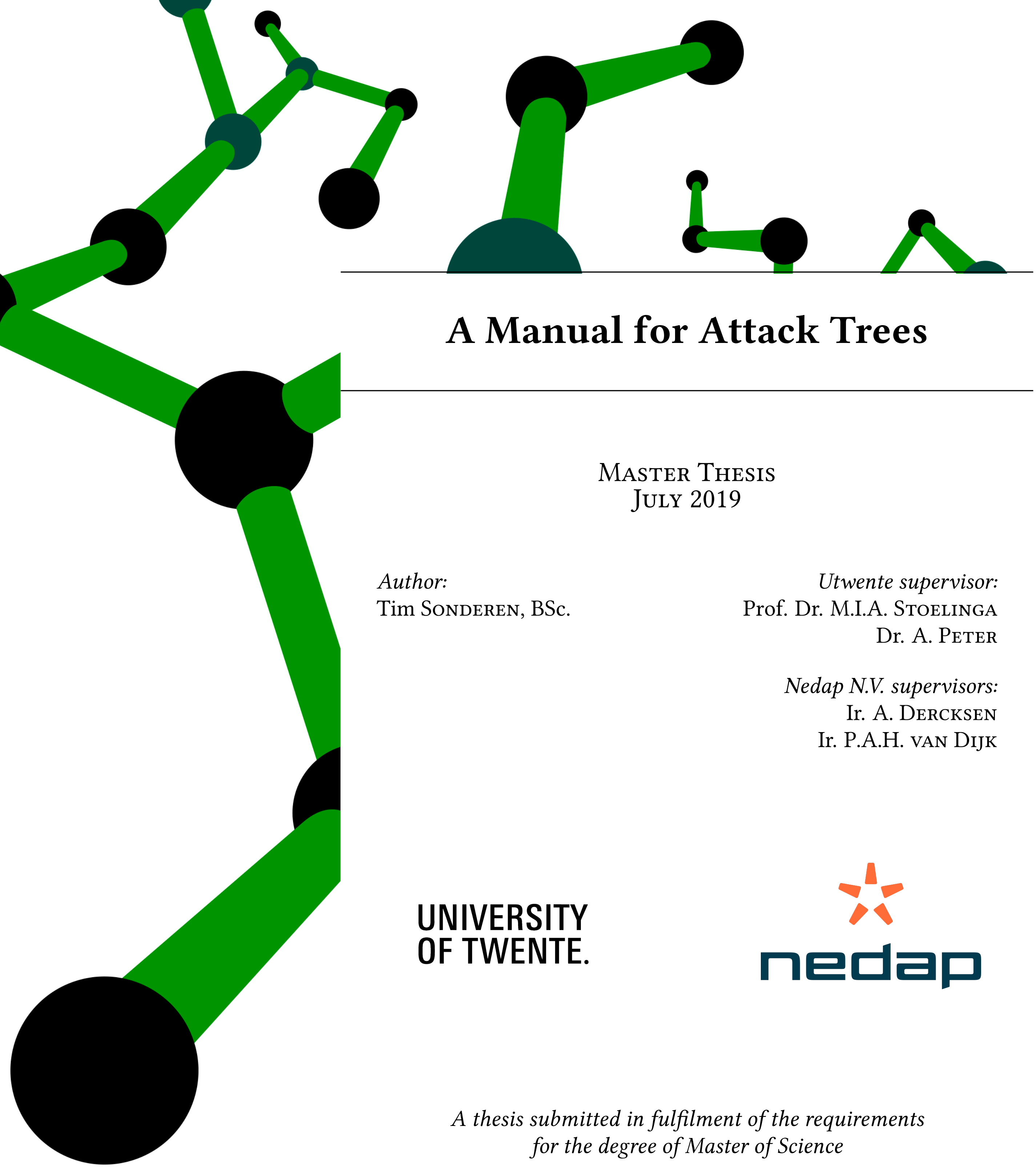


A Manual for Attack Trees

Master Thesis
July 2019

Author:
Tim Sonderen, BSc.

Utwente supervisor:
Prof. Dr. M.I.A. Stoelinga
Dr. A. Peter

Nedap N.V. supervisors:
Ir. A. Dercksen
Ir. P.A.H. van Dijk

UNIVERSITY
OF TWENTE.

nedap

A thesis submitted in fulfilment of the requirements
for the degree of Master of Science

in the field of

Computer Science

UNIVERSITY OF TWENTE

Abstract

Faculty of Electrical Engineering, Mathematics and Computer Science
Formal Methods and Tools

Master of Science

A Manual for Attack Trees

by Tim Sonderen, BSc.

Nowadays attack trees are often used by large organisations to analyse security threats against their systems. Designing such an attack tree requires detailed knowledge regarding attack trees and the systems to be analysed. In many cases this process relies heavily on personal experience and principles. This causes significant variance between attack trees.

In this thesis, guiding principles and building blocks that are used by experts in the field of attack trees have been analysed in an attempt to further standardise attack trees. This was done by analysing attack trees that have been created in the most prominent papers that regard attack trees. These principles and building blocks were then used to design a model for attack trees that specifies the structure of an attack tree in more detail, as well as an accompanying manual.

To evaluate it, system experts have been asked to create an attack tree for a semi-realistic case; First with only basic knowledge of attack trees, and thereafter with the help of the manual. The model has proven to improve attack discovery and understandability of the resulting attack trees. Additionally, the results were used to iteratively improve the manual. After this test, the model and manual were used in a real case study for Nedap N.V. and evaluated in a more qualitative manner.

Overall, the manual improved the experience of the user. However, the most significant improvements were made in attack discovery, improved detailing and in the understandability when evaluated by others. The model and manual stimulate attack discovery while simultaneously guiding the user towards creating a well structured attack tree.

Besides improvements for the manual creation of attack trees, the model provides opportunities for further automating the creation of attack trees.

Acknowledgements

I would like to thank my supervisors Mariëlle Stoelinga, Andreas Peter, Albert Dercksen and Peter van Dijk for their support and guidance. I also want to thank Nedap N.V. for welcoming me into the R&D team and for providing the case for the case study. Furthermore i want to thank everyone who participated in the experiments for their time and effort. Finally i want to thank my family and friends for supporting me during this time.

Contents

List of Figures

List of Tables

List of Abbreviations

AFT	**A**ttack **F**ault **T**rees
BDMP	**B**oolean logic **D**riven **M**arkov **P**rocesses
FES	Digital and Physical **F**orce, **E**xploits and **S**ocial engineering
GAP	**G**oal, **A**ttack, **P**rocess
OWA	**O**rdered **W**eighted **A**verage
SAND	**S**erial **AND**
SO	**S**pecification and **O**rganisation
SOR	**S**erial **OR**

Part I

Outline

In this part, the motivation and a small introduction will be given. From there the goal will be set and a number of research questions will be defined to support the process towards that goal. Finally, the methodology will be described which describes the rest of this thesis.

Chapter 1

Introduction

Being the target of a malicious attack is a serious concern for many companies and organisations, especially when they are a vital part of modern infrastructure such as telecom / utility providers, power plants and banks. An important part of these companies is analysing and managing risk of these malicious attacks. Now the European Union is making the laws of data protection stricter, causing smaller companies to face the need of risk analysis to protect their data and to gain security certificates.

Naturally, many models have been developed to aid the analysis of such threats. One of those models is the model of attack trees. Attack trees are particularly suitable for analysing the security of a system against malicious attackers. It puts the security expert in the shoes of an attacker to gain new insights in vulnerabilities of the system. In most cases the analysis is focused on a single goal. Such a goal would often be a goal that hurts the system or company, but also benefits the attacker. Attack tree analysis can be used for virtually any system, be it physical or digital.

In Figure 1.1 an attack tree for breaking into a bank vault is shown. The main goal is at the root of the tree: getting inside bank vault. That could be accomplished by either gaining access to the vault or by breaking into it. Gaining access could be done by blackmailing an employee with access or by infiltrating the bank e.g. get a job there. Breaking in could be done by blowing up the vault or by cracking the lock. As opposed to the other attacks, blowing up the vault requires multiple (non-trivial) things to be accomplished: Acquiring explosives and getting those explosives close to the vault.

Such an attack tree analysis potentially provides insight into attacks that were never thought of before, where weak spots are, and where improvements to the system have most value. This information could be used for important business decisions such as getting a new security system or moving or splitting important assets to other locations.

The basics of creating an attack tree are rather simple. The challenging part lies in creating one that can provide useful insights into the vulnerabilities of the system. This includes not getting lost in what to include and what to leave out so that the attack tree does not become unwieldy. Additionally, some parts of the system might be impossible to properly model using standard attack trees. Therefore multiple extensions on attack trees have been developed to solve such problems. However these extensions are described across a large number of papers and therefore not easily accessible or combinable for security or system experts. To retain all this (tacit) knowledge, larger companies often hire a dedicated security officer. Smaller companies however, do generally not have the resources for this.

In this thesis a model for a more defined structure for attack trees is proposed, as well as a manual for creating attack trees that comply with this model. The model defines structuring techniques such as layers and a set of common splits to standardise a structure for attack trees which is based on attack trees created by the most prominent researchers in the field of attack trees. This model not only improves the clarity of the attack tree, but also helps creators of attack trees discover more attacks. The accompanying manual provides an easier

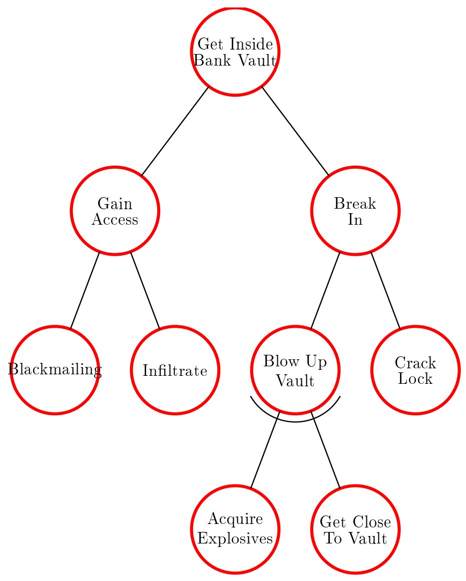


FIGURE 1.1: Example attack tree for a bank vault

entry point into the field of attack trees and provides an explanation on how this extended structure should be used.

The model and manual were evaluated and improved by a quantitative analysis of 34 semi-realistic cases. Thereafter the final version of the manual was evaluated in a qualitative analysis. This was done with a case study for Nedap N.V.

By standardising a more defined structure for attack trees, opportunities are created for better tooling and support in the process of creating attack trees. The model will also make it easier to start in the field of attack trees, giving the ability to analyse company security to anyone in the firm. The only requirement still left is knowledge of the system itself.

Chapter 2

Objectives

2.1 Research Questions and Goal

As mentioned in the introduction, the **goal** of this thesis is the following:

> *Allow (security) experts to be able to create a well-structured attack tree within an acceptable time frame*

This would reduce the time and resources needed to spend on researching attack trees before being able to use this analysis technique to analyse the security of your system or company.

One way of achieving this goal could be to design a manual for the creation of attack trees and for the attack trees themself. First however, there has to be a definition of a 'well-structured attack tree'. Besides for the purpose of the manual, by defining what such a tree looks like, an opportunity is created for standardisation and better tooling for attack trees.

Second, a first version of a manual such as mentioned before can be designed. Then, this manual will be tested for its usefulness and improvements will be made accordingly. This last step can be repeated either until the desired result is achieved or it can be concluded that a manual does not help.

This process results in the two following main **research questions**:

R.1 *What quantitative or qualitative metrics indicate that an attack tree is structurally well made for the use of security analysis?*

R.2 *Which guiding principles and building blocks are used by experienced security experts?*

R.3 *To what extend does a manual improve the ease and speed of the manual creation of an attack tree?*

Questions R.1 and R.2 will be answered in Chapter 6 and refined in Chapter 7. Question R.3 will be answered in Chapter 8. Naturally, all three research questions and the research goal will be recapped and answered in the conclusions in Chapter 10

Chapter 3

Related Work

In this chapter related work is described. These papers are also listed in table 3.1 together with a short description of their contributions. In the table as well as in the following section the papers are split into five categories and described in chronological order to easily determine the direction in which the research field is going.

In 2014, Kordy, Piètre-Cambacédès, and Schweitzer [1] presented the state of the art at that moment. In their paper they summarise (to our knowledge) all forms of attack defence modelling that were introduced at that moment.

3.1 Attack Trees

The first mention of using a logic tree for security assessments was mentioned by Weiss [2], they introduced threat logic trees. These trees were based on the previously existing fault trees.

The root of a threat logic tree is the main goal of a hypothetical attacker. From there, each node contains an action that can or must (OR-node or AND-node respectively) be accomplished to reach the goal of its parent. These actions can be split into smaller actions until the action becomes trivial, at which point it becomes a leaf. The nodes are connected by edges, these contain no extra information though.

Each leaf is assigned a level (1 to 10) of negative impact to the system and a level (1 to 10) of effort required from the attacker. From this the risk of the action happening is calculated. Next, these values are calculated for the rest of the tree bottom-up. An OR-node simply takes the highest values of its child nodes and for an AND-node the values are manually reconsidered.

Schneier [3] formalises these threat logic trees into attack trees and describes a methodology for analysing the security of systems using these trees. Mauw and Oostdijk [4] continued this, further formalising attack trees and introducing the concept of attack suites. These attack suites correspond to one path trough an attack tree that consists of one full attack. They also suggested that the value of attack trees lies mostly in the creation of them, not necessarily in the calculation of the semantics. Later, Whitley, Phan, Wang, *et al.* [5] introduced another way of assigning values to attack trees and the propagation of these values. Their technique was inspired from the field of electrical circuit analysis.

Ingoldsby [6] and Pieters, Hadziosmanovic, Lenin, *et al.* [7] introduce a new way of assigning values to attack trees. They propose that the values for the fields each node has should be independent of the attacker. It should represent e.g. a minimum skill level, a minimum amount of time spend or a minimum amount of risk that the attacker is willing to take. Then, the types of possible attackers should be determined and analysed. In a perfect scenario, the result of this is a graph for each field for each attacker that indicates how large or small of a sacrifice the attacker is willing to make in that field. Finally,

Paper	Year	Model	Contribution
[2]	1991	Attack tree	First notion, structure, simple semantics
[3]	1999	Attack tree	Formalisation, analysis methodology
[4]	2006	Attack tree	Formalisation, attack suites
[5]	2011	Attack tree	Attributes and propagation
[6]	2013	Attack tree	Separate attacker from tree
[7]	2014	Attack tree	Plug-and-play attackers
[8]	2005	Defence tree	Defence leafs, gamification
[9]	2008	Defence tree	Answer set optimisation
[10]	2006	Protection tree	Formalise propagation, separate defence
[11]	2009	Attack response tree	Defence on any tree level, defence prediction
[12]	2012	Attack countermeasure tree	Avoid state-space explosion, optimise defence placement
[13]	2014	Attack defence tree	Defence on any tree level, analysis methodology, tooling
[14]	2003	Fault tree	Components
[15]	2006	Attack tree	OWA-nodes
[16]	2010	Attack tree	Temporal order in semantic calculation
[17]	2010	Attack tree	Attack jungles
[18]	2010	Attack tree	Boolean logic driven markov processes
[19]	2012	Attack tree	Bayesian networks
[20]	2015	Attack tree	SAND- and SOR-nodes, extended metric analysis
[21]	2017	Attack Fault tree	Combine attack- and fault trees
[22]	2002	Attack tree	Automatic generation
[23]	2013	Attack defence tree	ADTool
[24]	2013	Attack defence tree	Quantitative questions and use of ADTool
[25]	2016	Attack tree	Soundness between model and reality
[26]	2017	Attack tree	Correctness between model and reality
[27]	2019	Attack tree	Visualisation, tooling
[28]	2019	Attack tree	Analysis, tooling
[29]	2019	Attack tree	ADTool, Tooling
[30]	2016	Attack defence tree	Case study ATMs
[31]	2019	Attack defence tree	Benchmarks, case study database

TABLE 3.1: Summary of the related work.

from these values Ingoldsby describe how to calculate multiple quantifiers such as rate of occurrence, absolute risk and annual loss.

3.2 Defence Trees

After these formalisations the notion of adding defences to attack trees arose: Bistarelli, Dall'Aglio, and Peretti [8] introduced defence trees. These trees are like attack trees, but extended with possible defences at the leaves. In addition to adding defensive leaves, they also demonstrated the use of game theory, in combination with the concepts 'return on investment' and 'return on attack', to determine which of the defences was best to implement (first). Later, they extended this prioritisation with the use of answer set optimisation [9]

At that same time Edge, Dalton, Raines, *et al.* [10] also introduced an extension to attack trees in order to include defences. Though he first revisited the assignment of metrics to leaves of attack trees and the propagation of those metrics up the tree structure. These metrics no longer have to be re-evaluated for every AND-node, but have a defined function for propagation. These metrics can then be used to better analyse the attack tree to find the easiest paths. Being able to pinpoint the weakest points, Edge, Dalton, Raines, *et al.* added the use of protection trees to complement attack trees. Such a protection tree is created by by inverting the attack tree so that for each attack (but with priority for the weakest points), there is a protection against it. By creating protection trees and comparing the cost of a node to the corresponding damages on the attack tree node, one can determine where it is most feasible to implement protections.

Later, Zonouz, Khurana, Sanders, *et al.* [11] advanced this concept to attack response trees which are attack trees that can have a defence under any node in the tree. Furthermore, they added a response and recovery engine which systematically predicts where a defence could be added, and what attack could be added to that defence. Naturally, this process quickly expands the state-space. Then Roy, Kim, and Trivedi [12] continued this work, resulting in attack countermeasure trees. These attack countermeasure trees avoid creating and solving a state-space and implement the defensive nodes, or countermeasures, in the form of detection and mitigation events.

Kordy, Mauw, Radomirovic, *et al.* [13] introduced the formalism of attack-defence trees, which is also a tree that can have defensive nodes at any level. They also discussed the appropriate analysis algorithm to analyse these trees. Finally they also mentioned that they are developing tool support for the attack-defence trees formalism and their plans to extend this to a directed acyclic graph.

3.3 Extensions

Kaiser, Liggesmeyer, and Mackel [14] developed the concept of components for fault trees. Even though their research was not specifically aimed at attack trees, attack trees are derived from fault trees. Therefore, such components will also work very well in attack trees. The use of components allows for an attack tree to be split up in more manageable parts. For example, if there is an expert for the digital part of the system and an expert for the physical part of the system, they can both create their part of the complete attack tree. These can then be combined as components.

On a lower level a component can represent a subtree that can be reused e.g. the subtree for a normal door. These components have predefined input and output ports and can replace any part of an attack tree. Furthermore the components allow a black-box type of usage. This means that a whole subtree can be compacted into one node so that it decreases the size of the tree and unnecessary or secret specifics are hidden.

Yager [15] introduced OWA (Ordered Weighted Average) Trees. These are built from OWA nodes instead of the classic AND- and OR-nodes. An OWA node makes it possible to model how many of its children should be satisfied in order to satisfy the node itself. They even allow probabilistic uncertainty of the number of children that needs to be satisfied.

Jürgenson and Willemson [16] introduced a temporal order in the attacks that can be carried out by the attacker. This brings the model closer to reality and is in some cases more intuitive. They found that with this temporal order, the attacker could achieve better outcomes. Jürgenson and Willemson also introduced a complementary algorithm for analysis of this new form of attack trees. Finally, they also shortly discussed a generalisation from attack trees to rooted directed acyclic graphs to avoid duplicating whole subtrees.

This generalisation is continued by Abdulla, Cederberg, and Kaati [17] as they introduce attack jungles, which are directed acyclic graphs. These can contain multiple roots, reusable nodes and allow cycles to exist. This makes for a more efficient and perhaps more intuitive model. This comes at the cost of more complexity, but this might be partially solved by good tool support. Naturally, they also introduce an algorithm to analyse this structure.

Piètre-Cambacédès and Bouissou [18] introduce the use of Boolean logic Driven Markov Processes (BDMP) in attack trees. BDMPs are dynamic, therefore they make it possible to model attack sequences. Besides attack sequences, they also allow the modelling of defensive aspects such as mitigation.

Poolsappasit, Dewri, and Ray [19] involve Bayesian networks into attack graphs. These Bayesian attack graphs support modelling and better understanding of nodes that are dependant on other nodes like in Bayesian networks. In turn, they show that this can help system administrators by optimising their resources.

Kumar, Ruijters, and Stoelinga [20] use priced timed automata to analyse attack trees. This analysis method can provide quantitive and qualitative information about optimal attack paths and values while allowing subtrees and retaining the intuitive representation of attack trees. Furthermore, they define the semantics of SAND- and SOR-nodes.

Kumar and Stoelinga [21] introduce Attack Fault Trees (AFT). These AFTs merge attack trees with fault trees to one model that combines both malicious attacks and accidental system failures. In addition to this notion, they add the transition from AFTs to stochastic timed automata to be able to analyse the metrics of AFTs. These metrics include cost, time and damages for different adversaries.

3.4 Automation and Validation

Soon after the formalization of attack trees the first notion of automatically generating them came along. This was first mentioned by Sheyner, Haines, Jha, *et al.* [22]. They propose an algorithm to automatically generate an attack tree based on a finite state diagram. Furthermore, they also introduce two algorithms for analysing the generated attack trees.

