	**Total**
Tree → Graph	12	6	18
Graph → Tree	13	5	18
Total	25	11	36

Table 2 summarizes the reported IT background. In total, 25 of 36 participants indicated an IT background (12/18 in Tree→Graph; 13/18 in Graph→Tree).

Table 3: Participant familiarity with tree and graph concepts

Familiarity	**n**
Familiar with both	16
Somewhat familiar with trees only	0
Somewhat familiar with graph theory only	3
Heard of both but don't understand well	8
Not familiar with either	9
Total	36

Table 3 reports familiarity with trees and graphs. 16 participants reported familiarity with both; 9 reported being unfamiliar with either; 8 had heard of them both but did not understand them well; and 3 were familiar with graph theory only. No participants reported being familiar with trees only.
Overall, the sample consisted of a majority of technically experienced participants. Most participants reported an IT-related background, but familiarity with tree and graph theory was more mixed: 16 participants were familiar with both concepts, while 17 either did not understand them well or were not familiar with them at all. Together with the broad age range in both presentation orders, this means that the results mainly reflect the views of a technically oriented group, but still include participants with different levels of prior exposure to tree and graph concepts. This is important to keep in mind when interpreting how general the findings are.

6.2 Scenario 1 - Goal identification

Scenario 1 examined how easily participants could identify the attacker's main goal in each representation, addressing RQ1 and testing H1. Participants viewed the scenario in one of the two presentation orders (Tree→Graph or (Graph→Tree) and indicated which representation made it easier to identify the attacker's main goal. Figure 11 summarizes the preferences.

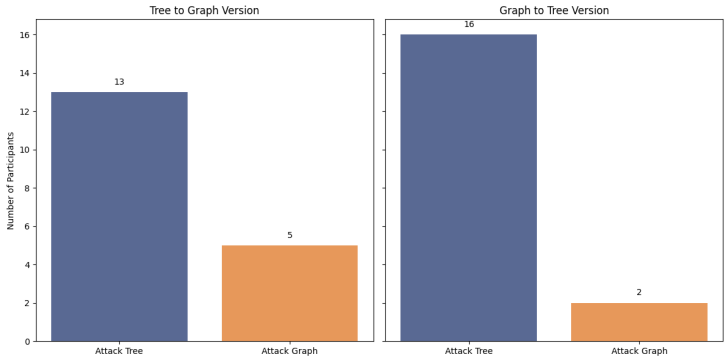


Figure 11: Preferred model for identifying the attacker's goal.

In the Tree→Graph order, 13 of 18 participants (72%) preferred the attack tree and 5 of 18 (28%) preferred the attack graph. In Graph→Tree order, 16 of 18 participants (89%) preferred the attack tree, and 2 of 18 (11%) preferred the attack graph. Thus, in both presentation orders, a clear majority reported that the tree made it easier to identify the attacker's main goal.

In addition to these preferences, almost all participants correctly identified the attacker's main goal in both simple scenarios and both representations. The lock scenario was presented using an attack tree, where 35 out of 36 responses (97%) were correct. The bypass building security scenario was presented using an attack graph, where 35 out of 36 responses (97%) were also correct. Accuracy was therefore very high and showed no indication either representation made goal identification more difficult. Note that in Scenario 1 the lock was always shown as a tree and the building security scenario always as a graph, so we cannot fully separate the effects of scenario content from the effects of representation. Within this limitation, the pattern is consistent with H1: for simple scenarios, both attack-defense trees and attack graphs support correct goal identification to a similar extent, even though participants clearly preferred the tree when asked which representation made the goal easier to see.

6.3 Scenario 2 - Countermeasures

Scenario 2 introduced countermeasures and examined how clearly their placement was understood in each representation, addressing RQ2 and testing H2. Participants were first asked to identify valid first steps toward the sub-goal **Learn Combination** (Question 7, see zenodo [Ger25]), and then to rate and compare the clarity of countermeasures in the tree and the graph.