Audinot and Pinchinat [25] define three notions of soundness of attack trees. This soundness targets the correspondence between the attack tree and the original system. The concepts they introduce are admissibility, consistency and completeness. In [26] Audinot, Pinchinat, and Kordy use these concepts to determine the correctness of an attack tree. They design a framework to analyse and express consistency of the attack tree and the system. However, this framework and the concepts it includes are limited to labelled transition systems.

Also, multiple projects such as TREsPASS [27], SecurITree [28] and SaToSS [29] have focused on creating tool support for attack and defence trees. All three of these have a different aim: TResPASS focuses on visualisation of risks while SecurITree is used and sold by Amenaza to consult companies. SaToSS has produced the free open source tool ADTrees for modelling attack trees [23]. Finally, Kordy, Mauw, and Schweitzer [24] shows how create well formed quantitative questions and how to use them to make an attack tree.

3.5 Case Studies

Of course many case studies have been done. In this section we list some of them. These will, among others, be used to study properties of a good attack tree.

Fraile, Ford, Gadyatskaya, *et al.* [30] report on the application of attack defence trees for analysing the security of ATMs. In their study they found the ease of the ADTree tool helps when creating a tree together with specialists who might not be familiar with the concept even though this was already incorporated in normal attack trees. Furthermore, they found that the corporate wish to include defences and mitigations brings an advantage for ADTrees over normal attack trees.

Kramer [31] are working on setting a benchmark for attack trees. They anonymise attack trees and collect them in a publicly available database. This way their syntactic structure can be analysed and statistics can be generated on how many trees use certain concepts.

Chapter 4

Background

4.1 Scope

In this chapter we introduce the type of attack tree that we will use for the rest of this thesis. This type of attack tree is based on the basic form of an attack tree, together with some of the extensions mentioned in the previous chapter. We use these extensions for two reasons. The first reason is that these they make using the attack trees more intuitive. They provide functionality that we believe is more in line with the real world e.g. temporal order and a specified number of children to be satisfied. The second reason is that this makes the attack trees more compact and gives the creator of such attack trees more freedom in general. There are more options to combine nodes, or just point to a different part of the tree in order to avoid duplicate sub-trees. The reason that we do not use all available extensions is to not add more complexity. Furthermore, the main focus in this thesis lies in the qualitative aspects of the attack tree, i.e. the structure.

Therefore in this thesis we use the formalized attack tree structure [3] [4] and the addition of defence nodes [13], together with the concepts temporal order in the form of SAND (Serial AND) nodes [16] and components [14] as well as a new way of assigning values to these trees [6].

4.2 Attack Trees

4.2.1 Notation

Before defining the possibilities regarding the building of attack trees it is useful to define the notation of those attack trees. Kordy, Kordy, Mauw, *et al.* [23] developed a latex style file for creating attack-defence trees. We made some small additions so it fits better to our needs.

In figure 4.1 the structure of the attack tree is displayed. This will be discussed in the next section, however it also displays all parts of an attack tree:

- Attack node: Red circle
- Defence node: Green rectangle
- Tree component: Black triangle
- Normal relation: Solid line
- Counter relation: Dotted line

Splits In the previous list, as well as the rest of this thesis, the process of subdividing nodes that represent an action into children that represent the proper sub-actions will often be referred to as splitting nodes.

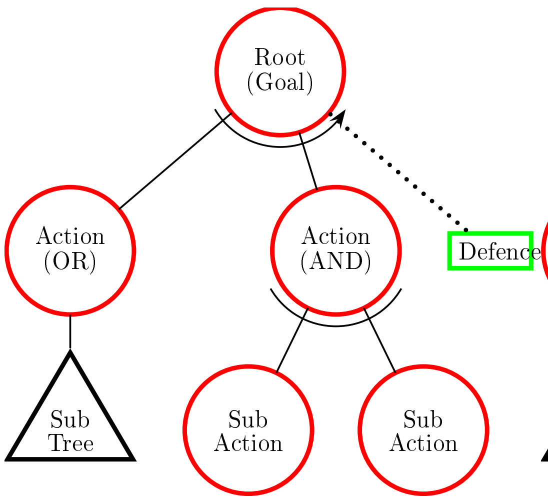


FIGURE 4.1: Basic attack tree structure

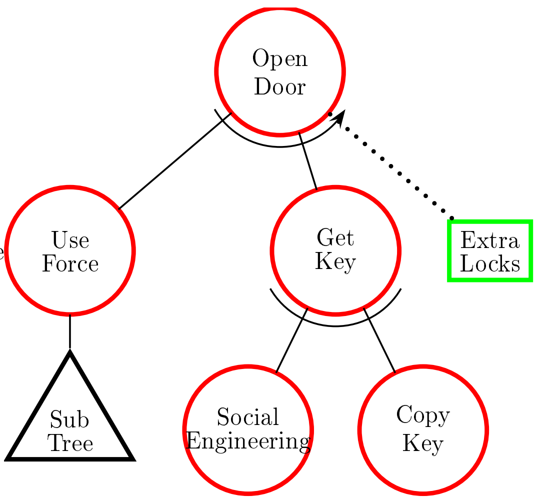


FIGURE 4.2: Simple attack tree for a door

Layers and Rows In the rest of this thesis a row will be defined as a horizontal row of nodes, so all nodes that are on the same level. This level is not necessarily the depth of the tree as nodes can be placed lower than they normally would. In that case the row is skipped.

Furthermore, layers are a collection of rows.

4.2.2 Top-Down Structure

An attack tree consists of nodes. Each node represents an action and is properly named after that action. Therefore, in this thesis the simulation of an action that is represented by a node will often be referred to as executing a node. When building an attack tree, the analyst starts with the root node, which is at the top of the tree. The root is the first node of the tree and represents an attack on the system. From there, if the node is not trivial, the attack is split up in actions required to successfully complete the attack. These sub-actions become child nodes and can include multiple ways of achieving the goal of the parent node. Each node is either an OR-, AND- or SAND- node. This determines which of its child nodes should be executed in order to complete it. In the case of an OR-node *any* of its children have to be executed. In the case of an AND-node *all* of its children have to be executed. The SAND-node is a special case of an AND-node where the children have to be executed in a specific order (in most cases from left to right).

This process of splitting the action, assigning children and determining whether the node should be an OR-, AND- or SAND-node can be repeated for each newly created node until the action can be seen as trivial. The basic structure of an attack tree and a potential (incomplete) realisation of it are displayed in figures 4.1 and 4.2 respectively.

It is also possible to add defence nodes. For example; there is an attack node, but there is already a counter in place a defence node can be added there. Then, from that defence node a new subtree can be created again to expand the defence or to model an attack on that defence.

Finally, each subtree can become a component. Not only can these components be compacted to a single node, they can also be copied and referenced from other nodes in the tree. Components are included, because they have the potential to greatly decrease the size of an attack tree. Furthermore, they add similar functionality as attack jungles, while not allowing the tree to become a graph and thereby adding complexity.

4.2.3 Bottom-Up Values

The attack tree structure in itself can be very useful for vulnerability insights in a system. Though it might be helpful to also analyse the system using quantifiable methods. In order to do this, more attributes must be added to the attack tree than just the name and an node type. An important note is though, that the structure has to be finished before any values can properly be assigned to these attributes.

All nodes in the tree must have the same attributes, however their value can vary. These attributes can be anything that might be of interest for the analyst. Attributes that are used often are risk appetite, time available, budget available and skill level/chance of success. Especially budget available is an easy measure to compare: If the attacker has to spend more money than he will gain, the chances of the attack happening are small. E.g. if an attacker has to spend €2000 on tools to get into a vault, but there is only €500 in the vault he will probably go look for another target. An important requirement for this statement is that the attacker knows the value of what is inside the vault.

Each leaf is assigned a value for each chosen attribute. This value is then propagated up the tree so that each node gets its own value for all its attributes. In the first versions of attack trees, the way to propagate these values up the tree was by re-evaluating each node. Now though, this has been standardised for easy automation and most extensions to attack trees bring their own algorithms to do this. In most cases the value assignment of the leaves still has to be done manually and is completely reliant on expert knowledge. Therefore it remains one of the most challenging tasks in appraising attack trees.

When propagating values up the tree we can make a distinction between three types of attributes. These types and their appropriate propagations are discussed next. In each of the formulas given x is a node in the attack tree and the function $child(x, i)$ returns the ith child of x. The function $type(x)$ returns the type of a specified node, which could be of the type (S)AND, OR or LEAF.

Probabilities These can be propagated using probability theory. For OR-nodes the probability that any one of the child nodes is successful is needed. This is calculated by negating the chance that all child nodes are not successful. For AND- and SAND-nodes all child nodes need to be successful. When the node is a leaf node, the probability should be given.

This results the function $P(x)$ as specified in Formula 4.1 which returns the probability of success of a specified node.

$$P(x) = \begin{cases} Given & \text{if } type(x) = \text{LEAF} \\ 1 - \prod_{i=1}^{n}(1 - P(child(x,i))) & \text{if } type(x) = \text{OR} \\ \prod_{i=1}^{n} P(child(x,i)) & \text{if } type(x) = \text{(S)AND} \end{cases} \tag{4.1}$$

Costs The second type of attribute that can be assigned to nodes are simple costs. This can be monetary costs, cost of time, etc. These costs just add up for every action that is taken. An OR-node simply takes over the cost of its cheapest child node; it is assumed that the attacker will take the cheapest path. For AND- and SAND-nodes all child nodes have to be successful, therefore the cost of all child-nodes is summed.

This results the function $C(x)$ as specified in Formula 4.2 which returns the cost of a specified node.

$$C(x) = \begin{cases} Given & \text{if } type(x) = \text{LEAF} \\ \min_{i=1}^{n} C(child(x,i)) & \text{if } type(x) = \text{OR} \\ \sum_{i=1}^{n} C(child(x,i)) & \text{if } type(x) = \text{(S)AND} \end{cases} \tag{4.2}$$

Requirements This could, for example, be a minimum skill level or a minimum amount of people. These requirements indicate a minimum value of a certain attribute. Therefore it is easily propagated: OR-nodes take over the requirement of their cheapest child node, following the easiest path. AND- and SAND-nodes take over the requirement of their most expensive child node, because all of the child nodes need to be satisfied.

This results the function $R(x)$ as specified in Formula 4.3 which returns the value which is at least required for success of a specified node.

$$R(x) = \begin{cases} Given & \text{if } type(x) = \text{LEAF} \\ \min_{i=1}^{n} R(child(x,i)) & \text{if } type(x) = \text{OR} \\ \max_{i=1}^{n} R(child(x,i)) & \text{if } type(x) = \text{(S)AND} \end{cases} \tag{4.3}$$

These propagation formulas can only be used when subtrees are not reused. This is acceptable for the purpose of this thesis. If the need to add reusable subtrees arises, the methods in [20] by Kumar, Ruijters, and Stoelinga will be used.

With these semantics, a more precise insight can be gained into the weaknesses of the system. Furthermore, the weakest points can be calculated automatically, even if the tree has grown too large to reasonably comprehend.

Chapter 5

Methodology

5.1 Approach

The methodology of this thesis is based on the design science research methodology for information systems research introduced by Peffers, Tuunanen, Rothenberger, *et al.* [32]. The aim of their research is to offer a common accepted framework for research in the field of design science. They define the following six steps to be part of such a methodology:

1. Problem identification and motivation
2. Define the objectives for a solution
3. Design and development
4. Demonstration
5. Evaluation
6. Communication

These steps, and therefore this research, are devisable in three parts. The first part is finding out what can be used to define objectives and then defining the objectives. The second part is a repeatable process of designing the manual and demonstrating and evaluating it. Finally, the last part is writing the thesis and presenting it.

5.2 Defining the Standard

The first step is to define a model for attack trees. This model will be based on attack trees made by professional researchers in the field of attack trees. It has been assumed that generally, these researchers create the highest quality attack trees, because they often have studied the field of attack trees for a long time and have contributed in creating attack trees as they are now. Therefore the attack trees they have made in their papers will be analysed to determine which aspects and building blocks are important to create a well structured and clear attack tree.

To determine the which papers should be used for this analysis, several paper catalogues and attack tree databases will be searched for the most cited papers.

Naturally, it is difficult to define what will be searched for in these attack trees before actually analysing them. Though the search will start with the following list, more could be added after discoveries have been made:

- Node architecture: Can the attack tree be split in layers, either horizontal, vertical or otherwise?

- Node grouping: Are nodes grouped together in a way not caused by the normal splitting of nodes?
- Splits: Is there a specific way of splitting nodes at certain levels of the tree?
- Rate of abstraction: How much are nodes abstracted each step?
- Tree traversal: Are nodes created depth-first, breadth-first or a combination of those?
- Other practices: Which other practices are used to manage an attack tree in e.g. size or modularity?

Some of these aspects might require a little more explanation than the questions described above; **Node architecture** refers to making a distinction between certain layers of the tree where specific types of nodes are grouped together in one layer. For example there could be a layer containing only high level attacks which has a layer below it that only includes the means needed to execute those attacks.

Node grouping aims at nodes that are purposely put somewhere else, i.e. positioning of nodes that is not caused by the natural flow of the tree.

Splitting refers to intentionally splitting the nodes in certain sub-nodes. A good example would be if an attack is split between digital attacks and physical attacks. In the list above, as well as the rest of this thesis, the process of subdividing nodes that represent an action into children that represent the proper sub-actions will be referred to as splitting nodes.

Rate of abstraction is in how much detail the children of a node describe their actions. For example the node 'Open door' could be assigned a child node 'Use a hammer', however it could also be assigned the node 'Use force' which is much more abstract.

Tree traversal mainly affects the thought process when brainstorming for new nodes. It could be advisable to think of attacks layer by layer because they need a different type of thinking, or it could be more efficient to first work out one full attack and then the next.

Other practices refer to handy practices which do not necessarily change anything to the contents of an attack tree, but are more focused on 'managing' it or adding extra information to it. This could include things like how to display a very large attack tree one a page.

This analysis will primarily be conducted during the first phase of the research, however also during the designing of the model and manual, this analysis will continue. This will ensure that all aspects that can be found will be found and included in the final design.

5.3 Designing the Model and Manual

The aspects that are found will be properly defined in a model, and where it is not possible to put them in a model, they will be properly defined separately. This model would ideally form a meta-model for attack trees, which gives as much structure to attack trees as possible, while staying simple enough to be able to explain relatively easily. The more practices, aspects and building blocks are gathered, the better this model can be used as a general standard. The extensiveness of the model also determines the level of enforcement that can be used.

After the model has been defined, it will effectively pose as a help-objective for the manual. I.e. an important goal of the manual is to guide users towards creating attack trees that are in line with the designed model. The model acts as an extra step between the manual and its original goal of allowing experts to create proper attack trees within an acceptable time frame.

Of course, the model, as well as the manual are unlikely to be perfect on the first try. Therefore the designing process of both will continue throughout the research and it is expected that, especially after the evaluation, significant changes will be made. Big or important changes will be documented whereas smaller details will be just be added in the next iteration.

5.4 Evaluating the Manual

To improve upon the model, but mainly the manual, they will be evaluated. This will be done first by a series of usability tests. These usability tests will be continued until it can no longer be improved through the feedback of usability tests or until there is no time left available to do more usability tests.

During these usability tests, participants will be asked to create an attack tree of a system they know. Ideally, half of them would do it with a manual and the other half would do it without any help. However, it is expected that participants are not willing to spend the required time and effort to read up on attack trees completely by themselves. Moreover, this is likely to cause a very wide range of attack tree types which are barely comparable.

Instead, the participants that would otherwise have worked without a manual now get a very simple manual. This simple manual will guide them in the correct direction and tell them the purpose of an attack tree being made. This way they know what the goal is for them and what they should be working towards.

After the participants have created their attack tree, a series of questions will be asked to determine what they thought of the usability and the helpfulness of the manual. Thereafter, they will be asked to compare attack trees made by other participants to determine whether the full manual improves the quality of the resulting attack tree.

To deal with the time constraint of the goal of this research, all participants are given limited time to complete their attack tree. How much time they are given will be determined by a quick pre-evaluation test.

5.5 Case Study

To properly demonstrate that the manual helps with quickly creating well structured attack trees, a case study will be done. This case study will be conducted at a company; a connection will be provided by Nedap N.V. During this case study, the manual will be presented instead of just provided to read. Then, the manual will be followed to create an attack tree that is a correct representation of the risks of the company. As with the rest of this thesis, the focus during this case study will be on the structure of the attack tree and discovering new attacks. Adding values to the nodes is an optional addition.

The case study is expected to come with a level of anonymity as some rather sensitive data is revealed during this case study. This can hopefully be solved by anonymising the company so that the full attack tree will be visible and does not have to be compromised in order to be published.

Part II

Design

After describing what the goal is and why, we proceed with analysing attack trees from previous literature. This gives insight as to what makes an attack tree a good one. This information could then be used to design a model as well as a manual to communicate that model and explain how to use it. This resulted in an initial version of the manual.

Chapter 6

Literature Analysis

In this chapter, the initial research towards quality defining aspects is documented. This initial research is gathered from previous literature and will help answering research questions R.1 and R.2.

Other researchers have worked with attack trees, some of them since the first notions of attack trees. They have followed the developments made in this field and gained experience accordingly. Intentional or not, each of them will have developed their own way of creating attack trees.

To find out how attack trees are commonly build, we analysed 60 attack trees sourced from the papers they wrote. The aim is to find a general standard for this field of research. Therefore Google scholar, Scopus and Science Direct were searched for the terms ’attack tree’, ’attack trees’, ’attack tree case study’ and ’attack trees case study’. Where possible the results were prioritised by reference count. Furthermore papers were taken from the review paper from [1] and from the library of attack trees created by the SaToSS group [29] of the university of Luxembourg. In this search for attack trees, papers without actually describing an attack tree were skipped. Also small trivial examples were not taken into account. If more than one attack tree was described in a paper the most advanced one was picked. This resulted in a set of attack trees sourced from [2], [4], [12], [13], [15], [17], [20], [30], [33]–[84].

At first, mainly the aspects mentioned in Section 5 were investigated: tree architecture, node grouping, node splitting and rate of abstraction. The aspect pruning and tree traversal were left for the reflection and interviews as that information is very difficult to obtain from only a tree, if not impossible. Of course, more patterns might arise. When one does it is documented and all previously analysed attack trees are reiterated to check for that pattern.

All of these patterns and how much they were used are described below and a full table of which attack tree contains which features is included in Appendix A

6.1 GAP Layering

Immediately from the start of this analysis six layers could be identified. These layers were *goals*, *sub-goals*, *attack types*, *attacks*, *process* and *choices*.

With these six layers most attack trees can be formed and definitely all attack trees that were analysed can be. Sometimes layers can be left out when they are irrelevant for a certain case and the attack tree is relatively small. To easily refer to this layering later in this thesis, it will be referred to as *GAP* (Goal, Attack, Process) layering. In the GAP layering the user starts at a goal, then explores its options regarding attacks eventually resulting in a layer with attacks. Finally it focusses on how those attacks should be executed, resulting in the process layer. Layers have often proven to be only one node deep, however this is not always the case. The layers of the GAP layering are displayed in Figure 6.1. The usage of these layers is discussed below and the corresponding percentages are displayed in Figure 6.2.

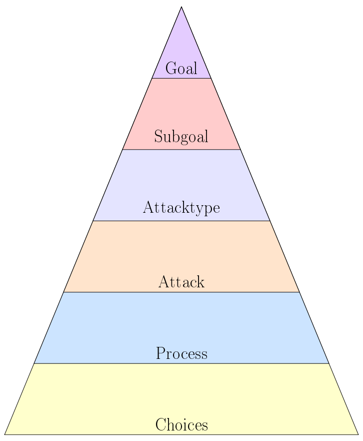


FIGURE 6.1: The GAP layering

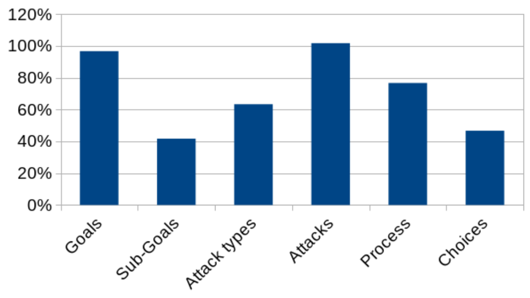


FIGURE 6.2: Percentage of attack trees that use a certain layer or layering. N=60.

6.1.1 Goals Layer

95 percent of attack trees have a goals layer. The goals layer consists of the main goal which always at the top of the tree. The most informative thing about this statistic is that it means that 5 percent of the attack trees analysed do not have a goal. This suggests that these attack trees are aimed at an attack, instead of a goal. It would be a possibility that they are aimed at attack types, explained hereafter, but in the case of these three attack trees they are not. So, in some cases, attack trees are also used to analyse a single attack, without any specific goal in mind.

6.1.2 Sub-goals Layer

40 percent of attack trees has a sub-goals layer. Introducing a sub-goals layer seems to be an obvious step when an attack tree gets larger, because it organizes the rest of the attack tree below it. For now this is important to keep in mind, because it indicates that things might be done differently (e.g. more layers) if attack trees grow large enough. The sub-goals layer generally consists of certain different parts of a system that could be targeted.

6.1.3 Attack Types Layer

62 percent of attack trees have an attack types layer. In this layer, the attacks are grouped by type. This layer is also where most of the common splits discussed later are placed. Whether an attack type layer is included in the attack tree or not is most dependent on the size of the attack tree and the number of attacks: A larger attack tree has a larger need of organisation to remain clear.

The line between the sub-goals layer and the attack types layer might seem vague, because they both organise the subtrees below them. However there is also a clear distinction. The most important difference between sub-goals and attack types is that sub-goals often specify what to attack and attack types obviously specify different types of attacks for each of those things to attack.