Table 4: Accuracy on Question 7 (valid steps to learn *Combination*)

	Correct		Incorrect		Total
Group	**n**	**%**	**n**	**%**	**n**
Tree → Graph	13	72	5	28	18
Graph → Tree	11	61	7	39	18

Note: Question 7 was answered based on the *first* representation seen by the participant: the *attack tree* in Tree→Graph and the *attack graph* in Graph→Tree.

Table 4 shows accuracy on Question 7 by initial representation. Participants who first saw the tree answered 13 out of 18 correctly (72%), whereas participants who first saw the graph answered 11 out of 18 correctly (61%). This suggests a small advantage for the tree on this specific step-identification task, but the difference is small and Q7 is only an indirect measure of countermeasure understanding. Our main evidence for RQ2 and H2 therefore comes from the clarity ratings and preference questions.

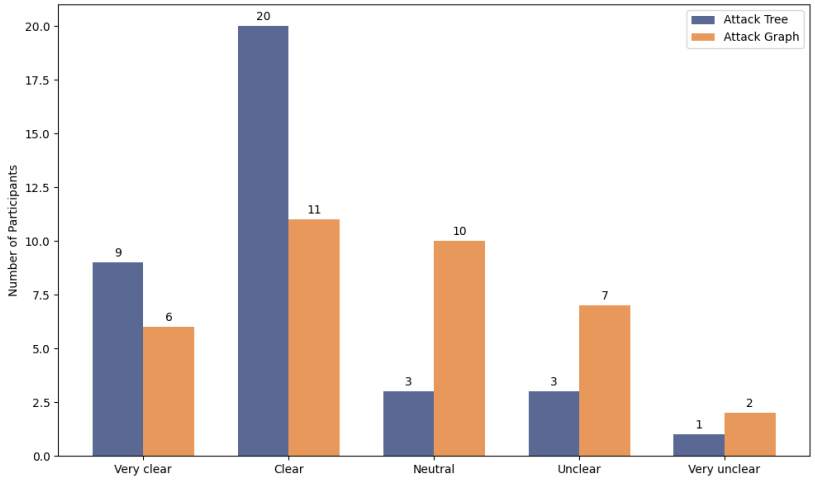


Figure 12: Clarity ratings for countermeasure representation.

Participants rated how clear the countermeasures were in each representation on a scale from "very unclear" to "very clear". Pooled across presentation orders, 29 of 36 (81%) rated the attack-defense tree as "clear" or "very clear", whereas 17 out of 36 participants (47%) did so for the attack graph (Figure 12). Thus, 12 more participants rated the tree as better at showing countermeasures, a difference of 34 percentage points in favor of the tree.

Table 5: Overall preferred representation in Scenario 2 ($n = 36$).

Response	**n**	**%**
Prefer tree	27	75
Prefer graph	4	11
Both equally easy	4	11
Both equally difficult	1	3

Finally, participants indicated which representation provided better overall understanding of Scenario 2. As shown in Table 5, 27 of 36 participants (75%) preferred the tree, 4 of 36 (11%) preferred the graph, 4 of 36 (11%) rated both as equally easy, and 1 of 36 (3%) rated both as equally difficult. Only 5 of 36 participants (14%) did not express a clear preference, which indicates that most participants formed a distinct preference between the two notations, and that this preference strongly favored the tree.
Taken together, these results address RQ2 by showing that participants consistently perceived countermeasures as clearer in the attack-defense tree than in the attack graph. Higher clarity ratings for the tree and a strong majority preference for the tree as providing better overall understanding in Scenario 2 provide clear support for H2.

6.4 Scenario 3 - Path enumeration

Scenario 3 assessed how well participants could enumerate all valid paths in a more complex setting involving AND, SAND, and OR relations, addressing RQ3 and testing H3. Participants were first asked to select all valid paths using only the representation they saw at the start of the scenario. Their performance by initial representation is summarized in Table 6.

Table 6: Correct identification of all attack paths by initial representation.

Outcome	**Tree first** ($n = 18$)	**Graph first** ($n = 18$)
Both correct	3	10
One correct	12	7
None correct	3	1

When the tree was shown first, only 3 of 18 participants listed both required paths correctly, 12 listed one path, and 3 listed none. When the graph was shown first, 10 of 18 participants listed both required paths correctly, 7 listed one path, and 1 listed none. In terms of average performance, participants who saw the tree first correctly identified on average 1.0 of the 2 valid paths, whereas those who saw the graph first identified on average 1.5 of the 2 valid paths. This pattern indicates that, for this more complex scenario, the attack graph helped participants produce a more complete enumeration of the valid attack paths

than the tree, in line with H3.

After viewing both representations, participants were asked which model made it easier to follow all possible paths. Preferences by presentation order are shown in Figure 13.

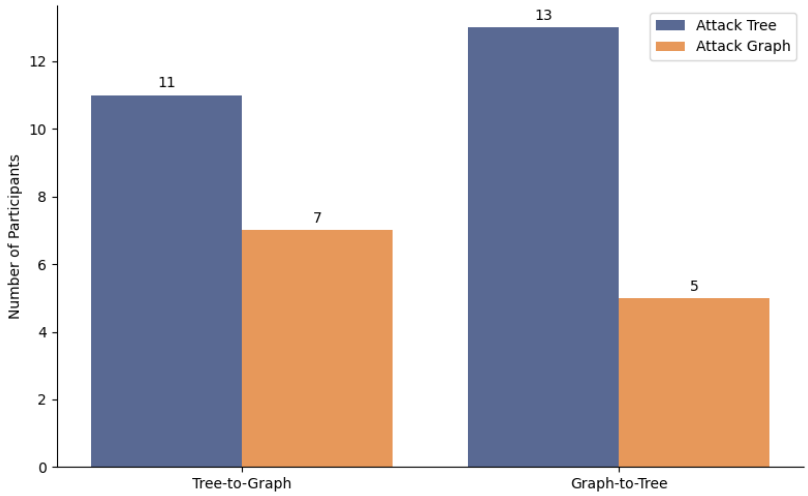


Figure 13: Preferred model for understanding all possible attack paths after viewing both representations.

When participants first viewed the graph, 13 preferred the tree and 5 preferred the graph. When they first viewed the tree, 11 preferred the tree and 7 preferred the graph. Thus, even though the attack graph led to more complete path enumeration when it was used for the initial task, a majority of participants in both orders reported that the tree made it easier to follow all possible paths once they had seen both representations.

Taken together, these results address RQ3: the attack graph supported more complete enumeration of valid paths in the complex scenario, providing evidence for H3, but many participants perceived the attack-defense tree as the easier model to use when identifying which paths are possible.

6.5 Overall preference after all scenarios

At the end of the questionnaire, 29 of 36 participants (81%) indicated an overall preference for the attack-defense tree and 7 of 36 (19%) for the attack graph (Table 7).

Table 7: Overall preferred representation after all scenarios ($n = 36$).

Representation	n
Attack tree	29
Attack graph	7

Overall, this shows a strong preference for the attack-defense tree as the favored representation in our sample.

Overall preference by IT background is summarized in Table 8. Among participants with an IT background, 21 of 25 (84%) preferred the tree and 4 (16%) preferred the graph. Among participants without an IT background, 8 of 11 (73%) preferred the tree and 3 (27%) preferred the graph. Thus, both groups showed a clear majority preference for the attack-defense tree, with only a slightly higher percentage of graph preferences among participants without an IT background.

Table 8: Overall preference by IT background

Group	**Attack Tree**	**Attack Graph**
IT background: Yes	21	4
IT background: No	8	3

We also inspected the overall preference by age group. In every age bracket, the attack-defense tree was the most frequently preferred representation. For example, in the 18–24 and 45–54 age groups, 7 out of 9 participants here preferred the tree over the graph, and in the 55–64 group 5 out of 8 participants preferred the tree. In the 25–34 and 35–44 age groups, all participants (4/4 and 5/5) preferred the tree. Although preferences for the attack graph were somewhat more common in the 55–64 group, the number of participants per age category is small and there is no clear trend across age groups. We therefore do not observe a systematic effect of age on the overall preference.

7 Discussion

This study examined how two visual representations, attack-defense trees and attack graphs, affect user's clarity and understanding of attack scenarios using a structured questionnaire with two different presentation orders. In this section, we interpret the results looking at the research question and hypotheses H1–H3, and discuss what they suggest about when each representation is most useful.