6.1.4 Attacks Layer

100 percent of attack trees have an attacks layer. It seems obvious that this is the most important layer of an attack tree. Whether it is used for a high level analysis of possible attacks or whether it is used to analyse a single attack, the attack layer is always included. Although they are always included in this test set, it is difficult to tell exactly when something is defined as an attack when it is in the lowest layer of the tree: Depending on the creator it could be meant as a category with further splits still to be made.

6.1.5 Process Layer

72 percent of attack trees have a process layer. This layer comes after the attack layer and describes what has to be done for the attack to be executed. It is often easily detected in an attack tree, mainly because there is a switch from largely OR nodes to largely AND and SAND (Serial AND) nodes. I.e. instead of considering options the analyst is now determining what has to be done to execute an attack. Because it describes what has to be done once more (previously in the goals layer) but now on a lower level, this layer is also the layer where subtrees might be started; That will be discussed further in Section 6.5.2. Finally, this layer can grow rather large in comparison to other layers when an attack tree is built without SAND nodes, because some times nested AND nodes are used to simulate a SAND node.

6.1.6 Choices Layer

45 percent of attack trees have a choices layer. This layer comes after the process layer. After being described what has to be done, this layer describes how that could be done one last time. Therefore it also clearly turns back from AND and SAND nodes to OR nodes. In all cases where this layer was included, it was also the last layer. Most often there is no reason to believe that there could be another layer, however there are attack trees where this layer gets rather large. The attack trees where this happened were focussed on software.

6.1.7 Relation to Size

Besides discovering what kind of layering is used, it is also of interest *when* which layers are used. With that information advise can be given on when to use certain layers. In Figure 6.3 the layers are listed with the average size of the attack trees that uses that layer.

As can be seen, most layers do not deviate much from the average. Mainly attack trees that have an attack types or choices layer are larger on average. This is in line with expectations as attack types are organisational and become obsolete when the number of attacks gets too small. The choices layer less so. However a reason that attack trees with a choices layer are larger on average could also be simply because they have that layer. The lower in a tree, the more nodes a layer has. Therefore just the addition of this layer could cause the size increase.

Attack trees with sub-goals are slightly larger on average, again in line with expectations. For small systems it is of little use to split into sub-goals or sub systems.

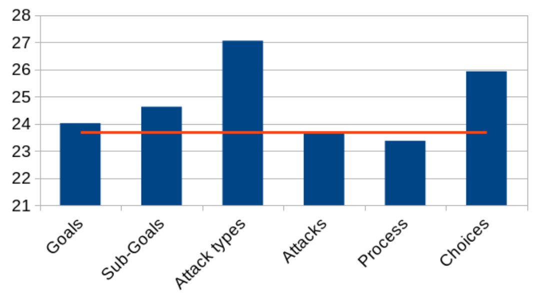


FIGURE 6.3: Average size of attack trees that use a certain layer. N=60, the total average size is 23.7 and is indicated by the horizontal line.

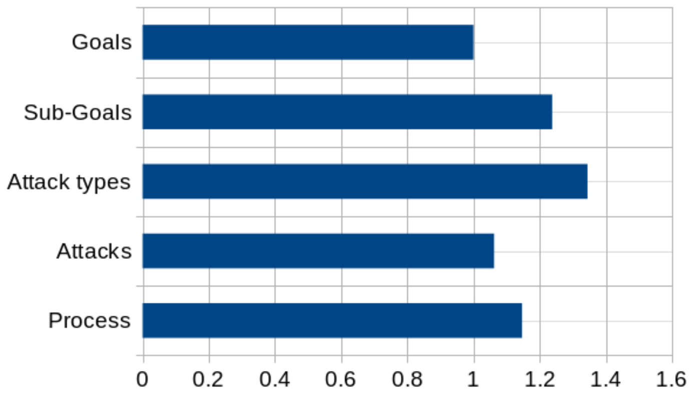


FIGURE 6.4: The average size per layer of the GAP layering. N=60.

Finally, this analysis also includes the size per layer of the GAP layering. These numbers are displayed in Figure 6.4. Only layers that are actually used are used in the count, so the minimum number of rows per layer is 1. The average number of rows per layer over all layers is 1.11.

6.1.8 Node Types

Finally, the attack trees were analysed to discover whether layers in the GAP layering mainly use (S)AND nodes or mainly OR nodes. For each occurrence of a layer, the type of split above and below it were noted. The type is checked bottom-up and top-down, even though the attack tree will be made top-down, to increase the chances of finding a pattern. In Figure 6.5 and Figure 6.6 the distribution of the two types is displayed per layer.

In both figures there is a rather clear preference for either type on every layer except for the type above the sub-goals. In first instance it, the split below a layer is most important, because the attack tree will be built top-down. The splits below the layers indicate an initial type; Attacks often split with (S)AND nodes while the other layers most often split with OR nodes. For the goals and sub-goals layers there is still a realistic possibility of a (S)AND node too.

From the types above the layers mostly the same thing can be seen, except naturally the bars have shifted by one layer. However, the one layer that stands out is the sub-goals layer. This layer was previously discussed as often separating the system in sub-systems. The fact that the type above sub-goals is divided almost 50/50 shows that these sub-systems could be either optional, or all required.

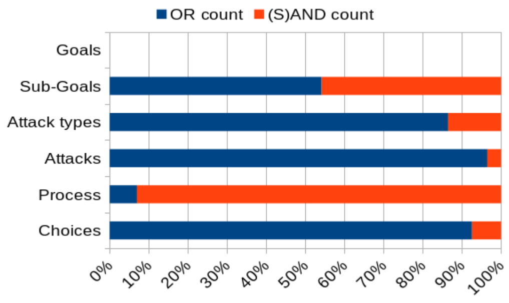


FIGURE 6.5: Distribution between (S)AND and OR nodes per layer. The split *above* the layer is documented. N=60.

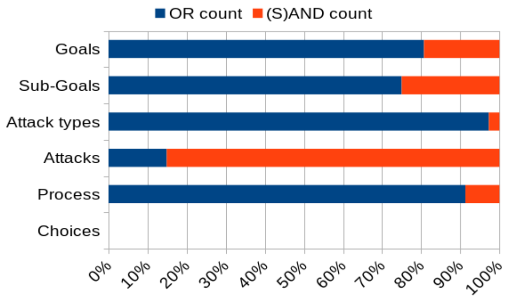


FIGURE 6.6: Distribution between (S)AND and OR nodes per layer. The split *below* the layer is documented. N=60.

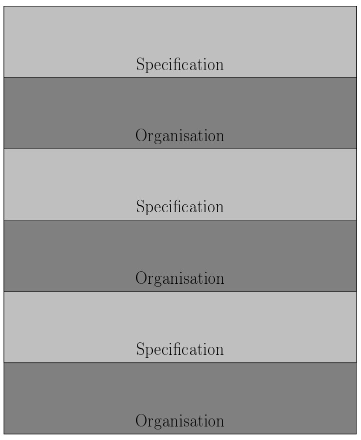


FIGURE 6.7: The SO layering

6.2 SO Layering

The *SO* (Specification/Organisation) layering is one of alternating layers of specification and organisation as displayed in Figure 6.7, or more specifically: alternating rows of (S)AND and OR nodes. At the root of a tree there is a specification of what is supposed to happen. Then there is a layer that organises the subtrees below into categories and then another specification of what is supposed to happen. Important to note here is that although it is called an organisational layer, it is not the case that they can be left out. In the remainder of this thesis this layering will be referred to as SO layering. 27 percent of attack trees fit in this type of layering.

6.2.1 Relation to Size

Also for this layering type, the size of attack trees that use it is analysed. Attack trees with the SO layering are slightly smaller on average. The total average size is 23.7 nodes while the average size of attack trees that comply with the SO layering is 22.13 nodes. The use of the SO layering was not expected to have a relation to size.

A possible cause for this could be that small attack trees have fewer rows and therefore more easily fit into the SO layering. For example: an attack tree with three rows only needs one layer with mostly (S)ANDs to fit.

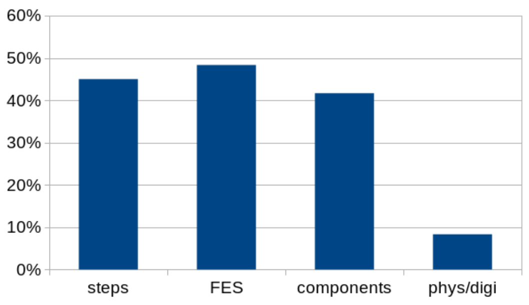


FIGURE 6.8: Percentage of attack trees that use a certain split. N=60.

6.3 Splits

Besides layering, the way certainnodes are split into sub-nodes is also investigated. These splits were mostly used as intuitive organisation so that the attention of the analyst is focussed in a particular direction. As described before, the list of expected splits is the following:

- A split between digital force, physical force, exploiting system flaws and social engineering
- A split between system components
- A split between physical and digital approaches
- A split that splits an attack or goal into a step by step plan

No recurring splits were found besides the ones that were expected.

Each of these splits and what percentage of attack trees used them are discussed below. The results are also shown in Figure 6.8.

6.3.1 FES Split

48 percent of attack trees use a split between digital force, physical force, exploits and social engineering. This split will be referred to as the FES split (Force, Exploit, Social engineering). This split is mostly used in the attack types layer to split a goal into the previously mentioned four types of attacks. Another option is that this split is used in the choices layer to give a standard grouping of options. Naturally, this is always a split of an OR node.

Most attacks fit in to at least one of these types. However that does not mean that all types are necessarily filled for every attack tree.

Even though only 29 out of 60 attack trees use this split explicitly, more attack trees *could* be split with FES split as they have its components. However, they are located in sub trees that are further apart. In other words; some attack trees have the force, exploits and social engineering somewhere in the attack tree, but not under the same node. Others have even effectively made this split, but named it differently. Because of this, there is a high possibility that more trees can be made to use the FES split if the intention is there.

6.3.2 Components Split

42 percent of attack trees use a split between components of the system. This is often an easy to understand grouping of subtrees when a system consists of multiple components that can be separately attacked. It can be used across the whole tree as long as there is

a system which has semi-separate components. However the components split is mostly used in the sub-goals and attack types layers. The likely cause of this is that systems that are large enough to be split into components are most often high up in the tree.

This split is simple and rather obvious when useable. Partially because of that most trees that do not split on components already will not be able to either. This is because as opposed to the FES split, it is not an abstract categorisation, but the system actually has to consist of those semi-separate components.

6.3.3 Steps Split

45 percent of attack trees use a split that splits a goal or attack into a step by step plan of action. In some cases this is done with the use of a SAND node, in other cases by nesting of AND nodes to simulate the same. This step-by-step split is very intuitive to use and to analyse. However, it does go against some earlier notions of how an attack tree was supposed to work. Some papers suggest that when an attack tree is constructed, attack scenarios should be acquired by starting at a leaf and working up to the root; so vertical attack scenarios. When this step-by-step split is used, the attack scenario essentially shifts from vertical to horizontal. Although it opposes earlier notions of attack scenarios in attack trees, it does synchronise with the layering seen in the previous subsection.

6.3.4 Physical/Digital Split

8 percent of attack trees use a split between physical and digital attacks. This split was expected to occur more often, however it has the same function as the FES split, which appearance is favourable. Even though there is not an explicit split between physical and digital attacks, it is more often than not easy to fit in. Though most analysts seem to choose to not add the extra node. This might be different when attack trees become larger, however that is just speculation.

6.3.5 Relation to Size

As with the layering, it is also important to know when to use certain splits. In Figure 6.9 the splits are listed with the average size of the attack trees that use that split.

From the figure, it is immediately clear that attack trees that contain a split between physical and digital attacks are generally much larger than the average attack tree size. This makes sense, because this is an organisational split and besides that, it is a very abstract split. This means that in order to be able to make such a split, the attack tree has to contain a rather large range of attacks and less abstract attack types already.

The attack trees that make use of the other three splits are generally smaller than the average. However their averages are close together and on further inspection they are almost exactly average when the attack trees with a split between physical and digital attacks are excluded. Therefore no solid conclusions can be made for a relation between these splits and the size of the attack trees in which they are used.

6.4 Grouping

Besides the logical structure, the way of displaying nodes has also been taken into account. From the 60 attack trees that were analysed, 82 percent of attack trees were graphically displayed. Of those attack trees, none showed any signs of node grouping besides the grouping that happens automatically by doing the right splits. Grouping nodes differently than how they come out by just splitting nodes is a trade off: The attack tree might become

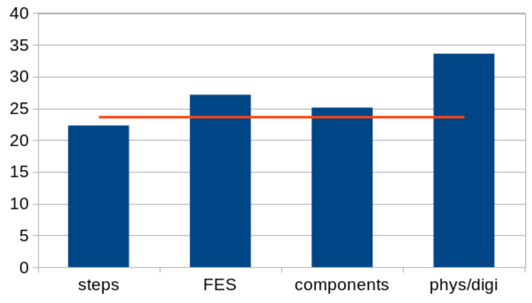


FIGURE 6.9: Average size of attack trees that use a certain split. N=60, the total average size is 23.7 and is indicated by the horizontal line.

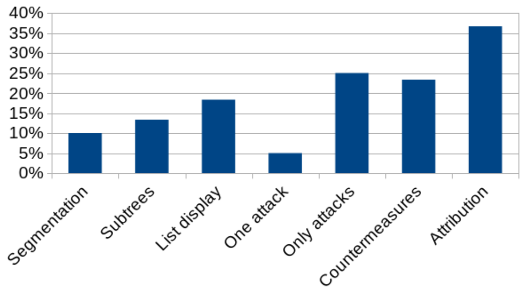


FIGURE 6.10: Percentage of attack trees that use a certain practice. N=60.

more clear when looking at just the nodes, but the lines become a tangled and less helpful. This might be useful when nodes are grouped for people that only need a certain group of nodes, but apparently the benefit does not outweigh the cost.

6.5 Other practices

Finally, some general practices have been investigated. These are relatively simple practices which often solve a practical problem. The practices that were checked for are the following: segmentation, focus on countermeasures, use of attributes, displaying the tree as a list, focusing on only one attack and the use of subtrees. What these practices entail and what percentage of attack trees use these practices is discussed below and the numbers are also displayed in Figure 6.10.

6.5.1 Segmentation

10 percent of attack trees use segmentation. Segmentation has multiple advantages, mainly for large attack trees. For one, huge attack trees can be divided into segments so that the main tree contains only the high level goals and perhaps attack types. Then each of those leafs can represent a segment that is placed elsewhere. This can be useful for displaying large attack trees in a paper. Furthermore, when done right, separate segments can be made by the appropriate system experts and combined later. This allows for unobstructed parallel work.

It is clear that the advantages of segmentation are not as great for smaller attack trees. However they can still be used there. Again for parallel work or just for making clear that that subtree is very much its own attack tree.

In comparison to subtrees described in Section 6.5.2, segments do not necessarily start at the begin of the GAP layering. Moreover, the defining part of a segment is that it is replaced by a segmentation node and displayed elsewhere whereas subtrees are mainly defined by the GAP layering restarting.

6.5.2 Subtrees

13 percent of attack trees use subtrees. Of course, every attack tree has subtrees, however some attack trees have specific subtrees that start again at the beginning of the GAP layering. These subtrees often start in the process layer when one of the child nodes is not yet (close to) trivial. These subtrees then complete a full GAP layering similar to the original tree did. Additionally, such subtrees often seem to be used for only one or two of the process nodes, never for all of them.

6.5.3 List display

18 percent of attack trees were displayed as a list rather than as a graph. In such a list each node has a number sequence which indicates its position in the attack tree. This number sequence is of the form $x.y.z$, where x is the number sequence of the parent and the node is child number y.

The main advantage of displaying the attack tree as a list instead of a graph is information density. A table has a much higher information density compared to a graph, especially in the case of trees. Therefore, when page size is limited and the attack tree is large, displaying it as a list might be the only option. Particularly wide attack trees quickly reach the limits of a page. Additionally, when each node has attributes assigned to it, it is easier to display these in a table, therefore they often go hand in hand. Again, the information density makes it so that the attack tree can be much larger.

6.5.4 One attack

5 percent of attack trees are focussed on only one attack. These attack trees have no goal specified and only describe the execution of a single attack, or in other words; The goal is the execution of a specific attack. In the case of these attack trees, this also means that there is no attacker that is a limiting factor or wants a specific thing to steer the attack in a certain direction. Therefore these attack trees can get rather large, because they specify in great detail the accomplishment of the attack.

6.5.5 Only attacks

25 percent of attack trees are focussed on just attacks. These attack trees have a goal, maybe some sub-goals and attack types and then the attacks. However the attack tree stops at the attack; there is no description of how to execute those attacks. This is likely where the attacks becomes trivial for the analyst, which means in principle the attack tree has served its purpose for him. This might make it incomplete for others when they need to work with the attack tree. It is however a good practice when just a global view of the situation is required. It also prevents the tree from growing too large, because the number of nodes in a tree can grow exponentially relative to the depth of the tree as there is no limit to the number of children per node.

6.5.6 Countermeasures

23 percent of attack trees are focussed on countermeasures. These are technically attack-countermeasure trees or defence trees. Rather than just focussing on the attacks, they also immediately include countermeasures in the tree. This helps determining where

countermeasures still need to be taken. Moreover, these countermeasures themselves can also be attacked, therefore a countermeasure is not necessarily a leaf. Arguably this statistic could easily change in either the real world or when the research area is changed to e.g. defence trees.

6.5.7 Attribution

37 percent of attack trees have attributes assigned to the nodes. These attributes help calculate certain odds, costs, etc. as discussed Section 3 and Section 4. The motivation for including these attributes are rather obvious: there is an interest to quantify the risks involved.

No correlation was found between attribution and either layers or splits. However, there did seem to be a correlation between attribution and the list display practice. This makes sense, because adding multiple values to a node in a visual display quickly makes it crowded. It is more organised to make the tree into a table with nodes' number, name and attributes on one row.

6.5.8 Relation to Size

Once more, it is important to know when to use certain practices. In Figure 6.11 the practices are listed with the average size of the attack trees that use that practice.

To start at the top: attribution seems mainly to be used in smaller attack trees. As long as the attack tree is created manually this makes sense, because for huge trees, the calculation of the attributes becomes exponentially more work.

Attack trees that focus on attacks and therefore stop at the attacks layer are about average in size. Moreover, they go over the average a little. One would expect that such an attack tree would be smaller because of the missing layers. Clearly this is not the case.

Another unexpected result is that attack trees that use list displays are slightly below average in size. The list display was expected to be a means to an end when the attack tree gets too large and needs to fit on a page.

In the same category, a more expected result; segmenting is mostly used in significantly larger attack trees, nearly reaching 50 percent larger attack trees than the average size. Segmenting might not only be a way to be able to display large attack trees, but it also causes large attack trees simply because it allows them to be large without the bottom layers being obviously trivial compared to the top layer.

The practice of designing an attack tree for only one attack and with no goals in mind generally produces smaller attack trees than the average. On average they are roughly 50 percent smaller than other attack trees. This is no surprise as not only is there no goal, there are also no other attacks to expand on.

Finally, adding countermeasures obviously increases the size of the attack tree. It does not change much to the attack tree itself except adding a number of defensive nodes, making the attack tree larger.

6.6 Relations and Positioning

In order to combine these models into one manual for attack trees, relations between these models are needed. The main focus lies on relations between the GAP layering and the splits. The SO layering is barely used and does not add much structure besides the structure already provided by the GAP layering. The practices are still important, but more useful when kept as a solution to specific problems instead of including them in the model.

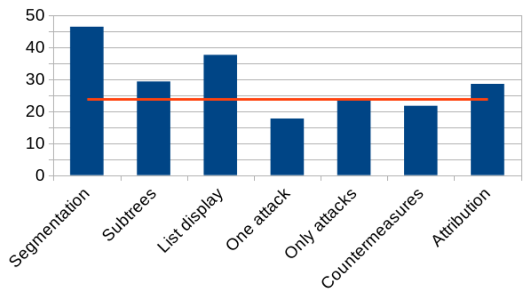


FIGURE 6.11: Average size of attack trees that use a certain layer. N=60, the total average size is 23.7 and is indicated by the horizontal line.

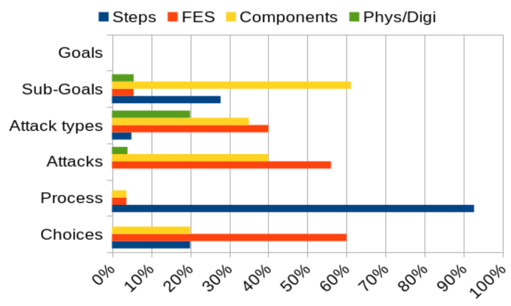


FIGURE 6.12: The location of a split is determined by its resulting nodes.

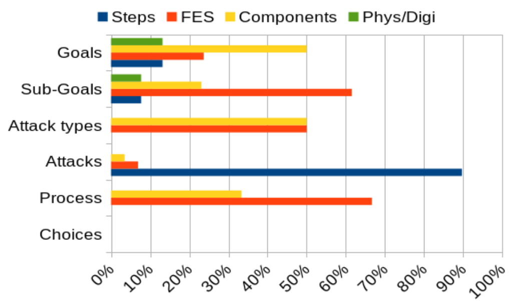


FIGURE 6.13: The location of a split is determined by its origin node.

FIGURE 6.14: Distribution of splits per layer to a total of 100% per layer. N=60.