7.1 Goal identification

RQ1 asked whether one representation makes it easier for users to identify the attacker's main goal. H1 stated that, for simple scenarios, there would be no substantial difference between attack-defense trees and attack graphs in correctly identifying the attacker's goal.
The results of Scenario 1 support this hypothesis at the level of accuracy. In both the lock scenario (shown as a tree) and the bypass building security scenario (shown as a graph), 35 out of 36 responses (97%) correctly identified the attacker's main goal. This suggests that, at least for simple scenarios where the main goal is visually highlighted (root of the tree or terminal state in the graph), the choice of representation has little impact on whether users ultimately find the correct goal. In other words, as long as the goal is easy to see, both notations are sufficient for this task.
At the same time, participants clearly preferred the tree when asked which representation made the goal easier to see: in both presentation orders, a large majority selected the attack tree as their preferred representation for goal identification. Because in Scenario 1 the lock scenario is always a tree and the building security scenario always a graph, we cannot separate the representation effects from the content of the scenario. Even so, the results are consistent with H1: both representations support correct goal identification in simple cases, but trees are perceived as more intuitive for spotting the main goal.

7.2 Countermeasures

RQ2 asked whether one representation makes the placement and impact of countermeasures clearer. H2 stated that attack-defense trees would be perceived as clearer for understanding the placement and effect of countermeasures than attack graphs.

Scenario 2 provides several indications that trees are clearer for countermeasures. First, Question 7 asked participants to select all valid first steps toward the sub-goal Learn Combination. Participants who first saw the tree answered 13/18 (72%) correctly, whereas those who first saw the graph answered 11/18 (61%) correctly (Table 4). This suggests a small advantage for the tree on this step-identification task. However, Q7 mixes reasoning about paths and countermeasures and was not designed as a direct measure of countermeasure clarity,

so this difference should be interpreted with caution.

Stronger evidence comes from the clarity ratings and preference questions. When asked how clear the countermeasures were in each representation, 29 of 36 participants (81%) rated the attack-defense tree as "clear" or "very clear", compared to 17 of 36 (47%) for the attack graph (Figure 12). In addition, 27 of 36 participants (75%) reported that the tree provided better overall understanding of Scenario 2, while only 4 (11%) preferred the graph and 5 (14%) expressed no clear preference (Table 5). Together, these findings indicate that most participants found the tree easier to use for understanding where countermeasures are placed and how they influence the attack. This is in line with H2 and supports the idea that visually attaching a defense node directly to the attack step it mitigates helps readers reason about defenses.

7.3 Path enumeration

RQ3 asked whether one representation makes it easier for users to enumerate valid attack paths. H3 stated that, in more complex scenarios, attack graphs would lead to a higher number of correctly enumerated distinct attack paths than attack-defense trees.

Scenario 3 was designed to test this in a setting with OR, AND and SAND relations. When participants were asked to enumerate all valid paths using only the representation they saw first, those who started with the tree performed noticeably worse than those who started with the graph. As shown in Table 6, participants who saw the tree first, correctly identified on average 1.0 of the 2 valid paths, whereas those who saw the graph first, identified on average 1.5 of the 2 valid paths. This difference is in line with H3 and suggests that the attack graph gives more direct support for listing all valid paths in complex scenarios.

These descriptive results suggest that, for this more complex scenario, the attack graph helped participants produce a more complete list of valid attack paths than the attack-defense tree, which is consistent with H3. This contrasts with Scenario 2, where the tree showed a small advantage on the simpler step-identification task in Question 7. This may indicate that trees are fine for small, local decisions, whereas graphs are better suited for checking completeness in complex path structures.
However, reported preferences tell a slightly different story. After seeing both representations in Scenario 3, a majority of participants in both presentation orders still reported that the tree made it easier to follow all possible paths (Figure 13). At the same time, 12 of 36 participants (33%) preferred the graph in this scenario, which is notably higher than in Scenario 2 (4/36) or in the overall preference question after all scenarios (7/36). Thus, the graph became more attractive when the task focused on path enumeration, even though the tree remained the majority choice.

This contrast between performance and preference suggests that participants experience the tree as a more comfortable overall overview, while the graph provides stronger support for exhaustive path reasoning. In other words, H3 is supported at the level of performance in the complex scenario, but this is only partially reflected in participants' reported preferences.

7.4 Overall preference and participants background

We also examined which representation participants preferred overall. After completing all scenarios, 29 of 36 participants (81%) indicated an overall preference for the attack–defense tree and 7 (19%) for the attack graph (Table 7). This confirms that, in this sample, the tree is generally perceived as the more intuitive or clearer notation.
We explored whether this preference might be driven by technical experience. Among participants with an IT background, 21 of 25 (84%) preferred the tree and 4 (16%) preferred the graph; among participants without an IT background, 8 of 11 (73%) preferred the tree and 3 (27%) preferred the graph (Table 8). Because participants with an IT background make up roughly two-thirds of the sample, their responses dominate the overall percentages. However, the non-IT subgroup also shows a clear majority preference for trees. With the current small sample, we do not see evidence that non-IT participants systematically prefer graphs instead, but larger and more balanced samples would be needed to draw stronger conclusions about the role of technical background.

A similar pattern holds for age. In every age bracket, more participants preferred the attack-defense tree than the graph, with some variation in how often the graph was chosen. Given the small number of participants per age category, no clear trend emerges, and we do not observe a systematic effect of age on overall representation preference.

Taken together, these findings answer the research question as follows. Attack–defense trees are typically clearer for showing the overall structure of an attack and how countermeasures apply, whereas attack graphs are clearer for tracing how all permissible attack paths unfold in complex scenarios. This explains the pattern of preferences: most participants favored trees overall, yet graphs provided measurable benefits on path enumeration tasks. In practice, this suggest that the choice of representation should be guided by their goal. Use the attack-defense tree to communicate the big picture and the placement of defenses, and switch to an attack graph when completeness over paths and detailed reasoning about alternative routes is required. Using a tool that allows users to switch between tree and graph representation could therefore exploit the strengths of both representation.

8 Limitations

This work has several limitations that should be considered when interpreting the results.

First, the study involved a relatively small sample size (n = 36), and most participants reported an IT background (25 out of 36). This limits the generalizability of the findings, especially to non-technical populations, because the non IT subgroup is comparatively small (11 versus 25). However, the main preference pattern was similar in both groups: 84% of participants with an IT background and 73% without preferred trees overall. This consistency suggests that the core findings are reasonably robust within this sample, while still needing confirmation in larger and more diverse populations.

Second, the scenarios in the questionnaire were intentionally small and simplified, and the graph visualizations did not include cycles. In real-world settings, attack models are often larger, may contain cycles, and use alternative notations or layouts. These factors could change how users experience complexity and clarity in practice. Working with small, cycle free examples and a consistent visual style was necessary to focus on the differences between attack-defense trees and attack graphs themselves, rather than on difference in notation or domain. The findings should therefore be seen as a baseline: they show the basic strengths of each representation under controlled conditions, while the size and structure of other models may influence how strong these effects are.

Third, the questionnaire measured immediate comprehension of static visualizations within a single session. Participants did not build the models themselves or use interactive tools. As a result, the findings mainly reflect first impressions and short-term understanding, rather than how preferences might change with training or repeated use. However, first impression clarity is important in practice as well, for example when explaining attack scenarios or deciding which representation to show in a tool. The results therefore provide a useful starting point for choosing which representation to use when explaining attack scenarios or supporting analysis.

Finally, the study used a single tree-to-graph conversion in the ADT-WebApp, keeping the underlying model constant across representations. This means that differences in participants' responses mainly reflect the visual structure of trees and graphs, rather than the changes to the model. At the same time, it is a limitation: other conversion strategies or visual styles for attack graphs might lead to different usability and preference patterns, and could be explored in future work.

9 Conclusion & Future Work

This thesis investigated how two visual representations, attack–defense trees and attack graphs, affect users' understanding of attack scenarios. To do so, we extended the ADT-WebApp with support for SAND logic and a tree-to-graph conversion, and conducted a questionnaire study with 36 participants comparing goal identification, path enumeration, interpretation of countermeasures, and overall preference.