Therefore the attack trees were analysed more thoroughly. This time for each split it was noted in which layer it occurred. A difference was made in how the location of a split is defined. The first basically anchors the split by the node that starts it (origin node), the second one anchors the split by the resulting nodes. This was done to increase the possibility that a relation was found, be it in the origin nodes or in the resulting nodes.

This analysis produced four distributions; The distribution of layers per split based on the origin node (Figure 6.13) and based on the resulting nodes (Figure 6.12), and the distribution of splits per layer based on the origin node (Figure 6.16) and based on the resulting nodes (Figure 6.15). In these distributions it is important to remember that not every split that is done in an attack tree is one of the splits that were analysed. This means that if, for example, a layer is indicated to consist completely of one type of split in the figures, other splits than the ones analysed could still be used too.

These figures provide a solid basis for a model for attack trees. Especially in the lower layers of the tree there is a clear line in the relations. The upper layers provide more options for which split is done when. This is likely due to the fact that when an upper layer is skipped the tree still continues to other layers. This provides more room for relations to other layers besides just the next layer. If one of the lower layers is skipped the tree is likely to end there.

In Table 6.1 findings are listed that could be important for the model that will be designed in Chapter 7. The goal of these findings is to clearly list what kind of splits a certain layer starts and what layer comes after that split. Furthermore low percentages are often disregarded as the aim of this thesis is to design a standard model which every attack tree can be made to fit into, not to include all options that are used now.

Finding	Layer	Split
A goals layer often starts a components split and most component splits get started in the goals layer. This components split can also be a steps split	Goals	Components; steps
A goals layer some times starts a FES or phys/digi split. Most phys/digi splits start in the goals layer and the goals layer is one of three layers a FES split starts in	Goals	FES; phys/digi
A sub-goals layer is most often the result of a components split which sometimes also is a steps split.	Sub-goals	Components; steps
A sub-goals layer can start a FES split or a components split.	Sub-goals	FES; components
An attack types layer often consists of the result of a FES split or components split	Attack types	FES; components
An attack types layer some times is the result of a phys/digi split.	Attack types	phys/digi
An attack types layer always starts a FES split or a components split.	Attack types	FES; components
An attacks layer is almost always the result of a FES split or a components split.	Attacks	FES; components
An attacks layer almost always starts a steps split.	Attacks	Steps
A process layer is almost always the result of a steps split.	Process	Steps
A process layer always starts a components split or a FES split, however neither of those occur often in the process layer.	Process	Components; FES
A choices layer is mostly started by a FES split, however very little of the FES splits result in a choices layer.	Choices	FES

TABLE 6.1: List of findings regarding the relations between splits and layers

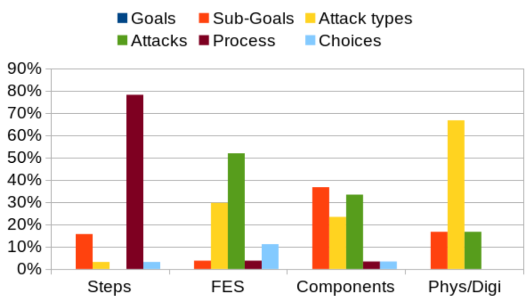


FIGURE 6.15: The location of a split is determined by its resulting nodes.

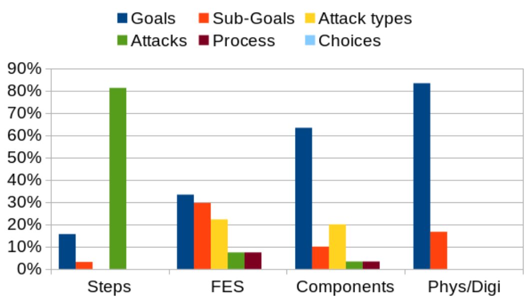


FIGURE 6.16: The location of a split is determined by its origin node.

FIGURE 6.17: Distribution of layers per split to a total of 100% per split. N=60.

Chapter 7

Design

In this chapter the design for the design choices for the manual will be made. The most important part of this is designing a model for attack trees. Around that model the manual has been created.

To achieve this, before creating the model and manual, a series of design choices has been made and decided on based on previous findings. Thereafter the model and manual were made according to those choices.

7.1 Visualisation

First of all a way to visualise the model is needed. This is not only important for the user of the model, but also to be consistent within this thesis. Mainly the layers and the splits need to be clearly identifiable.

For the layers there are four options: create bars across the attack tree with the name on the side; add an icon to each node; make nodes that belong to the same layer a certain shape; give the nodes different colors. The last option immediately fails for colour-blind people or when the paper is printed in black and white. Furthermore, different shapes for all layers and splits will become confusing. Therefore we choose to use small icons in each node that identify to what layer it belongs. To add clarity, nodes of the same layer will also be displayed on the same level, however there will be no name on the side.

For the splits there are three options; the same as the ones for the layers except the one where the name is on the side. For the same reasons, but his time also for consistency, we choose to use icons in the middle of a split to indicate what type of split it is. Besides icons for the splits that were analysed, the table also includes an icon for custom splits so they can be added when necessary.

In Table 7.1 all layers and splits are listed with the icon that represents them below.

Finally, whether a node is a SAND (Serial AND), AND or OR node is displayed as is done in the rest of the thesis; With an extra line below the node for an AND node and an arrow for a SAND node.

7.2 Which Layering Type

As became clear in Section 6.1 and Section 6.2, the GAP (Goal, Attack, Process) layering is rather widely used whereas the SO (Specification/Organisation) layering is used in only a quarter of the attack trees. Furthermore, the way both layer types are defined, the GAP

Element	Goals	Sub-goals	Attack types	Attacks	Process	Choices	Steps	FES	Components	Phys/digi	Custom
Icon	◎	◎	📦	🗡	⚙	?	👣	⚗	▦	🖥	✎

Table 7.1: Table of which icon identifies which split or layer

layering incorporates much of the SO layering; The GAP layering is less strict on the alternating AND and OR nodes.

That said, it is obvious that the SO layering will not be included in the model. The GAP layering however will be used. Especially because of its solid relations with the splits that were found.

7.3 Layer Size

The number of rows a layer can consist of is important for the size of the attack tree. Moreover, it is also important for each layer as fewer rows in a layer mean less room to add certain splits.

In most of the attack trees analysed only one row per layer is used; 91% of all GAP layers consist of only one row. Furthermore, the average number of rows per layer is 1.11 according to Section 6.1.7.

Some times a layer is only used in one branch of the attack tree and not in the others. Finally, in some cases there are layers of multiple rows. This mostly happens in the process layer when AND nodes are being stacked and in the choices layer when the lowest level options are plentiful.

Ideally, each layer consists of one row. This will keep the attack trees that come from the model relatively compact. Furthermore the analysis proves that one layer of each is plenty for most attack trees. This limit could provide a useful manner to outline the abstraction per layer.

As for the option to not use a layer or use a certain layer only in a portion of the branches of the attack tree: This will be allowed, because it is not useful to force an attack tree to be larger than it has to be.

7.4 Layer and Splits Relations

In Section 6.6 the goal was already to phrase the relations in such a way that the relations between layers and splits became rather definitive. Also, most weak relations (low percentages) are already left out. Therefore the step towards actual relations is small.

The thing that is most important to realise is that the relations that were defined assume that the splits that were analysed are the only splits that exist. However this is not the case. Therefore, for weak relations, an increased possibility for a custom split should be included.

From Table 6.1 a new table, Table 7.2, can be formed, which lists all relations from layer to layer with a certain split in between. In this table, custom splits are also added to complement low usage of analysed splits. If the icon of a split is between brackets it is used less. Furthermore a '|' means it is either one or the other type of split. Finaly, a '+' means that a split is both types.

As is clear from the table, the number of relations has been narrowed down quite far. There is one main line of relations running trough each of the layers in the intended order. Besides those there are only three relations that can be used to skip layers.

7.5 Starting Point

Every attack tree starts with a goal. It is important that the user knows that this goal should be one of a malicious attacker. Furthermore, the goal of creating an attack tree in the first place is to discover possible attacks on the system. Both of these goals should be clear from the manual.

Start layer	Split	End layer
Goals	Components (+Steps)	Sub-goals
Goals	FES | Phys/digi	Attack types
Goals	(Components)	Attacks
Sub-goals	FES (|Phys/digi)	Attack types
Sub-goals	FES	Attacks
Attack types	FES | Custom	Attacks
Attacks	Steps	Process
Steps	Custom	Choices

Table 7.2: List of findings regarding the relations between splits and layers

Furthermore, it will be specified that the goal can be very broad as to not limit the discovery of attacks by a goal that is too specific. Therefore, the goal does not have to be SMART (Specific, Measurable, Attainable, Relevant and Time bound) [85]. Especially measurable and attainable are too limiting for a useful attack tree.

7.6 Intuitive Representation

To complement the model, an intuitive description of the model will be added to create a manual. So far the focus has been on the the model itself.

During the analysis it was found that the GAP layering seems to have two stages. The first stage is the discovery stage. This stage spans over the goals, sub-goals and attack types layers. As the name suggests, in this stage the focus is on discovering ways to attack. Anything is possible here, the aim is to broaden the view.

The second stage is the detailing stage which consists of the attacks, process and choices layer. At the beginning of this phase there is a set of attacks available from the first stage. In this stage each attack will be explored in further detail.

To summarise: the discovery stage is the ’what’ while the detailing stage is the ’how’.

A more layer by layer approach would be as follows: The goal is a high level goal and can be very broad or very specific. The sub-goals split this goal in pieces. This is often done when the target system can be split in multiple components. The attack types categorise the attacks in types, there are standard splits available, however other splits are also allowed. Next, there are the attacks. These are the actual attacks that can be executed. These are followed by the process which describes (often in steps) which things have to be done to execute such an attack successfully. Finally, there can be choices. These choices are about certain options when a step in the process can be done one way or the other.

Clearly, the names of the layers are already aimed to represent this intuitive description.

7.7 Process

The aim of this thesis is to design a model and a manual that are intuitive to use. Therefore it is obvious that the process of creating an attack tree using this manual will follow closely from its intuitive representation defined in Section 7.6. Therefore the process will follow the discovery and detailing stages that were mentioned there to globally organise the attack tree. Furthermore, the more specific description of the layers will give more structure to the attack tree and will be used to determine what goes where. The latter is important to be known beforehand so that the process has as little stops as possible.

The first step is to set a goal of course. How this should be done is described in Section 7.5

Then, the discovery stage begins which can be done as a brainstorm session. Ideally this is done in a group, but can also be done alone. The aim of this stage is to discover as many attacks as possible. Sub-goals, attack types and attacks can all be added. During the brainstorm, the user should try to place ideas in the correct place already, this can help by being able to trace ideas up or down the attack tree. However, they should not spend too much time on position right now, it is more important to keep the ideas flowing.

At the end of the brainstorm, the ideas should be correctly positioned where that is not already done and additional nodes can be added to fill in the gaps.

When the attacks are discovered and positioned, it is time to switch to the detailing stage. Here, each attack will be examined individually to determine what steps have to be taken in order to execute this attack. This requires a lot of expert knowledge and often not the knowledge from all the experts. Therefore it is a good idea to divide the attacks according to who knows most about a certain attack.

7.8 Use of Practices

As became clear in Section 6.5, the practices that were analysed where not used very frequently. Attribution was most used; in 40% of the attack trees. Partially because of this, but also because each practice specifically solves a problem, the decision was made to not directly include these in the model. However, they will not be completely disregarded either.

As said, these practices solve a specific problem. Therefore they will be mentioned in the manual as a solution to their respective problems. This will cause them to not interfere with the creation of the attack tree, while still profiting from their usefulness.

7.9 Model Strictness

Now that most decisions have been made regarding the model and the manual, it is important to decide how important this model actually is when creating an attack tree. On one hand, the aim of the model is to standardise attack trees and their creation. On the other hand, it also aims to support the user in the process of creating an attack tree for a certain system.

To standardise attack trees it would be helpful to strictly follow the model and allow nothing else. However in the end, the most important part of an attack tree is that the system is modelled correctly and that as many threats as possible have been identified. The model must not be a hindrance and can not be the cause of threats being left out.

To add to this, at the moment there are only four commonly used splits. Until more such splits are added, the user should not be forced to only use predefined splits. At this moment, the user will have to add custom splits in the attack tree.

To summarise; Following the model as strictly as possible is encouraged. Especially for the layers this is expected to provide ample opportunity to create a good attack tree. However if the model falls short in options such as the splits, it is acceptable to customise the attack tree. Still, for future work the aim is to add more features to this model so that attack trees can be created by strictly following the model.

7.10 Result

The previous decisions result in the model displayed in Figure 7.1. In this model each node represents a layer from an attack tree. The arrows indicate transitions between layers using specified splits. All nodes and relations are identified by icons. Some arrows have multiple icons, this means that both could be used or a combination of the two.

After the very first version of the manual was made, it was discussed with several potential users. This did not include creating an attack tree with the manual yet, only a first look review and understandability check.

From this, several things were found that were unclear. The most important factor was that there were too many icons, which caused a steeper learning curve instead of making it easier. Therefore, only layers are indicated by icons and both layers and splits are indicated by name. Furthermore, there were mainly minor changes such as rewording certain parts, rotating the model and improving the visual design.

The resulting first version of the manual, which includes this model, as well as a guide on how basic attack trees are created and the formalisation found in this thesis, is included in Appendix B. The manual is divided in numbered blocks which each shortly discuss a part of the process. This leads to a clear manual that concisely describes each part.

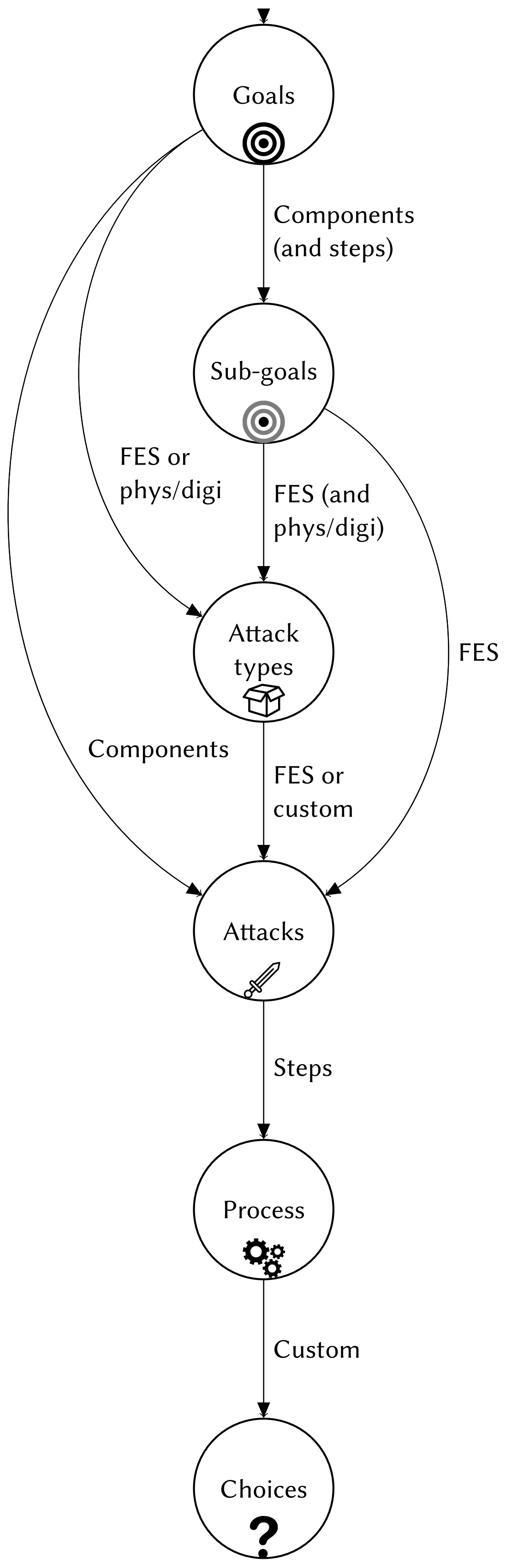


FIGURE 7.1: Model for attack trees

Part III

Evaluation and Refinement

After the first version of the manual, which was based on previous literature, had been designed we proceeded by testing the performance of the manual and improved it where necessary. This was done in two parts. First a quantitative usability test was done. This provided the feedback necessary to perfect the manual. After the model could no longer be improved this way, it was used in a larger, qualitative case study. This case study was conducted at a company affiliated with Nedap N.V. and gave us a final look at how well it works.

Chapter 8

Evaluation - Quantitative

In this chapter, the manual for attack trees, including the model that should define a proper structure for attack trees, is evaluated. This is done by usability tests described in this chapter. After 12 participants, the manual and setup of the test changed slightly. Thereafter, 22 more participants did the test for a total of 34 participants.

Because of this, the results are split in two parts. The first part contains the results of the test before the changes were made. These are discussed in Section 8.2. The second part contains the results of the test after the changes were made. These are discussed in Section 8.5.

The purpose of these tests is to test the manual for the following four aspects:

- Understandability
- Helpfulness in the discovery of more attacks
- Degree of guidance regarding the process of creating an attack tree
- Quality of the resulting attack tree in regards of understandability for others

To test these aspects, a group of people that are familiar with the same system were asked to create an attack tree for that system. To know what the exact help of the manual is, the ideal test would be one where half the group works with the manual and the other half has to figure it out on their own using any source they can find. This would however require too much time from the participants. Furthermore, it would give too much opportunity to create a completely different type of risk analysis or different type of attack tree. Because of that, it was decided to do this test with two versions of the manual; The normal manual and a version that only explains the basics of an attack tree as we described in Section 4. This would guide them in the same direction as the actual manual, however without the structure of the model included.

To summarise: this test will not give an exact baseline for the manual, though the usefulness will still be quantified. For the addition of the model there will be a baseline as there will be a version with and without the model included.

8.1 Setup

In this section, the setup of the experiment will be explained. Each paragraph represents a step in the test. First information that the participants need is provided. Then the participants can create their attack tree and reflect on the process. Finally, their attack tree will be reviewed by other participants.

Provide Motivation The participants are also provided with a small description of why one would want to create an attack tree in the first place and what the motivation for this specific case would be. This would help them get in the right mindset and help them set the goal for their attack tree. In real life situation, this information could come from the security officer, or from their own wish to do a risk analysis.

Provide Case This test was conducted at the water sports complex at the University of Twente. The goal of the attack tree to be created was to steal the new fridge from the kitchen. This kitchen is in a room that is located roughly in the middle of the building and is locked down whenever it is not used by an association. Furthermore, the kitchen contains rather expensive machines and this scenario could provide a risk analysis for the managers of the kitchen as a side effect. This case is small enough that creating an attack tree would not take too long for participants and would also make for a decent attack tree. The participants are people who are very familiar with the building and security systems in place.

Provide Manual Half of the participants will be given a basic version of the manual while the other half will be given the full version. The basic version of the manual can be found in Appendix C. The full manual can be found in Appendix B. To minimise prejudice, the participants are not told which manual they are using.

Create Attack Tree Participants are given 20 minutes to create their attack tree. This sets an equal opportunity for all participants to create an attack tree to the best of their abilities. It ensures participants do not spend much more time than others to create better attack trees as this would result in a qualitative difference not caused by the manual.

Quantitative Reflection After creating an attack tree with either of the manuals, the participants have to reflect on six statements. This is done by assigning a number to each statement. These numbers range from one (disagree) to five (agree). The following statements will be rated by the participants:

1. It was clear what the goal of the assignment was
2. It was clear what i had to do
3. The manual was clearly understandable
4. The manual helped me discover more attacks
5. The manual gave structure to the process of creating this attack tree
6. Now, i feel like i know how to create a proper attack tree

These questions can be easily quantified so that the difference between the manual with and without the model is clear.

To add another quantifiable factor, the number of nodes of each attack tree is counted. Even though this might not exactly represent the number of attacks found, it is likely that a larger attack tree includes more attacks. In any case, it is certainly an indicator of detail.

Qualitative Reflection Thereafter two optional open questions are asked; The first being what field of work or study they are in. This could give insight in what fields can work better or worse with the model. Besides that, more importantly, it could provide proof that different kinds of system experts are able to create attack trees with this manual.

The second question is whether there are any other remarks regarding the model. This allows the user to give any other feedback or comments that was not asked about. The results of these questions will be discussed in Section 8.2.

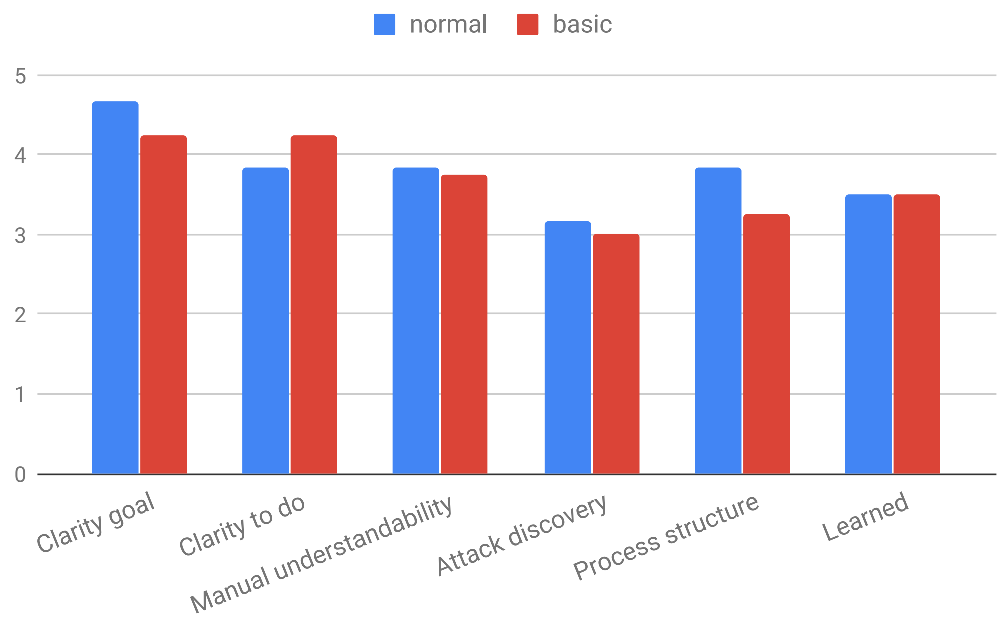


FIGURE 8.1: Results of usability test 1. Statements are rated 1 (disagree) trough 5 (agree) by each participant after working with the basic manual or with the full version. The numbers in the table are averages of those ratings.