The questionnaire results suggest a division of strengths between the two representations. In our sample, attack-defense trees were more often experienced as clearer for communicating the overall structure of an attack and how countermeasures apply. The tree was preferred in both representation orders for identify the main goal, was rated clearer for countermeasures, and was chosen as the overall preferred representation by 81% of participants after all scenarios. Attack graph, in contrast, provided observable benefits on path enumerating tasks in the more complex scenarios with AND and SAND relations: participants who first saw the graph were more likely to list all valid paths correctly, and on average, identified more valid paths than those who first saw the three. These patterns were similar for participants with and without an IT background, although the sample was small and skewed towards technically experienced participants. This suggests that that difference in preference are more likely related to the structural and visual properties of the representations than by solely technical expertise.

Taken together, our findings indicate that, in this study, attack-defense trees tended to work better for presenting the "big picture" of an attack and the placement of defenses, whereas attack graphs tended to offer stronger support for tracing all permissible attack paths in more complex scenarios. This helps explain the overall pattern: most participants favored trees as the more intuitive notation, while graphs provided measurable advantages on path enumeration tasks. In practice, this points to a task dependent choice of representation: attack-defense trees appear more suitable when the goal is to communicate the main goal and where defenses act, whereas attack graphs appear more suitable when completeness over paths and detailed reasoning about alternative route is required. Tools that support switching between tree and graph views can exploit the strengths of both representations.

Future work could replicate this study with larger and more diverse participant samples and more complex models, and could evaluate interactive tooling that allows users to switch between tree and graph representations while performing realistic security analysis tasks.

9.1 Acknowledgement

I would like to thank my supervisors, Nathan Schiele and Olga Gadyatskaya, for their guidance and support throughout the thesis. Their feedback helped me to better understand threat modeling with attack-defense trees and attack graphs.
I would also like to thank my parents and my friend Robin The for their continued encouragement and support throughout my bachelor. Thank you.

References

[AWJS16] Nawaf Alhebaishi, Lingyu Wang, Sushil Jajodia, and Anoop Singhal. Threat modeling for cloud data center infrastructures. In *International symposium on foundations and practice of security*, pages 302–319. Springer, 2016.

[BFM04] Eric J Byres, Matthew Franz, and Darrin Miller. The use of attack trees in assessing vulnerabilities in scada systems. In *Proceedings of the international infrastructure survivability workshop*, volume 202. Lisbon, 2004.

[BSCS19] Delphine Beaulaton, Najah Ben Said, Ioana Cristescu, and Salah Sadou. Security analysis of iot systems using attack trees. In *International Workshop on Graphical Models for Security*, pages 68–94. Springer, 2019.

[CGH+19] Frank Capobianco, Rahul George, Kaiming Huang, Trent Jaeger, Srikanth Krishnamurthy, Zhiyun Qian, Mathias Payer, and Paul Yu. Employing attack graphs for intrusion detection. In *Proceedings of the new security paradigms workshop*, pages 16–30, 2019.

[FW20] Barbara Fila and Wojciech Wideł. Exploiting attack–defense trees to find an optimal set of countermeasures. In *2020 IEEE 33rd computer security foundations symposium (CSF)*, pages 395–410. IEEE, 2020.

[Ger25] Renee Gerritse. From attack trees to attack graphs: Implementing conversion and user evaluation: Dataset, 2025.

[GMSW06] Dominik Grolimund, Luzius Meisser, Stefan Schmid, and Roger Wattenhofer. Cryptree: A folder tree structure for cryptographic file systems. In *2006 25th IEEE Symposium on Reliable Distributed Systems (SRDS'06)*, pages 189–198. IEEE, 2006.

[Gou12] Ronald Gould. *Graph theory*. Courier Corporation, 2012.

[GYA18] Jonathan L Gross, Jay Yellen, and Mark Anderson. *Graph theory and its applications*. Chapman and Hall/CRC, 2018.

[HKA17] S Haque, M Keffeler, and T Atkison. An evolutionary approach of attack graphs and attack trees: A survey of attack modeling. In *Proceedings of the International Conference on Security and Management (SAM)*, pages 224–229. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2017.