Peer Reviews To answer the question regarding improved understandability of the resulting attack tree for others, there will be one more round of evaluation. This time each participant will reflect on two attack trees from other participants; one made with the basic manual and one made with the normal manual. Participants will have to choose which attack tree is better according to them and why.

Which attack tree is better is easily quantifiable while the 'why' provides us with insight whether it is actually caused by the model or if it is caused by something else. The results of this evaluation will be discussed in Section 8.2.

8.2 Results - Part 1

Questions A group of 12 people participated in this test so far. Six used the normal manual to create an attack tree and six used the basic manual to create an attack tree. The averaged results of the quantifiable statements are listed in Table 8.1. The improvements are small, though existent. Mainly statements three, four and five are important indicators for the usability of the manual. In either case, the manual seems to be workable at least.

In Figure 8.2, the average number of nodes is listed for attack trees made with and without the manual. Here, it becomes visible that, on average, using the full manual doubles the number of nodes of the attack trees. This would indicate that, even though the answers to question four are not overwhelmingly positive, the manual does help to discover more attacks.

Peer Reviews and Reflection For deciding which attack trees were better, 23 votes were cast. From these votes, 14 went to an attack tree that was made with the full manual, while 9 went to an attack tree that was made with the basic version of the manual. This

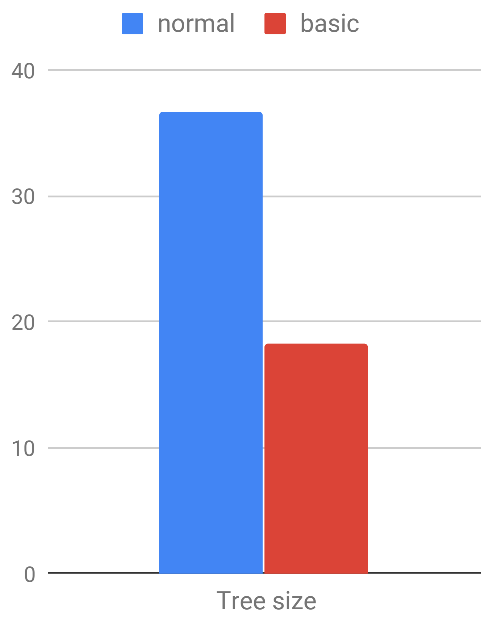


FIGURE 8.2: Results of usability test 1. The number of nodes per attack tree were counted. These are averaged in this figure.

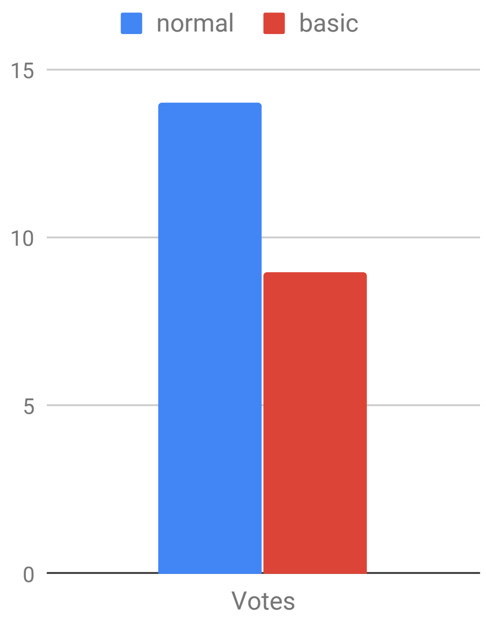


FIGURE 8.3: Results of the peer reviews of usability test 1. These are the number of votes for the attack trees made with the full manual versus with the basic manual.

indicates a clear preference to the attack trees created with the normal manual, though there is room for improvement.

When analysing the attack trees it became clear that even across participants using the same manual, the attack trees would widely vary in quality. I.e. some participants using the basic manual would create a clear attack tree which had a structure that approached parts of the model, while other participants that did use the full manual created attack trees that incorporated nothing from the model. After this was discovered, participants were asked about their field of work to explore whether the difference could be caused by a technical background. However, nothing could be concluded from this.

The open feedback section suggests that this might be caused by a combination of reasons. First of all, a clear explanation of exactly what to do with the model was missing. All the parts are there, however it was often not understood how they should be combined to create a proper attack tree. This became much clearer after a short explanation after the test. Second, the example helped to quickly understand what was expected. However, this also provided the opportunity to just look at the example and only roughly skip trough the rest of the manual. Besides that, participants complained that it limited their creativity as they were guided too much in a single direction by the example. So the example helped understand what was expected quicker, but therein helped people skip important details as well as limiting creativity.

Aside from the example and extra explanation, participants expressed that it would be helpful to have the splits and layers a little more concrete. A series of questions could help in guiding them in the right direction for each layer or split.

The last observation worth discussing is the fact that participants seemed to quickly resort to creating a graph instead of a tree. This was often caused by the wish to avoid duplication.

8.3 Changes to the Manual

To improve the manual, it has been updated according to the results of this test so far. These changes include the following:

- The example has been removed. To still show users what the different types of nodes look like, a new picture has been added which only shows each type of node, along with the note that a node can have any number of child nodes.
- A quick guide has been added which provides a question for each layer to get started.
- A block with a more concrete example for each split has been added to help users better understand what each split means.
- A (differently coloured) block has been added which describes what block tells what and how the user is supposed to use all the information that is in the manual. It brings the other blocks together.

The rest of the changes are small changes in wording and layout. The resulting second version of the manual can be found in Appendix D.

8.4 Changes to the Setup

Besides the changes to the manual, the setup was also changed to accommodate for differences between participants and to improve the number of participants. The changes that were made are described below:

Different Cases Not only because of lack of participants, but also to make sure that the results are not dependant on the specific case, a different case is studied this time. Moreover, this time there are two separate cases and the possibility to define a different case. It is possible for participants to evaluate attack trees from another case as the aim of the test is not to check if the attack trees are correct, but to test their clarity and number of attacks.

The first case is at a student home which has a relatively important server running in one of the rooms. The goal of the attack tree will be to make the server unreachable. This case is relatively simple; a large part of the security is the fact that few people know that the server is there and the fact that 11 students live there.

The second case is a case at Nedap N.V. The goal of the attack tree in this case is to steal one or more laptops from one of the offices. Even though there are likely to be more important things to be stolen or broken, this is a relatively simple case of which many employees will be able to find a way for.

Finally, participants can define their own case after a description of what such a case should look like and how large it should be. Most cases that were defined in this way were checked by us before the test was continued.

Updated Manual Of course, the manual is updated according to the results of the test so far. The basic version of the manual can be found in Appendix E and the normal version of the manual can be found in Appendix D.

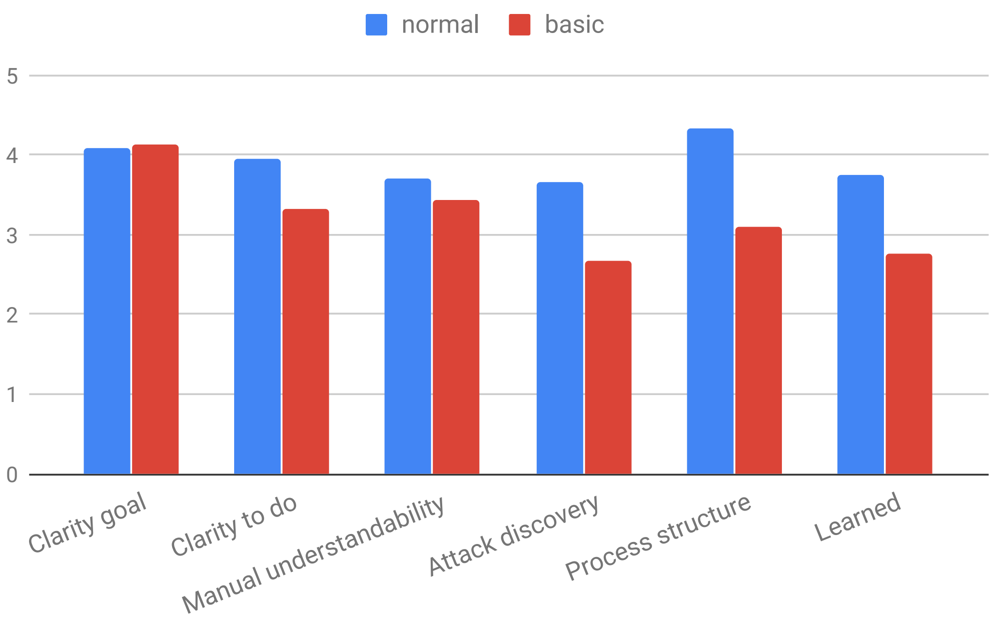


FIGURE 8.4: Results of usability test 2. Statements are rated 1 (disagree) trough 5 (agree) by each participant after working with the basic manual or with the full version. The numbers in the table are averages of those ratings.

Create Two Attack Trees Instead of letting participants create an attack tree using one version of the manual, the participants had to create an attack tree with each manual; First an attack tree with the basic manual, then a new one with the normal manual. Using both manuals after each other would have close to no effect on the results. Everything in the basic version is also what would be read first in the normal version, so no extra information is given. By using both manuals, attack trees can be properly compared without interference from personal conceptions of the model.

Peer Reviews During the peer reviewing of the attack trees, instead of comparing two random trees, the two trees from the same participant will be compared. This avoids any influence of differences in views on the model between the participants.

8.5 Results - Part 2

Questions 22 more people participated in the test after the changes. They all created an attack tree with the basic manual and one with the normal manual. The averaged results of the quantifiable statements are listed in Figure 8.4. This time, the numbers show more improvement. Especially statements 4 and 5 regarding exploring more attacks and giving structure. The rating of those statements improved by 38 and 40 percent respectively.

The changes that were made to the model clearly improved the understandability of the manual and with that the use of the model.

In Figure 8.5, the average number of nodes is listed for attack trees made with and without the manual. The improvement is not as significant as in the first part of the test, however the average number of nodes the attack trees have is increased by 61 percent by the manual.

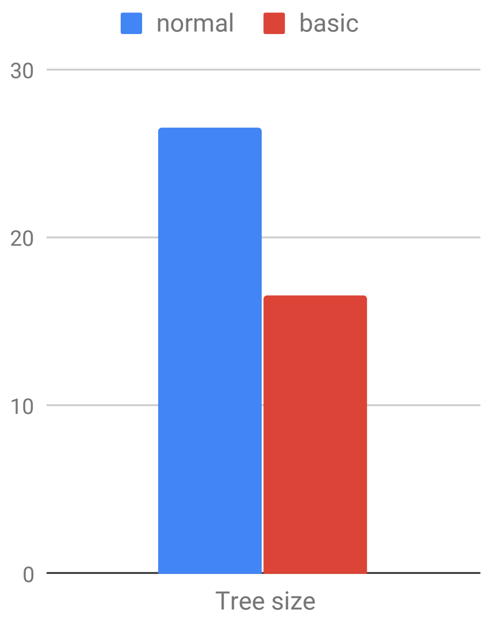


FIGURE 8.5: Results of usability test 2. The number of nodes per attack tree were counted. These are averaged in this figure.

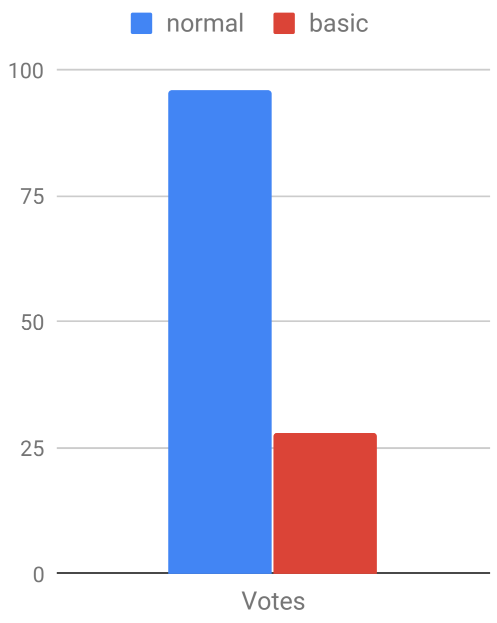


FIGURE 8.6: Results of the peer reviews of usability test 2. These are the number of votes for the attack trees made with the full manual versus with the basic manual.

Peer Reviews and Reflection For deciding which attack trees were better, 124 votes were cast. From these votes, 96 went to an attack tree that was made with the full manual, while 28 went to an attack tree that was made with the basic version of the manual. In Figure 8.6 it becomes visible that there is a clear preference to the attack trees created with the normal manual; more so than in the previous test.

From the open feedback, it became clear the people prefer to have an example so they know what is expected from them visually. After the previous test the example was removed. After discussing this with several participants, it became clear that an example with only nodes and lines without text would also suffice. This could then even be improved by indicating which node belongs to which layer.

Additionally, combining this information with the field of work the participants were in gave a good indication of what models they normally work with: Without a visual example, many participants with a technical background produced an attack tree which was rather close to what was expected. Participants that worked in fields like business administration and management functions often resorted to a visualisation type that is often used for drawing processes.

With the new manual and the new test, the resulting attack trees varied less. This was between attack trees that were compared, but also between all attack trees made with the normal manual.

Chapter 9

Evaluation - Qualitative

As a demonstration of the designed manual and as a final test, a case study was conducted. Although previous tests were also based on relatively real scenarios, in this case study everything had to be correct. The goal for such a case study is, of course, to provide a company with insights in weak spots in a system of theirs. However if a company already has (some) insight in their risks and weak spots, it would be interesting to compare the results.

9.1 The Company

The company where the case study was conducted wishes to remain anonymous as the resulting attack tree of course reveals weak spots. The company in question is a software consultancy bureau; They create software products for other companies. Therefore they have access to the applications and possibly data of many of their clients. They have indicated that protecting their data, and even more important the data of their applications, is most important to them. Hence the goal of the attack tree: *'Steal data'*. Such a goal leaves enough room to explore many attacks while also being relatively concrete.

To create an attack tree for this company, an interactive interview was conducted with the director and a project lead. Guided by the manual and assisted by an explanation to avoid giving them just an assignment, the process of creating an attack tree was explained. Thereafter we used the layers, questions, splits and examples as a guide trough a brainstorming session regarding how an attacker could steal data. The result of this further discussed in Section 9.2. Including the introduction into attack trees, the interview took a total of 45 minutes and resulted in a well structured attack tree.

9.2 The Attack Tree

The attack tree that resulted from this case study is shown in Figure 9.1. Instead of the standard colouring, the nodes of the attack tree are coloured corresponding to the layer they are a part of and a line is drawn between each of the layers. By doing this, it is easy to see the use of layers.

As previously mentioned, the goal in this attack tree is to steal data. We started by splitting the goal into three sub-goals. These sub-goals represent the locations where the data could be obtained from. This could be from the cloud where the applications are running, it could be from a local device, or it could be from the client who owns the application. The attack opportunities from the side of the client who owns the applications are the responsibility of that client. Because of this, there was little knowledge regarding their systems and that node was made a separate segment. The sub-trees for stealing the data from the cloud and stealing the data from a local device are explained next.

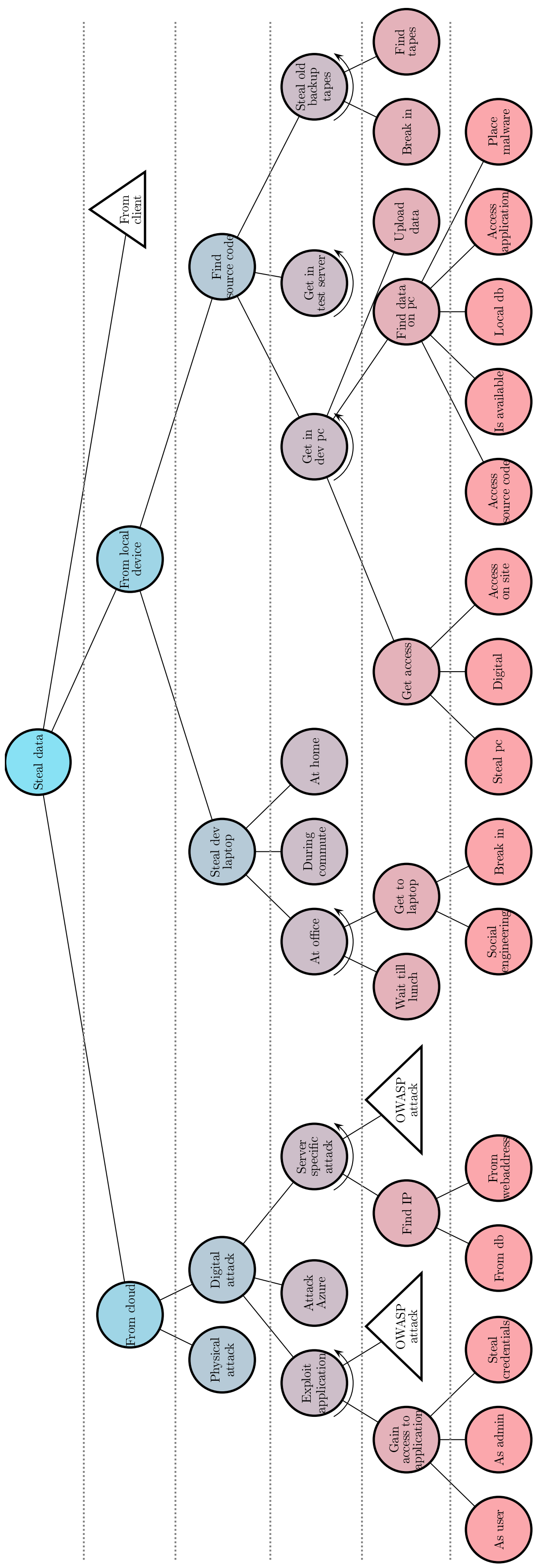


FIGURE 9.1: Attack tree constructed as case study for an anonymous company. Instead of the standard colouring, in this attack tree nodes are coloured based on the layer they are part of. Furthermore, each layer is separated by a dotted line.

9.2.1 Steal from The Cloud

Attacks to the cloud were split into digital and physical attacks. Even though at first it seemed relatively easy to think of a way to get to a server, ideas quickly halted at the realisation that no one knew where the servers were actually located. Therefore, this branch was left unrefined. To add to this, this would be more fitting as a part of the risk analysis for Azure - the hosting service. Three digital attacks were defined. The first one was to attack Azure in general, however with the same reasoning as was used for physical attacks on the cloud, this node was also left.

The second attack was to exploit the application to get to the server. This would require access to the server, which could be gained by being a user, admin (employee of the client) or by social engineering a person which is one of those. After gaining access an OWASP attack might be executed. OWASP attacks are put in the tree as a component as it is a grouping of attacks. These are further discussed in Section 9.2.3.

The third attack was to do an attack directly targeted at the server. This would require the IP address of the server which could be gained from a public database or from the web address that is related to the service of the clients. After determining the internet address of the server, an OWASP attack might be executed.

9.2.2 Steal from Local Device

Stealing data from a local device was split into two attack types: stealing the laptop of a developer and finding source code which could give insight or credentials for accessing data.

The laptop of a developer could be stolen at home, while commuting or while at the office. The first two are personal responsibility and there for not further expanded upon. The latter however, is of relevance in this analysis. Unless the attackers is doing an armed assault, the attack will have to be done during lunch as at night all laptops are brought home by the employees. The second step was to get to the laptop, which could be done by social engineering or by simply breaking a window.

Finding source code could be done gaining access to the pc of one of the developers, by getting into a test server or by stealing old backup tapes. The latter is relatively simply done by breaking in and searching for the tapes. Gaining access to the test server is almost exactly the same as getting access to the PC of one of the developers, except it is more difficult and contains more data. Because these are not aspects represented in this attack tree, this attack will only be displayed once; under the node to get into the PC of a developer.

To get into the PC of one of the developers, the first step is to get access. This could be done by stealing the PC, by hacking it or by gaining access on site (e.g. when a laptop is left unlocked and unguarded). The second step is to find data on the PC. There are many options for this, some of them being: Access source code, data is simply stored in a file, a local database is still open, access one of the developed applications or place malware to gain data later. The final step is to upload the data.

9.2.3 OWASP

On two occasions there is a component node named 'OWASP' in the attack tree. These refer to the OWASP vulnerabilities and especially the top 10 of those [86] which are listed below.

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure

4. XML Eternal Entities
5. Broken Access Control
6. Security Misconfiguration
7. Cross-site Scripting
8. Insecure Deserialisation
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

According to [86], these vulnerabilities are the most common and important vulnerabilities for web-based applications.