[ICL+09] Kyle Ingols, Matthew Chu, Richard Lippmann, Seth Webster, and Stephen Boyer. Modeling modern network attacks and countermeasures using attack graphs. In *2009 Annual Computer Security Applications Conference*, pages 117–126. IEEE, 2009.

[JKM+15] Ravi Jhawar, Barbara Kordy, Sjouke Mauw, Saša Radomirović, and Rolando Trujillo-Rasua. Attack trees with sequential conjunction. In *IFIP International Information Security and Privacy Conference*, pages 339–353. Springer, 2015.

[KKMS10] Barbara Kordy, Piotr Kordy, Sjouke Mauw, and Patrick Schweitzer. Foundations of attack–defense trees. In *International Symposium on Formal Aspects of Security and Trust*, pages 80–95. Springer, 2010.

[LDB17] Harjinder Singh Lallie, Kurt Debattista, and Jay Bal. An empirical evaluation of the effectiveness of attack graphs and fault trees in cyber-attack perception. *IEEE Transactions on Information Forensics and Security*, 13(5):1110–1122, 2017.

[LDB18] Harjinder Singh Lallie, Kurt Debattista, and Jay Bal. Evaluating practitioner cyber-security attack graph configuration preferences. *Computers & Security*, 79:117–131, 2018.

[LDB20] Harjinder Singh Lallie, Kurt Debattista, and Jay Bal. A review of attack graph and attack tree visual syntax in cyber security. *Computer Science Review*, 35:100219, 2020.

[LMG17] Patrick Luckett, J Todd McDonald, and William Bradley Glisson. Attack-graph threat modeling assessment of ambulatory medical devices. *arXiv preprint arXiv:1709.05026*, 2017.

[MO05] Sjouke Mauw and Martijn Oostdijk. Foundations of attack trees. In *International Conference on Information Security and Cryptology*, pages 186–198. Springer, 2005.

[Moh23] MM Mohalaia. *Implementing a User Interface for AttackDefense Tree WebApp Using Human-Computer Interaction Principles*. PhD thesis, Bachelor's Thesis, LIACS, Leiden University, 2023.[Online]. Available: https . . . , 2023.

[NH25] Chiu Yeong Ng and Mohammad Khatim Bin Hasan. Cybersecurity serious games development: A systematic review. *Computers & Security*, 150:104307, 2025.

[PAV14] Sophie Pinchinat, Mathieu Acher, and Didier Vojtisek. Towards synthesis of attack trees for supporting computer-aided risk analysis. In *International Conference on Software Engineering and Formal Methods*, pages 363–375. Springer, 2014.

[RGG24] Atul Rana, Sachin Gupta, and Bhoomi Gupta. A comprehensive framework for quantitative risk assessment of organizational networks using fair-modified attack trees. *Frontiers in Computer Science*, 6:1304288, 2024.

[Sal24] Marwa Salayma. Threat modelling in internet of things (iot) environments using dynamic attack graphs. *Frontiers in the Internet of Things*, 3:1306465, 2024.

[Sch99] Bruce Schneier. Modeling security threats. *Dr. Dobb's journal*, 24(12), 1999.

[SDP08] Vineet Saini, Qiang Duan, and Vamsi Paruchuri. Threat modeling using attack trees. *Journal of Computing Sciences in Colleges*, 23(4):124–131, 2008.

[SG21] Nathan Daniel Schiele and Olga Gadyatskaya. A novel approach for attack tree to attack graph transformation. In *International Conference on Risks and Security of Internet and Systems*, pages 74–90. Springer, 2021.

[SHJ+02] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M Wing. Automated generation and analysis of attack graphs. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pages 273–284. IEEE, 2002.

[SK20] Sakshi Singh and Suresh Kumar. The times of cyber attacks. *Acta Technica Corviniensis-Bulletin of Engineering*, 13(3):133–137, 2020.

[SS19] Sam Scholefield and Lynsay A Shepherd. Gamification techniques for raising cyber security awareness. In *International conference on human-computer interaction*, pages 191–203. Springer, 2019.