The company in question regularly hires a professional to test all systems and applications for these vulnerabilities. Therefore, it is of no further interest to work out how these attacks work in detail. It is however, important to know that these vulnerabilities exist and what is done to mitigate them.

9.3 Influence of the Manual

Of course there is no comparison of this attack tree with an attack tree created without the manual. However, especially with the current colouring, it is clearly visible that this attack tree incorporates a lot of aspects from the model. First of all the layers are clearly used, no layers had to be skipped even and interest in what exactly happens faded once the nodes in the choices layer were added.

Second, the splits have helped with inspiring different directions to discover more attacks. Some notable splits are certainly the components split at the top and the obvious splits between physical and digital. Besides those, although FES splits are not used very clearly, they were a great source of inspiration at many points in the tree. Finally, all attacks that are refined further are refined by a steps split.

Part IV

Conclusions

In the previous parts we have designed a model as well as a manual to convey that model and how to work with it. Thereafter we have evaluated it extensively trough quantitative and qualitative evaluation. In this part the conclusions that came forth in this thesis will be described. This part also includes the discussion where the virtues and flaws of this thesis are discussed and finally a chapter regarding potential future work.

Chapter 10

Conclusion

The goal of this thesis was to *allow (security) experts to be able to create a well structured attack tree within an acceptable time frame* as stated in Chapter 2. In order to achieve this goal, three research questions were determined to assist in reaching this goal. In this chapter the results of this research are discussed. First the research questions in Section 10.1 and thereafter the goal and resulting manual for attack trees in Section 10.2.

10.1 Research Questions

To recite the research questions, they are listed below once more:

R.1 *What quantitative or qualitative metrics indicate that an attack tree is structurally well made for the use of security analysis?*

R.2 *Which guiding principles and building blocks are used by experienced security experts?*

R.3 *To what extend does a manual improve the ease and speed of the manual creation of an attack tree?*

The answers to questions R.1 and R.2 are very much entangled and therefore their answer is mostly the same. In this thesis the assumption was made that experts in the field of attack trees generally create structurally well made attack trees.

After the extensive literature study in Chapter 6, we can conclude that the proper incorporation of layers is the main indicator of a well structured attack tree. Besides layers, defining certain splits adds more structure and standardisation to attack trees. Also, there are certain practices which do not change the content of the attack tree, but help to keep it manageable. Furthermore, from the evaluation of the manual in Chapter 8 we can add to this conclusion that a strict tree structure was preferred over a graph structure, with an exception for duplication of large sub trees.

Question R.3 is a more difficult question to answer as there is no baseline for how easy it is to create an attack tree, or how long that normally takes. To add to that, it is hardly reasonable nor possible to ask test participants to create an attack tree by themselves and with no further explanation. This would result in results varying so widely that still no conclusions could be drawn from it. That being said, in Section 8 it has been proven that when system experts use a manual which describes the basic building blocks for attack trees, as well as a more advanced structure for attack trees, the resulting attack trees are more extensive and more clear to other system experts. Moreover, the experience of the creator of the attack tree is also improved.

10.2 Research Goal and Success of the Manual

From the evaluation in Chapter 8 and the case study in Chapter 9, it can be concluded that this manual allows system experts to be able to create well structured attack trees within an acceptable time frame. When given just the manual, many participants of the tests proved to be able to create an attack tree conform the model within 20 minutes (Longer when the manual is presented instead of read as is with the case study). Because the model was designed to represent the structure that is generally used by professionals, this would mean that attack trees that are conform to it are well structured.

Also, from that same evaluation, it can be concluded that system experts find it fairly helpful to have a manual to hold on to while creating an attack tree. Incorporating the model from this thesis into the manual only slightly improved this experience. The resulting attack trees however, improved very significantly; The majority of system experts found attack trees better (more attacks and more clear) when they were made with the manual that had the model incorporated in it. To add to that, the number of nodes of the attack tree doubled on average, suggesting that more attacks were discovered.

Chapter 11

Discussion

Naturally, not everything went completely according to plan. In this chapter it is discussed what was accomplished and what was not, as well as what could have been done better.

Manual for Attack Trees First of all, a manual for creating attack trees was created. This manual proved to help system experts with creating a proper attack tree while not taking hours if not days of research. There is still some margin for variety in interpretation. We found that without enough motivation, people tend to not read part of the manual and it shows. This also brings us to the point of the (user experience) design. Even though people can work with it relatively well, the user experience can certainly be improved upon.

Model for Attack Trees To support the manual, a model for attack trees was designed. This model is arguably more interesting than the manual as it provides new opportunities for standardisation and extended tooling. The model is based on attack trees created by the most prominent researchers in the field of attack trees. Furthermore, it can be extended and improved to accommodate for better and more extensive attack trees.

Besides the model, a number of other common practices were identified and evaluated. Some proved to be useful for certain situations, while others were not.

No Baseline for Evaluation We are sorry to say that it was not doable to set proper baselines for the creation of attack trees without any previous knowledge. This goes for the time it would take as well as the quality of the resulting attack tree. However, during the evaluation it still became clear that a manual at all helps with reducing time required and increasing the quality of the resulting attack tree. Although purely speculation, it is a relatively safe assumption that creating an attack tree takes much longer than 20 minutes without help or instructions.

Two Takes on Evaluation Although still successful, the first part of the usability test had quite some imperfections. This was mainly caused by a difference in interpretation of the model between the participants. This caused a wildly varying attack tree quality, even between participants using the same manual. After slightly changing the manual and the test, the variation between participants decreased and the test results improved.

Chapter 12

Future Work

In this chapter it is discussed what was accomplished and what was not as well as what could have been done better. Furthermore, recommendations for future work are also discussed.

Improve Usability Although LaTeX helped a lot with making the manual visually appealing, it could still be improved both visually and content wise. The user experience proved to be decent with the current model, however we believe that with more effort, this could be improved upon. A possible improvement could be reducing the amount of text to be read while still making clear how the model is supposed to work. Although this seems simple, it has proven to be rather difficult.

Additional Splits The manual designed in this thesis helps system experts to create attack trees. It also provides common splits that can be used either literally or as a source of inspiration for new sub trees. To improve the manual, more commonly used splits could be defined. When the number of predefined splits is large enough there could even be a point where only those certain splits have to be used to create a proper attack tree. This would be ideal for standardisation of attack trees. An important aspect here is that these splits should not be limiting the creativity of the user unless all useful possible splits are documented, which is nearly impossible and brings new problems yet again.

Tooling A part of the motivation for creating this manual is standardisation, and standardisation provides structure for tooling. Of course, there are multiple tools for creating attack trees already, however if attack trees get a more defined structure, it is becomes easier to add more extensive tooling support for this. Furthermore, such a structure could provide an opportunity for better semi-automated attack tree creation; A program could ask a series of questions to the user and with the answers given, the rest of creating an attack tree could be automated.

Part V

Appendices

Appendix A

Data of the Literature Analysis

N = 60	Layers (nodes only)							Layers type (below)						Layers type (above)					
Paper	Goals	Sub-Goals	Attack types	Attacks	Process	Choices	Spec/Org	Goals	Sub-Goals	Attack types	Attacks	Process	Choices	Goals	Sub-Goals	Attack types	Attacks	Process	Choices
[33]	1	1	1	1	1	1	0	OR	OR	OR	SAND	OR			OR	OR	OR	SAND	OR
[30]	1	0	2	1	1	1	1	OR		OR	AND	OR				OR	OR	AND	OR
[34]	1	1	1	1	0	0	0	AND	OR	OR					AND	OR	OR		
[35]	1	0	0	1	1	1	1	OR			AND	OR					OR	AND	OR
[36]	1	0	1	1	1	0	0	OR		OR	OR					OR	OR	OR	
[37]	1	0	2	1	0	0	0	OR		OR						OR	OR		
[38]	1	0	0	1	4	0	0	OR			AND						OR	AND	
[39]	1	1	1	1	0	1	0	OR	OR	OR	OR				OR	OR	OR		OR
[40]	1	0	0	1	1	1	1	OR			AND	OR					OR	AND	OR
[41]	1	0	1	1	2	0	0	OR		OR	SAND					OR	OR	SAND	
[42]	1	0	0	1	1	1	1	OR			SAND	OR					OR	SAND	OR
[43]	1	0	1	1	1	1	0	OR		OR	AND	OR				OR	OR	AND	OR
[44]	1	0	1	1	1	1	0	OR		OR	AND	OR				OR	OR	AND	OR
[45]	0	0	0	1	1	1	1				SAND	OR						SAND	OR
[46]	1	0	0	1	1	1	0	OR			AND	OR					OR	AND	OR
[47]	1	2	0	1	0	0	0	OR	OR						OR		OR		
[48]	1	0	2	1	1	0	0	OR		OR	AND					OR	OR	AND	
[49]	1	0	2	1	1	0	0	OR		OR	AND					OR	OR	AND	
[50]	1	2	0	1	1	0	0	SAND	AND		AND				SAND		AND	AND	
[4]	1	0	1	1	1	1	0	OR		OR	AND	OR				OR	OR	AND	OR
[51]	1	1	0	1	4	1	0	OR	OR		OR	OR			OR		OR	OR	OR
[17]	0	0	0	1	3	1	0				AND	OR						AND	OR
[52]	1	0	0	1	1	0	0	OR			AND						OR	AND	
[53]	1	0	2	2	1	1	0	OR		OR	AND	OR				OR	OR	AND	OR
[54]	1	1	1	1	0	0	0	OR	OR	OR					OR	OR	OR		
[55]	1	0	1	1	1	1	1	OR		OR	AND	OR				OR	OR	AND	OR
[56]	1	1	0	1	1	0	1	AND	OR		AND				AND		OR	AND	
[57]	1	1	0	1	1	0	1	AND	OR		AND				AND		OR	AND	
[58]	1	0	1	2	0	0	0	OR		OR						OR	OR		
[59]	1	1	0	1	0	1	0	OR	OR		OR				OR		OR		OR
[60]	1	0	2	1	1	1	0	OR		OR	AND	OR				OR	OR	AND	OR
[61]	1	0	2	1	1	0	0	OR		OR	AND					OR	OR	AND	
[62]	1	3	1	1	1	0	1	OR	AND	OR	AND				OR	AND	OR	AND	
[15]	1	0	2	1	1	0	0	OR		OR	AND					OR	OR	AND	
[2]	1	0	1	1	1	0	0	OR		OR	AND					OR	OR	AND	
[63]	1	1	0	1	0	0	1	AND	OR						AND		OR		
[64]	1	1	0	1	0	1	0	OR	OR		OR				OR		OR		OR
[65]	1	0	2	1	0	0	0	OR		OR						OR	OR		
[66]	1	0	1	1	0	0	0	OR		OR						OR	OR		
[67]	1	0	1	1	1	0	0	OR		OR	AND					OR	OR	AND	
[12]	1	0	2	1	1	0	0	OR		OR	AND					OR	OR	AND	
[13]	1	2	1	1	1	0	0	AND	AND	OR	AND				AND	AND	OR	AND	
[68]	1	1	0	1	0	0	0	OR	OR						OR		OR		
[69]	1	0	1	1	0	1	0	OR		OR	OR					OR	OR		OR
[70]	1	1	1	1	0	0	0	AND	OR	OR					AND	OR	OR		
[20]	1	0	1	1	1	1	0	OR		OR	AND	OR				OR	OR	AND	OR
[71]	1	1	2	1	1	0	0	OR	OR	OR	AND				OR	OR	OR	AND	
[72]	1	0	0	1	1	1	1	OR			AND	OR					OR	AND	OR
[73]	1	1	2	1	0	0	1	OR	AND	OR					OR	AND	OR		
[74]	1	1	1	2	0	0	0	OR	OR	OR					OR	OR	OR		
[75]	1	0	0	1	1	1	1	OR			AND	OR					OR	AND	OR
[76]	0	0	0	1	1	1	1				AND	OR						AND	OR
[77]	1	1	1	1	1	1	0	SAND	SAND	SAND	OR	SAND			SAND	SAND	SAND	OR	SAND
[78]	1	1	1	1	1	0	0	AND	OR	OR	AND				AND	OR	OR	AND	
[79]	1	1	0	1	1	0	1	SAND	OR		AND				SAND		OR	AND	
[80]	1	0	0	1	1	1	0	OR			AND	AND					OR	AND	AND
[81]	1	1	0	1	1	1	0	SAND	OR		SAND	OR			SAND		OR	SAND	OR
[82]	1	0	1	1	1	1	0	OR		OR	AND	OR				OR	OR	AND	OR
[83]	1	0	1	1	0	0	0	OR		OR						OR	OR		
[84]	1	1	1	1	1	0	1	OR	AND	OR	AND				OR	AND	OR	AND	

Table A.1: Results of the analysis of attack trees from previous literature regarding layer occurance, size and type. 60 attack trees were analysed. For each the size of each layer is listed as well as the type of split above and below it. If a layer has different types of splits, the one that occurs the most in that layer is listed.

N = 60	steps						FES						components						phys/digi					
Paper	Goals	Sub-Goals	Attack types	Attacks	Process	Choices	Goals	Sub-Goals	Attack types	Attacks	Process	Choices	Goals	Sub-Goals	Attack types	Attacks	Process	Choices	Goals	Sub-Goals	Attack types	Attacks	Process	Choices
[33]					1				1					1										
[30]					1				1												1			
[34]		1								1						1								
[35]					1																			
[36]									1															
[37]									1															
[38]					1											1								
[39]						1																		
[40]					1					1												1		
[41]					1																			
[42]					1					1														
[43]					1																			
[44]					1										1									
[45]					1																			
[46]					1											1								
[47]														1										
[48]																1								
[49]										1						1								
[50]		1						1																
[4]					1																			
[51]					1							1												
[17]					1																			
[52]					1											1								
[53]															1									
[54]									1					1										
[55]															1									
[56]					1					1														
[57]					1					1														
[58]																								
[59]										1		1												
[60]												1												
[61]					1											1					1			
[62]														1	1									
[15]									1															
[2]										1														
[63]														1		1								
[64]										1				1										
[65]															1									
[66]																								
[67]					1																1			
[12]																1								
[13]										1				1		1				1				
[68]														1										
[69]									1															
[70]																					1			
[20]					1					1														
[71]																								
[72]																		1						
[73]																								
[74]									1					1	1									
[75]					1					1														
[76]					1																			
[77]		1	1		1									1										
[78]														1			1							
[79]		1								1														
[80]																								
[81]		1			1																			
[82]										1														
[83]															1									
[84]					1						1													

Table A.2: Results of the analysis of attack trees from previous literature regarding split location. 60 attack trees were analysed for splits and in which layer they occur. In this table the location of a split depends on the location of its resulting nodes.

N = 60	steps						FES						components						phys/digi					
Paper	Goals	Sub-Goals	Attack types	Attacks	Process	Choices	Goals	Sub-Goals	Attack types	Attacks	Process	Choices	Goals	Sub-Goals	Attack types	Attacks	Process	Choices	Goals	Sub-Goals	Attack types	Attacks	Process	Choices
[33]				1				1					1											
[30]				1			1												1					
[34]	1								1						1									
[35]				1																				
[36]							1																	
[37]							1																	
[38]				1									1											
[39]				1																				
[40]				1			1												1					
[41]				1																				
[42]				1			1																	
[43]				1																				
[44]				1									1											
[45]				1																				
[46]				1									1											
[47]													1											
[48]															1									
[49]									1						1									
[50]	1						1																	
[4]				1																				
[51]				1							1													
[17]				1																				
[52]				1									1											
[53]													1											
[54]								1					1											
[55]													1											
[56]				1				1																
[57]				1				1																
[58]																								
[59]								1		1														
[60]											1													
[61]				1											1				1					
[62]													1	1										
[15]							1																	
[2]									1															
[63]													1	1										
[64]								1					1											
[65]													1											
[66]																								
[67]				1															1					
[12]															1									
[13]									1				1		1				1					
[68]													1											
[69]							1																	
[70]																				1				
[20]				1					1															
[71]																								
[72]																	1							
[73]																								
[74]								1					1	1										
[75]				1			1																	
[76]				1																				
[77]	1	1		1									1											
[78]													1			1								
[79]	1							1																
[80]																								
[81]	1			1																				
[82]									1															
[83]													1											
[84]				1						1														

Table A.3: Results of the analysis of attack trees from previous literature regarding split location. 60 attack trees were analysed for splits and in which layer they occur. In this table the location of a split depends on the location of its origin node.

N = 60		Practices						
Paper	size	Segmentation	Subtrees	List display	One attack	Only attacks	Countermeasures	Attribution
[33]	70	0	0	1	0	0		1
[30]	67	1	1	0	0	0	1	0
[34]	29	0	0	0	0	1	0	1
[35]	10	0	0	0	0	0	0	0
[36]	10	0	0	0	0	1	0	0
[37]	110	1	0	1	0	1	0	1
[38]	28	0	1	0	0	0	0	1
[39]	26	0	0	1	0	0	0	1
[40]	33	0	0	0	0	0	1	1
[41]	20	1	0	0	0	0	0	1
[42]	21	0	0	0	0	0	0	1
[43]	35	1	0	0	0	0	0	0
[44]	23	0	0	0	0	0	0	1
[45]	31	1	0	0	1	0	0	0
[46]	12	0	0	0	0	0	0	0
[47]	48	0	0	0	0	1	0	0
[48]	25	0	0	0	0	0	1	0
[49]	16	0	0	0	0	0	0	1
[50]	12	0	1	0	0	0	0	1
[4]	18	0	0	0	0	0	0	0
[51]	21	0	0	1	0	0	0	0
[17]	17	0	0	0	1	0	0	0
[52]	7	0	0	0	0	0	0	0
[53]	28	0	0	0	0	0	0	0
[54]	24	0	0	0	0	1	0	1
[55]	21	0	0	0	0	0	0	1
[56]	15	1	0	1	0	0	0	0
[57]	11	0	0	0	0	1	0	0
[58]	14	0	0	0	0	1	0	0
[59]	15	0	0	0	0	0	0	0
[60]	51	0	0	1	0	0	0	0
[61]	20	0	0	1	0	0	0	1
[62]	65	0	1	0	0	0	0	1
[15]	13	0	0	0	0	0	0	0
[2]	13	0	0	0	0	0	0	1
[63]	7	0	1	0	0	1	1	1
[64]	24	0	0	1	0	0	0	1
[65]	14	0	0	0	0	1	0	0
[66]	8	0	0	0	0	1	1	0
[67]	12	0	0	0	0	0	0	0
[12]	25	0	0	0	0	0	1	0
[13]	42	0	1	0	0	0	1	0
[68]	13	0	0	0	0	1	0	0
[69]	16	0	0	0	0	0	0	0
[70]	14	0	0	0	0	1	0	0
[20]	12	0	0	0	0	0	0	1
[71]	18	0	0	0	0	0	1	1
[72]	16	0	0	0	0	0	1	0
[73]	18	0	0	1	0	1	1	0
[74]	15	0	0	0	0	1	0	0
[75]	9	0	0	0	0	0	1	0
[76]	5	0	1	0	1	0	1	0
[77]	25	0	0	0	0	0	0	0
[78]	9	0	0	0	0	0	0	0
[79]	8	0	1	0	0	0	1	0
[80]	13	0	0	1	0	0	0	1
[81]	45	0	0	1	0	0	0	0
[82]	36	0	0	0	0	0	0	0
[83]	22	0	0	0	0	1	1	1
[84]	17	0	0	0	0	0	0	0

TABLE A.4: Results of the analysis of attack trees from previous literature regarding size and practices. 60 attack trees were analysed practices and their size was listed.

Appendix B

Manual for Attack Trees - Version 1

Manual for Attack Trees

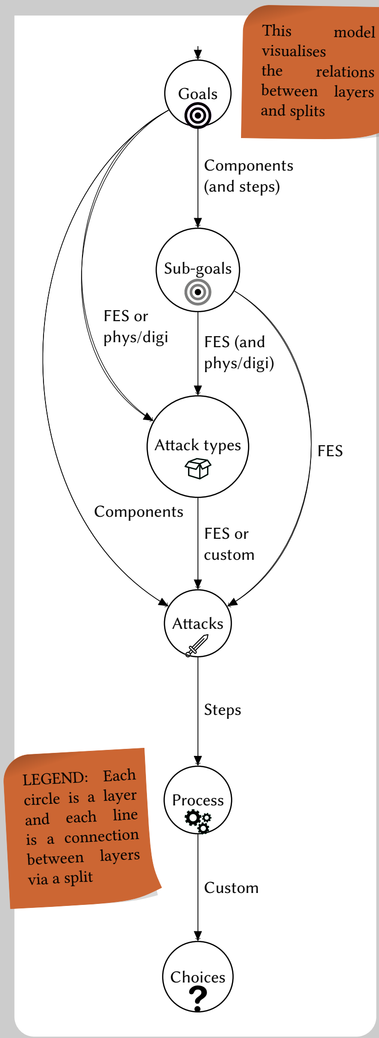
Tim Sonderen
University of Twente & Nedap N.V.

Introduction

Attack trees are a tool to explore vulnerabilities in a system, be it physical, digital or both. The idea of creating an attack tree is for the security analysts and/or system experts to look at the system from the attackers' side. Because of the nature of attack trees, most information is gained by the creation of the attack tree itself. A small example can be found in the example block.

Goals

The goal of building an attack tree is to **explore attacks** on a system and expose vulnerabilities.
Therefore the root (first node) of an attack tree is a goal an attacker would have, e.g. 'Steal valuables'.



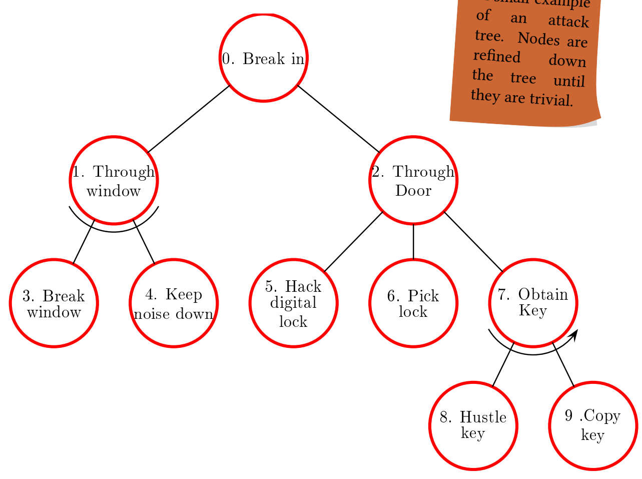
This model visualises the relations between layers and splits

LEGEND: Each circle is a layer and each line is a connection between layers via a split

Basic building blocks

An attack tree consists of the following parts:

- **Root** node - The goal of the attack and start of the attack tree (example node **0**)
- **OR** nodes - A node of which only one of its child nodes needs to be successful (example node **2**)
- **AND** nodes - A node of which all of its child nodes need to be successful (example node **1**)
- **SAND** nodes - Like an AND node, but the child nodes are done in order (example node **6**)
- **LEAF** nodes - Leaves at the bottom of the attack tree, trivial attacks (example node **3**)



A small example of an attack tree. Nodes are refined down the tree until they are trivial.

Creating an attack tree

After the root is set, the rest of the tree can be created by refining each node until the action in the node becomes trivial. This can also be seen in the example.

Layers

Attack trees can be divided in layers which are most often only one node deep. The following six layers are commonly used:

- **Goals** - Is generally just the root node
- **Sub-goals** - Groups smaller goals / different approaches
- **Attack types** - Groups attacks by type
- **Attacks** - This is where the actual attacks are
- **Process** - Includes what needs to be done per attack
- **Choices** - Adds a final possibility of choice

NOTE: Not all of these layers *have* to be used, except for the goals and attack layers. For example when the tree is relatively small (top layers can be left out) or when nodes become trivial before the choices layer is reached (bottom layers can be left out).

Splits

There are also four splits which are commonly used. These four do not include all splits that are available, therefore there is also a custom split included:

- **Components** - Splits a system into smaller components (example node **0**)
- **Phys/digi** - Splits attacks between physical and digital attacks (example node **2**)
- **FES** - Splits attacks between force, exploits and social engineering (example node **2**)
- **Steps** - Splits an attack into steps needed for success (example node **6**)
- **Custom** - Any other split not previously mentioned, tailored to the situation (example node **1**)

TIP: Try to use these splits as a source of inspiration when thinking of new attacks

Creating an attack tree - Discovery phase

First, a goal should be set. From there, the discovery phase begins. This phase fills the sub-goals, attack types and attacks layers (if they are used).
The goal in this phase is to discover as many different attacks as possible. This could be done in the form of a brainstorm, also multiple people joining together can result in more successful ideas and thought flows.

As mentioned before: try to make use of the layers, splits and the relations between them. They may give new inspiration and structure to your brainstorm.

This phase should **result** in a tree which at least contains the root and a layer of attacks. Furthermore there can be a sub-goals layer and attack types layer in between to organise the attacks.

Creating an attack tree - Detailing phase

Next, it is time to detail the attacks, i.e. expand on exactly how each attack should be executed. By doing this, the process and choices layer are filled (if they are used).
The attacks may be divided among system experts whom have explicit knowledge on how a specific attack could be executed. These can then be combined again in the attack tree.

When the details of each attack are added to the attack tree, this should **result** in a complete attack tree.

Appendix C

Basic Manual - Version 1

Manual for Attack Trees

Tim Sonderen
University of Twente & Nedap N.V.

Introduction

Attack trees are a tool to explore vulnerabilities in a system, be it physical, digital or both. The idea of creating an attack tree is for the security analysts and/or system experts to look at the system from the attackers' side. Because of the nature of attack trees, most information is gained by the creation of the attack tree itself. A small example can be found in the example block.

Goals

The goal of building an attack tree is to **explore attacks** on a system and expose vulnerabilities.
Therefore the root (first node) of an attack tree is a goal an attacker would have, e.g. 'Steal valuables'.

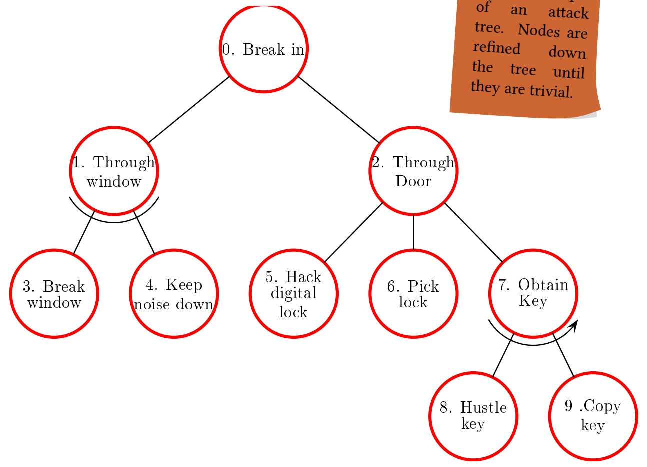
Basic building blocks

An attack tree consists of the following entities:

- **Root** node - The goal of the attack and start of the attack tree (example node **0**)
- **OR** nodes - A node of which only one of its child nodes needs to be successful (example node **2**)
- **AND** nodes - A node of which all of its child nodes need to be successful (example node **1**)
- **SAND** nodes - Like an AND node, but the child nodes are done in order (example node **7**)
- **LEAF** nodes - Leaves at the bottom of the attack tree, trivial attacks (example node **3**)

Creating an attack tree

After the root is set, the rest of the tree can be created by refining each node until the action in the node becomes trivial. This can also be seen in the example.



A small example of an attack tree. Nodes are refined down the tree until they are trivial.

Appendix D

Manual for Attack Trees - Version 2

Manual for Attack Trees

Tim Sonderen
University of Twente & Nedap N.V.

Introduction

Attack trees are a tool to explore vulnerabilities in a system, be it physical, digital or both. The idea of creating an attack tree is for the security analysts and/or system experts to look at the system from the attackers' side. Because of the nature of attack trees, most information is gained by the creation of the attack tree itself. A small example of three nodes can be found in the example block. A larger example can be found on the back.

Goals

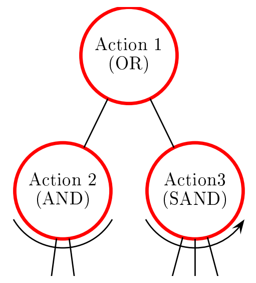
The goal of building an attack tree is to **explore attacks** on a system and expose vulnerabilities.
Therefore the root (first node) of an attack tree is a goal an attacker would have, e.g. 'Steal valuables'.

Basic building blocks

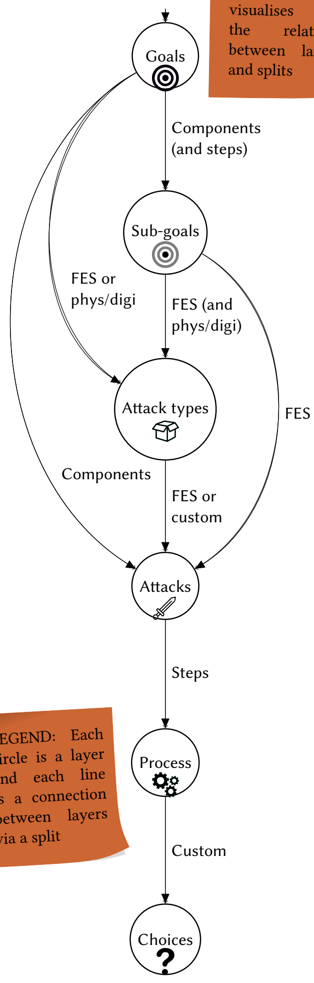
An attack tree consists of the following parts:

- **Root** node - The goal of the attack and start of the attack tree
- **OR** nodes - A node of which only one of its child nodes needs to be successful
- **AND** nodes - A node of which all of its child nodes need to be successful
- **SAND** nodes - Like an AND node, but the order of the child nodes matters
- **LEAF** nodes - Leaves at the bottom of the attack tree, trivial attacks

Example



Note: A node can have any number of child nodes and the type of node is indicated by the line below, you don't need to add the type between brackets.



This model visualises the relations between layers and splits

LEGEND: Each circle is a layer and each line is a connection between layers via a split

Creating an attack tree

In the blocks above, you can see the blocks you can use to build your own attack tree. How you fill them in is entirely up to you. After the root is set, the rest of the *tree* can be created by refining each node until the nodes become trivial (Leaf node).
To give this creation process more detail, the model on the left was developed, together with a number of layers and splits (see respective blocks below).
Again, the analysis will result in a tree structure, so it will get wider as you go down. In the model, each circle represents one layer (often one row of nodes). The arrows between those layers represent splits, which splits a node from the first layer into multiple nodes in the second. You can use this model as a guide as to which split is often used in a certain layer. There is a standard line straight down, which includes all layers, however you could also skip certain layers, following a different path (curved arrows) in the model.
Furthermore, in the quick guide block, there are a number of questions. Each one corresponds to a layer and acts as a starter question for that layer.

Layers

NOTE: Some of these layers can be skipped if they are not needed

Attack trees can be divided in layers which are most often only one node deep. The following six layers are commonly used:

1. **Goals** - Is generally just the root node
2. **Sub-goals** - Groups attacks by sub system
3. **Attack types** - Groups attacks by type
4. **Attacks** - This is where the actual attacks are
5. **Process** - Includes what needs to be done per attack
6. **Choices** - Adds a final possibility of choice

Quick Guide

Be creative! Make it a brainstorm and try to think outside the box!

The following line of questioning can be used to start the most used form of an attack tree.

1. What would be the goal of an attack on the system?
2. What systems can be attacked to reach the goal?
3. What *type* of attacks can be performed on this part?
4. What attacks can be performed on this part?
5. What steps are required to execute such an attack?
6. What alternatives approaches are there to the step?

Splits

TIP: Try to use these splits as a source of inspiration when thinking of new attacks

These four splits are commonly used

- **Components** - Splits a system into smaller components
- **Phys/digi** - Splits attacks between physical and digital attacks
- **FES** - Orders attacks in force, exploits and social engineering
- **Steps** - Splits an attack into steps needed for success
- **Custom** - Any other split you make

Examples

These are some examples for the splits to make them more clear:

- **Components** - Attack server / Attack network
- **Phys/digi** - Physical attack / Digital attack
- **FES** - Break in / Exploit error / Pressure someone
- **Steps** - Get lockpicks / Pick lock
- **Custom** - Any other split you make

Creating an attack tree - Discovery phase

First, a goal should be set. From there, the discovery phase begins. This phase fills the sub-goals, attack types and attacks layers (if they are used).
The goal in this phase is to discover as many different attacks as possible. This could be done in the form of a brainstorm, also multiple people joining together can result in more successful ideas and thought flows.

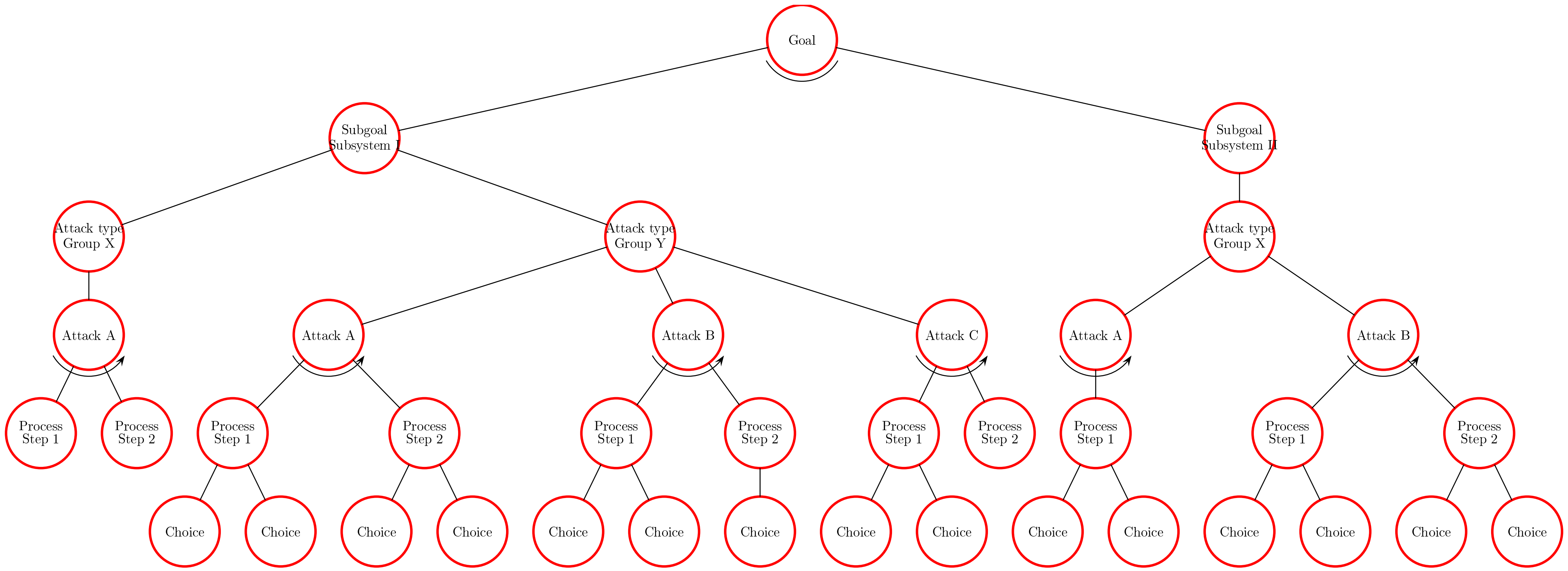
As mentioned before: try to make use of the layers, splits and the relations between them. They may give new inspiration and structure to your brainstorm.

This phase should **result** in a tree which at least contains the root and a layer of attacks. Furthermore there can be a sub-goals layer and attack types layer in between to organise the attacks.

Creating an attack tree - Detailing phase

Next, it is time to detail the attacks, i.e. expand on exactly how each attack should be executed. By doing this, the process and choices layer are filled (if they are used).
The attacks may be divided among system experts whom have explicit knowledge on how a specific attack could be executed. These can then be combined again in the attack tree.

When the details of each attack are added to the attack tree, this should **result** in a complete attack tree.


Goal
Subgoal Subsystem I
Subgoal Subsystem II
Attack type Group X
Attack type Group Y
Attack type Group X
Attack A
Attack A
Attack B
Attack C
Attack A
Attack B
Process Step 1
Process Step 2
Process Step 1
Process Step 2
Process Step 1
Process Step 2
Process Step 1
Process Step 2
Process Step 1
Process Step 1
Process Step 2
Choice
Choice
Choice
Choice
Choice
Choice
Choice
Choice
Choice
Choice
Choice
Choice
Choice
Choice
Choice

Appendix E

Basic Manual - Version 2

Manual for Attack Trees

Tim Sonderen
University of Twente & Nedap N.V.

Introduction

Attack trees are a tool to explore vulnerabilities in a system, be it physical, digital or both. The idea of creating an attack tree is for the security analysts and/or system experts to look at the system from the attackers' side. Because of the nature of attack trees, most information is gained by the creation of the attack tree itself. A small example can be found in the example block.

Goals

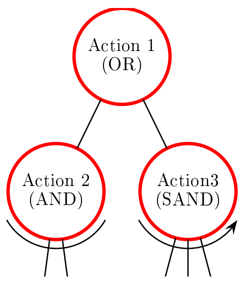
The goal of building an attack tree is to **explore attacks** on a system and expose vulnerabilities.
Therefore the root (first node) of an attack tree is a goal an attacker would have, e.g. 'Steal valuables'.

Basic building blocks

An attack tree consists of the following parts:

- **Root** node - The goal of the attack and start of the attack tree
- **OR** nodes - A node of which only one of its child nodes needs to be successful
- **AND** nodes - A node of which all of its child nodes need to be successful
- **SAND** nodes - Like an AND node, but the order of the child nodes matters
- **LEAF** nodes - Leaves at the bottom of the attack tree, trivial attacks

Nodes



Note: A node can have any number of child nodes

Creating an attack tree

In the blocks above, you can see the blocks you can use to build your own attack tree. How you fill them in is entirely up to you. After the root is set, the rest of the *tree* can be created by refining each node until the nodes become trivial (Leaf node).

LaTeX TikZposter

Bibliography

[1] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer, "DAG-based attack and defense modeling: Don't miss the forest for the attack trees", *Computer Science Review*, vol. 13-14, no. C, pp. 1–38, 2014, ISSN: 15740137. DOI: 10.1016/j.cosrev.2014.07.001. arXiv: 1303.7397. [Online]. Available: http://dx.doi.org/10.1016/j.cosrev.2014.07.001.

[2] J. D. Weiss, "A system security engineering process", *National Computer Security Conference*, vol. 2, no. 14, pp. 572–581, 1991. [Online]. Available: https://csrc.nist.gov/csrc/media/publications/conference-paper/1991/10/01/proceedings-14th-national-computer-security-conference-1991/documents/1991-14th-ncsc-proceedings-vol-2.pdf.

[3] B. Schneier, "Attack Trees", *Secrets and Lies*, pp. 318–333, 1999, ISSN: 1044-789X. DOI: 10.1002/9781119183631.ch21. [Online]. Available: http://doi.wiley.com/10.1002/9781119183631.ch21http://www.ddj.com/184411129https://www.schneier.com/academic/archives/1999/12/attack_trees.html.

[4] S. Mauw and M. Oostdijk, "Foundations of Attack Trees", in, 2006, pp. 186–198. DOI: 10.1007/11734727_17. [Online]. Available: https://www.researchgate.net/publication/225151465_Foundations_of_Attack_Treeshttp://link.springer.com/10.1007/11734727_17.

[5] J. N. Whitley, R. C. Phan, J. Wang, and D. J. Parish, "Attribution of attack trees", *Computers and Electrical Engineering*, vol. 37, no. 4, pp. 624–628, 2011, ISSN: 00457906. DOI: 10.1016/j.compeleceng.2011.04.010. [Online]. Available: http://dx.doi.org/10.1016/j.compeleceng.2011.04.010.

[6] T. R. Ingoldsby, "Attack Tree-based Threat Risk Analysis", PhD thesis, 2013. [Online]. Available: https://www.amenaza.com/downloads/docs/AttackTreeThreatRiskAnalysis.pdf.

[7] W. Pieters, D. Hadziosmanovic, A. Lenin, L. Montoya, and J. Willemson, "TREsPASS: Plug-and-Play Attacker Profiles for Security Risk Analysis", *35th IEEE Symposium on Security and Privacy*, no. 35, p. 4, 2014.

[8] S. Bistarelli, M. Dall'Aglio, and P. Peretti, "Strategic Games on Defense Trees", *Formal Aspects in Security and Trust*, pp. 1–15, 2005, ISSN: 03029743. DOI: 10.1007/978-3-540-75227-1_1. arXiv: arXiv:1011.1669v3. [Online]. Available: http://link.springer.com/10.1007/978-3-540-75227-1_1.

[9] S. Bistarelli, P. Peretti, and I. Trubitsyna, "Analyzing Security Scenarios Using Defence Trees and Answer Set Programming", *Electronic Notes in Theoretical Computer Science*, vol. 197, no. 2, pp. 121–129, 2008, ISSN: 15710661. DOI: 10.1016/j.entcs.2007.12.021. [Online]. Available: http://dx.doi.org/10.1016/j.entcs.2007.12.

021https://linkinghub.elsevier.com/retrieve/pii/S1571066108000601.

[10] K. Edge, G. Dalton, R. Raines, and R. Mills, “Using Attack and Protection Trees to Analyze Threats and Defenses to Homeland Security”, in *MILCOM 2006*, IEEE, 2006, pp. 1–7, ISBN: 1-4244-0617-X. DOI: 10.1109/MILCOM.2006.302512. [Online]. Available: http://ieeexplore.ieee.org/document/4086696/.

[11] S. A. Zonouz, H. Khurana, W. H. Sanders, and T. M. Yardley, “RRE: A game-theoretic intrusion Response and Recovery Engine”, in *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, IEEE, 2009, pp. 439–448, ISBN: 978-1-4244-4422-9. DOI: 10.1109/DSN.2009.5270307. [Online]. Available: http://ieeexplore.ieee.org/document/5270307/.

[12] A. Roy, D. S. Kim, and K. S. Trivedi, “Attack countermeasure trees (ACT): towards unifying the constructs of attack and defense trees”, *Security and Communication Networks*, vol. 5, no. 8, pp. 929–943, 2012, ISSN: 19390114. DOI: 10.1002/sec.299. [Online]. Available: http://doi.wiley.com/10.1002/sec.299.

[13] B. Kordy, S. Mauw, S. Radomirovic, and P. Schweitzer, “Attack-defense trees”, *Journal of Logic and Computation*, vol. 24, no. 1, pp. 55–87, 2014, ISSN: 0955-792X. DOI: 10.1093/logcom/exs029. [Online]. Available: https://academic.oup.com/logcom/article-lookup/doi/10.1093/logcom/exs029.

[14] B. Kaiser, P. Liggesmeyer, and O Mackel, “A New Component Concept for Fault Trees”, *Australian workshop on Safety critical systems and software*, vol. 33, no. 8, pp. 37–46, 2003.

[15] R. R. Yager, “OWA trees and their role in security modeling using attack trees”, *Information Sciences*, vol. 176, no. 20, pp. 2933–2959, 2006, ISSN: 00200255. DOI: 10.1016/j.ins.2005.08.004.

[16] A. Jürgenson and J. Willemson, “Serial model for attack tree computations”, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5984 LNCS, pp. 118–128, 2010, ISSN: 03029743. DOI: 10.1007/978-3-642-14423-3_9.

[17] P. A. Abdulla, J. Cederberg, and L. Kaati, “Analyzing the security in the GSM radio network using attack jungles”, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6415 LNCS, no. PART 1, pp. 60–74, 2010, ISSN: 03029743. DOI: 10.1007/978-3-642-16558-0_8.

[18] L. Piètre-Cambacédès and M. Bouissou, “Beyond Attack Trees: Dynamic Security Modeling with Boolean Logic Driven Markov Processes (BDMP)”, in *2010 European Dependable Computing Conference*, IEEE, 2010, pp. 199–208, ISBN: 978-1-4244-6593-4. DOI: 10.1109/EDCC.2010.32. [Online]. Available: http://ieeexplore.ieee.org/document/5474179/.

[19] N. Poolsappasit, R. Dewri, and I. Ray, “Dynamic security risk management using Bayesian attack graphs”, *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 1, pp. 61–74, 2012, ISSN: 15455971. DOI: 10.1109/TDSC.2011.34.

[20] R. Kumar, E. Ruijters, and M. Stoelinga, "Quantitative attack tree analysis via priced timed automata", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9268, pp. 156–171, 2015, ISSN: 16113349. DOI: 10.1007/978-3-319-22975-1_11. arXiv: arXiv:1408.6104v3. [Online]. Available: http://link.springer.com/10.1007/978-3-642-15297-9http://link.springer.com/10.1007/978-3-319-22975-1_11.

[21] R. Kumar and M. Stoelinga, "Quantitative security and safety analysis with attack-fault trees", *Proceedings of IEEE International Symposium on High Assurance Systems Engineering*, pp. 25–32, 2017, ISSN: 15302059. DOI: 10.1109/HASE.2017.12.

[22] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. Wing, "Automated generation and analysis of attack graphs", in *Proceedings 2002 IEEE Symposium on Security and Privacy*, IEEE Comput. Soc, 2002, pp. 273–284, ISBN: 0-7695-1543-6. DOI: 10.1109/SECPRI.2002.1004377. [Online]. Available: http://ieeexplore.ieee.org/document/1004377/.

[23] B. Kordy, P. Kordy, S. Mauw, and P. Schweitzer, "ADTool: Security analysis with attack-defense trees", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8054 LNCS, no. 318003, pp. 173–176, 2013, ISSN: 03029743. DOI: 10.1007/978-3-642-40196-1_15. arXiv: arXiv:1305.6829v2.

[24] B. Kordy, S. Mauw, and P. Schweitzer, "Quantitative questions on attack-defense trees", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7839 LNCS, pp. 49–64, 2013, ISSN: 03029743. DOI: 10.1007/978-3-642-37682-5_5. arXiv: 1210.8092.

[25] M. Audinot and S. Pinchinat, "On the soundness of attack trees", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9987 LNCS, pp. 25–38, 2016, ISSN: 16113349. DOI: 10.1007/978-3-319-46263-9_2.

[26] M. Audinot, S. Pinchinat, and B. Kordy, "Is my attack tree correct? Extended version", pp. 1–18, 2017. arXiv: 1706.08507. [Online]. Available: http://arxiv.org/abs/1706.08507.

[27] *The TREsPASS Project*, 2019. [Online]. Available: https://www.trespass-project.eu/ (visited on 02/21/2019).

[28] *Amenaza Technologies, SecurITree for Attack Tree analysis*, 2019. [Online]. Available: https://www.amenaza.com/ (visited on 02/21/2019).

[29] *Security and Trust of Software Systems*, 2019. [Online]. Available: http://satoss.uni.lu/ (visited on 02/21/2019).

[30] M. Fraile, M. Ford, O. Gadyatskaya, R. Kumar, M. Stoelinga, and R. Trujillo-Rasua, "Using attack-defense trees to analyze threats and countermeasures in an ATM: A case study", *Lecture Notes in Business Information Processing*, vol. 267, pp. 326–334, 2016, ISSN: 18651348. DOI: 10.1007/978-3-319-48393-1_24.

[31] J. Kramer, *Attack Tree Benchmarks*, 2019. [Online]. Available: https://www7.in.tum.de/~kraemerj/upload/index.php (visited on 04/05/2018).

[32] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research", *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007, ISSN: 0742-1222. DOI: 10.2753/MIS0742-1222240302. arXiv: z0022. [Online]. Available: https://www.tandfonline.com/doi/full/10.2753/MIS0742-1222240302.

[33] T. Torn, “Security Analysis of Estonian I-Voting System Using Attack Tree Methodologies”, 2014.

[34] S. Pudar, “A pragmatic method for integrated modeling of security attacks and countermeasures”, 2007. [Online]. Available: `http://lib.dr.iastate.edu/rtd/14567`.

[35] R. Jhawar, K. Lounis, S. Mauw, and Y. Ramírez-Cruz, “Semi-automatically Augmenting Attack Trees Using an Annotated Attack Tree Library”, in, 2018, pp. 85–101. DOI: `10.1007/978-3-030-01141-3_6`. [Online]. Available: `http://link.springer.com/10.1007/978-3-030-01141-3_6`.

[36] M. Warren, S. Leitch, and I. Rosewall, “Attack vectors against social networking systems: the Facebook example”, *Australian Information Security Management - Conference*, pp. 5–7, 2011. DOI: `10.4225/75/57b54ba6cd8cc`. [Online]. Available: `http://ro.ecu.edu.au/ism/131%0Ahttp://ro.ecu.edu.au/ism/131/`.

[37] M. A. Siddiqi, R. M. Seepers, M. Hamad, V. Prevelakis, and C. Strydis, “Attack-tree-based Threat Modeling of Medical Implants”, *International Workshop on Security Poofs for Embedded Systems (PROOFS)*, no. 7, pp. 32–49, 2018. DOI: `https://easychair.org/publications/paper/P43q`.

[38] H. K. Kong, M. K. Hong, and T. S. Kim, “Security risk assessment framework for smart car using the attack tree analysis”, *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, no. 3, pp. 531–551, 2018, ISSN: 18685145. DOI: `10.1007/s12652-016-0442-8`.

[39] M. Benini and S. Sicari, “Risk assessment in practice: A real case study”, *Computer Communications*, vol. 31, no. 15, pp. 3691–3699, 2008, ISSN: 01403664. DOI: `10.1016/j.comcom.2008.07.001`.

[40] C. S. Cho, W. H. Chung, and S. Y. Kuo, “Using Tree-Based Approaches to Analyze Dependability and Security on I&C Systems in Safety-Critical Systems”, *IEEE Systems Journal*, vol. 12, no. 2, pp. 1118–1128, 2018, ISSN: 19379234. DOI: `10.1109/JSYST.2016.2635681`.

[41] M. Gribaudo, M. Iacono, and S. Marrone, “Exploiting Bayesian Networks for the analysis of combined Attack Trees”, *Electronic Notes in Theoretical Computer Science*, vol. 310, pp. 91–111, 2015, ISSN: 15710661. DOI: `10.1016/j.entcs.2014.12.014`. [Online]. Available: `http://dx.doi.org/10.1016/j.entcs.2014.12.014`.

[42] F. Arnold, H. Hermanns, R. Pulungan, and M. Stoelinga, “Time-dependent analysis of attacks”, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8414 LNCS, pp. 285–305, 2014, ISSN: 16113349. DOI: `10.1007/978-3-642-54792-8_16`.

[43] A. Marback, H. Do, K. He, S. Kondamarri, and D. Xu, “Security test generation using threat trees”, *Proceedings of the 2009 ICSE Workshop on Automation of Software Test, AST 2009*, pp. 62–69, 2009. DOI: `10.1109/IWAST.2009.5069042`.

[44] J Zhong, S Du, L Zhou, H Zhu, F Cheng, C Chen, and Q Xue, “Security Modeling and Analysis on Intra Vehicular Network”, *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, pp. 1–5, 2017. DOI: `10.1109/VTCFall.2017.8288289`.

[45] P. Maynard, K. McLaughlin, and S. Sezer, “Modelling Duqu 2.0 Malware using Attack Trees with Sequential Conjunction”, pp. 465–472, 2016. DOI: `10.5220/0005745704650472`.

[46] A. Pretschner, J. Braun, M. Rumez, J. Dürrwang, and R. Kriesten, "Enhancement of Automotive Penetration Testing with Threat Analyses Results", *SAE International Journal of Transportation Cybersecurity and Privacy*, vol. 1, no. 2, 2018, ISSN: 2572-1054. DOI: 10.4271/11-01-02-0005.

[47] C. Fung, Y. L. Chen, X. Wang, J. Lee, R. Tarquini, M. Andersen, and R. Linger, "Survivability analysis of distributed systems using attack tree methodology", *Proceedings - IEEE Military Communications Conference MILCOM*, vol. 2005, pp. 1–7, 2005. DOI: 10.1109/MILCOM.2005.1605745.

[48] A. Roy, D. S. Kim, and K. S. Trivedi, "Cyber security analysis using attack countermeasure trees", p. 1, 2010. DOI: 10.1145/1852666.1852698.

[49] Opel Alexander, "Internship Thesis Design and Implementation of a Support Tool for Attack Trees", *Department of Mathematics and Computer Science at Eindhoven university*, 2005.

[50] R. Kumar, S. Schivo, E. Ruijters, B. M. Yildiz, D. Huistra, J. Brandt, A. Rensink, and M. Stoelinga, "Effective analysis of attack trees: A model-driven approach", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10802 LNCS, 2018, pp. 56–73, ISBN: 9783319893624. DOI: 10.1007/978-3-319-89363-1_4. [Online]. Available: http://link.springer.com/10.1007/978-3-319-89363-1_4.

[51] X. Qin and W. Lee, "Attack plan recognition and prediction using causal networks", *Proceedings - Annual Computer Security Applications Conference, ACSAC*, pp. 370–379, 2004, ISSN: 10639527. DOI: 10.1109/CSAC.2004.7.

[52] M. Sanford, D. Woodraska, and D. Xu, "Security analysis of FileZilla server using threat models", *SEKE 2011 - Proceedings of the 23rd International Conference on Software Engineering and Knowledge Engineering*, 2011.

[53] H. Suleiman and D. Svetinovic, "Evaluating the effectiveness of the security quality requirements engineering (SQUARE) method: A case study using smart grid advanced metering infrastructure", *Requirements Engineering*, vol. 18, no. 3, pp. 251–279, 2013, ISSN: 09473602. DOI: 10.1007/s00766-012-0153-4.

[54] A. Morais, A. Cavalli, and E. Martins, "A model-based attack injection approach for security validation", p. 103, 2011. DOI: 10.1145/2070425.2070443.

[55] O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle, and B. Weyl, "Security requirements for automotive on-board networks", *2009 9th International Conference on Intelligent Transport Systems Telecommunications, ITST 2009*, pp. 641–646, 2009. DOI: 10.1109/ITST.2009.5399279.

[56] A. P. Moore, R. J. Ellison, and R. C. Linger, *Attack Modeling for Information Survivability*, 2001.

[57] G. S. Parnell, D. L. Buckshaw, J. M. Wallner, W. L. Unkenholz, O. S. Saydjari, and D. L. Parks, "Mission Oriented Risk and Design Analysis of Critical Information Systems", *Military Operations Research*, vol. 10, no. 2, pp. 19–38, 2011, ISSN: 10825983. DOI: 10.5711/morj.10.2.19.

[58] G. Luca, "Critical Information Infrastructures Security", vol. 6712, no. May, pp. 7–8, 2011, ISSN: 03029743. DOI: 10.1007/978-3-642-21694-7. arXiv: 9780201398298. [Online]. Available: http://link.springer.com/10.1007/978-3-642-21694-7.

[59] A. Buoni, M. Fedrizzi, and J. Mezei, "A Delphi-based approach to fraud detection using attack trees and fuzzy numbers", no. November, 2016.

[60] B. Schneier, “Attack trees: modeling security threats”, *Dobb’s J. Softw. Tools*, vol. 24, p. 12, 2002. [Online]. Available: http://www.ddj.com/security/184414879.

[61] E. J. Byres, M. Franz, and D. Miller, “The Use of Attack Trees in Assessing Vulnerabilities in SCADA Systems”, *Critical Infrastructure Assurance Group, Cisco Systems Inc.*, 2004, ISSN: 05678315. DOI: 10.1.1.466.1887.

[62] C.-w. Ten, S. Member, G. Manimaran, and S. Member, “Cybersecurity for Critical Infrastructures: Attack and Defense Modeling”, *Ieee Transactions on Systems*, vol. 40, no. 4, pp. 853–865, 2010, ISSN: 10834427. DOI: 10.1109/TSMCA.2010.2048028.

[63] A. Bagnato, B. Kordy, P. H. Meland, and P. Schweitzer, “Attribute Decoration of Attack–Defense Trees”, *International Journal of Secure Software Engineering*, vol. 3, no. 2, pp. 1–35, 2012, ISSN: 1947-3036. DOI: 10.4018/jsse.2012040101. [Online]. Available: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jsse.2012040101.

[64] A. Morais, E. Martins, A. Cavalli, and W. Jimenez, “Security protocol testing using attack trees”, *Proceedings - 12th IEEE International Conference on Computational Science and Engineering, CSE 2009*, vol. 2, pp. 690–697, 2009. DOI: 10.1109/CSE.2009.206.

[65] S. McLaughlin, D. Podkuiko, and P. McDaniel, “Energy theft in the advanced metering infrastructure”, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6027 LNCS, pp. 176–187, 2010, ISSN: 03029743. DOI: 10.1007/978-3-642-14379-3_15.

[66] S. Bistarelli, F. Fioravanti, and P. Peretti, “Defense trees for economic evaluation of security investments”, *Proceedings - First International Conference on Availability, Reliability and Security, ARES 2006*, vol. 2006, pp. 416–423, 2006. DOI: 10.1109/ARES.2006.46.

[67] S. McLaughlin, D. Podkuiko, S. Miadzvezhanka, A. Delozier, and P. McDaniel, “Multi-vendor penetration testing in the advanced metering infrastructure”, p. 107, 2010. DOI: 10.1145/1920261.1920277.

[68] K. S. Trivedi, D. S. Kim, A. Roy, and D. Medhi, “Dependability and security models”, in *2009 7th International Workshop on Design of Reliable Communication Networks*, IEEE, 2009, pp. 11–20, ISBN: 978-1-4244-5047-3. DOI: 10.1109/DRCN.2009.5340029. [Online]. Available: http://ieeexplore.ieee.org/document/5340029/.

[69] M. H. Diallo, J. Romero-mariona, S. E. Sim, and D. J. Richardson, “A Comparative Evaluation of Three Approaches to Specifying Security Requirements”, *12th Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 2–7, 2006.

[70] S. Pudar, G. Manimaran, and C.-C. Liu, “PENET: A practical method and tool for integrated modeling of security attacks and countermeasures”, *Computers & Security*, vol. 28, no. 8, pp. 754–771, 2009, ISSN: 01674048. DOI: 10.1016/j.cose.2009.05.007. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167404809000522.

[71] S. Du, X. Li, J. Du, and H. Zhu, "An attack-and-defence game for security assessment in vehicular ad hoc networks", *Peer-to-Peer Networking and Applications*, vol. 7, no. 3, pp. 215–228, 2014, ISSN: 1936-6442. DOI: 10.1007/s12083-012-0127-9. [Online]. Available: http://link.springer.com/10.1007/s12083-012-0127-9.

[72] V. Gandotra, A. Singhal, and P. Bedi, "Identifying Security Requirements Hybrid Technique", in *2009 Fourth International Conference on Software Engineering Advances*, IEEE, 2009, pp. 407–412, ISBN: 978-1-4244-4779-4. DOI: 10.1109/ICSEA.2009.65. [Online]. Available: http://ieeexplore.ieee.org/document/5298889/.

[73] Y. H. Chang, P. Jirutitijaroen, and C.-w. Ten, "A simulation model of cyber threats for energy metering devices in a secondary distribution network", in *2010 5th International Conference on Critical Infrastructure (CRIS)*, IEEE, 2010, pp. 1–7, ISBN: 978-1-4244-8080-7. DOI: 10.1109/CRIS.2010.5617494. [Online]. Available: http://ieeexplore.ieee.org/document/5617494/.

[74] A. Buoni, M. Fedrizzi, and J. Mezei, "Combining attack trees and fuzzy numbers in a multi-agent approach to fraud detection", *International Journal of Electronic Business*, vol. 9, no. 3, p. 186, 2011, ISSN: 1470-6067. DOI: 10.1504/IJEB.2011.042541. [Online]. Available: http://www.inderscience.com/link.php?id=42541.

[75] K. Lounis, "Stochastic-based Semantics Of Attack-Defense Trees For Security Assessment", *Electronic Notes in Theoretical Computer Science*, vol. 337, no. 318003, pp. 135–154, 2018, ISSN: 15710661. DOI: 10.1016/j.entcs.2018.03.038. [Online]. Available: https://doi.org/10.1016/j.entcs.2018.03.038.

[76] B. Kordy, M. Pouly, and P. Schweitzer, "Probabilistic reasoning with graphical security models", *Information Sciences*, vol. 342, pp. 111–131, 2016, ISSN: 00200255. DOI: 10.1016/j.ins.2016.01.010. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2016.01.010.

[77] F. Arnold, D. Guck, R. Kumar, and M. Stoelinga, "Sequential and Parallel Attack Tree Modelling", in, ser. Lecture Notes in Computer Science, M. Felici and K. Kanoun, Eds., vol. 1698, Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 291–299, ISBN: 978-3-540-66488-8. DOI: 10.1007/978-3-319-24249-1_25. [Online]. Available: http://link.springer.com/10.1007/978-3-319-24249-1_25.

[78] S. Paul and R. Vignon-Davillier, "Unifying traditional risk assessment approaches with attack trees", *Journal of Information Security and Applications*, vol. 19, no. 3, pp. 165–181, 2014, ISSN: 22142126. DOI: 10.1016/j.jisa.2014.03.006.

[79] R. Jhawar, K. Lounis, and S. Mauw, "A Stochastic Framework for Quantitative Analysis of Attack-Defense Trees", in, ser. Lecture Notes in Computer Science 318003, R. Accorsi and S. Ranise, Eds., vol. 8203, Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 138–153, ISBN: 978-3-642-41097-0. DOI: 10.1007/978-3-319-46598-2_10. [Online]. Available: http://link.springer.com/10.1007/978-3-642-41098-7http://link.springer.com/10.1007/978-3-319-46598-2_10.

[80] A. Bobbio, L. Egidi, and R. Terruggia, "A methodology for qualitative/quantitative analysis of weighted attack trees", *IFAC Proceedings Volumes*, vol. 46, no. 22, pp. 133–138, 2013, ISSN: 14746670. DOI: `10.3182/20130904-3-UK-4041.00007`. [Online]. Available: `http://dx.doi.org/10.3182/20130904-3-UK-4041.00007https://linkinghub.elsevier.com/retrieve/pii/S1474667015340003`.

[81] M. Cheah, S. A. Shaikh, J. Bryans, and P. Wooderson, "Building an automotive security assurance case using systematic security evaluations", *Computers & Security*, vol. 77, pp. 360–379, 2018, ISSN: 01674048. DOI: `10.1016/j.cose.2018.04.008`. [Online]. Available: `https://linkinghub.elsevier.com/retrieve/pii/S0167404818303584`.

[82] J. L. Bayuk, "Security as a theoretical attribute construct", *Computers and Security*, vol. 37, pp. 155–175, 2013, ISSN: 01674048. DOI: `10.1016/j.cose.2013.03.006`. [Online]. Available: `http://dx.doi.org/10.1016/j.cose.2013.03.006`.

[83] D. Baca and K. Petersen, "Countermeasure graphs for software security risk assessment: An action research", *Journal of Systems and Software*, vol. 86, no. 9, pp. 2411–2428, 2013, ISSN: 01641212. DOI: `10.1016/j.jss.2013.04.023`. [Online]. Available: `http://dx.doi.org/10.1016/j.jss.2013.04.023`.

[84] N. Shahmehri, A. Mammar, E. Montes De Oca, D. Byers, A. Cavalli, S. Ardi, and W. Jimenez, "An advanced approach for modeling and detecting software vulnerabilities", *Information and Software Technology*, vol. 54, no. 9, pp. 997–1013, 2012, ISSN: 09505849. DOI: `10.1016/j.infsof.2012.03.004`. [Online]. Available: `http://dx.doi.org/10.1016/j.infsof.2012.03.004`.

[85] K. F. Cross and R. L. Lynch, "The "SMART" way to define and sustain success", *National Productivity Review*, vol. 8, no. 1, pp. 23–33, 1988, ISSN: 02778556. DOI: `10.1002/npr.4040080105`. [Online]. Available: `http://doi.wiley.com/10.1002/npr.4040080105`.

[86] D. Wichers and J. Williams, *OWASP*, 2017. [Online]. Available: `https://www.owasp.org/` (visited on 06/24/2019).