ABSTRACT
This paper explores the idea of visualizing attack trees by using circle packing; a tree visualization method which looks like a treemap but with circles. A proposal and proof of concept is presented to show how a basic attack tree can be visualized in this new way. And lastly, options to extend this visualization to support more kinds of different attack trees are explored.

Keywords
Attack trees, visualization, circle packing

1. INTRODUCTION
The demand for better cyber security is rapidly increasing. A lot of tools have been created in the last couple of years to help analyze and build effective measures to counter cyber attacks. A major step was made when attack trees[10] were introduced. The idea originated from fault trees. It uses roughly the same principles to provide a method to model attack paths to a given valuable asset.

1.1 Attack trees
In attack trees, the root node of the tree is often seen as an asset of value. Its children then describe how this asset can be compromised. This is demonstrated in Figure 1.

In this figure the circles are the nodes of the tree and the lines show a relationship. The root node is always at the top. In this case the root node is "Open safe". To open the safe, according to this figure, an attacker has to either pick the lock, learn the combo or cut open the safe. To learn the combo of the safe, the attacker has to either find a written combo or get the combo from a target.

This figure uses only disjunctive nodes, meaning that at every node, only one of its children is enough to compromise this node. On the other hand, a node can be conjunctive, meaning that all of its children are needed in order to compromise this node. An example of this is shown in Figure 2. This figure is almost the same as Figure 1, except for the fact that there are two nodes added which show what needs to be done in order to compromise the node "Get combo from target". The lines below this node are connected with an arc-shaped line, meaning that this node is conjunctive: In order to get the combo from a target the attacker has to both listen to the conversation and get the target to state the combo, one of those actions is not enough.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Figure 1. Example of an attack tree with only disjunctive nodes.

Figure 2. Example of an attack tree with both disjunctive and conjunctive nodes.
Figure 3. Example of an attack tree which is hard to read.

or a valuable asset which could be compromised.

1.2 Attack tree visualization
Most of the time, attack trees have been drawn similar to the structure they are based on: As a tree. This is great for understanding the data and editing it, but if the tree increases in size it becomes harder to draw conclusions from it without reading and understanding the complete diagram. This makes it harder for this kind of attack analysis to be understood by people with no in-depth knowledge about attack trees. Another property of this kind of visualization is that it is an explicit visualization method. This means that it displays the nodes and edges of the structure as they are. This gives the possibility of a very large visualization which could make it potentially hard to read. Figure 3 shows that the diagram becomes hard to read very quickly.

It is thus interesting to visualize attack trees in an implicit way. This way the structure is visualized in a space-filling way, which means that it is made to fit in a specific space and adapt according to the space available. Research has been conducted about methods to visualize attack trees in different ways [9, 6], but they do not cover visualization using circle packing.

1.3 Visualization by circle packing
Visualization by circle packing[12, 11] is such an implicit visualization method for tree structures, and an interesting way of visualizing because it makes it easier for the user to see groupings and structural relationships[11] compared to other implicit methods because it has the ability to display the whole hierarchy at once. It is very similar to a treemap[7] visualization, but it uses circles instead of squares.

This visualization method works as follows: The root node of a tree is drawn as a circle. This circle is drawn in such a way that is uses up the whole available space. Its children are then drawn within this circle in such a way that they fill up as much of the area of its parent as possible. This is then repeated until all the leaf nodes are drawn. This means that every tree, independent of its size, can be drawn within a specific area. The downside of this is the fact that for very large trees some nodes could be drawn so small that it is not possible to read them properly anymore. An example of this visualization method is shown in Figure 4.

There is, however, a huge difference between a digital and a printed version of this visualization: When this visualization is made digital, there are for example possibilities to zoom, in a smart way. For example: if a user clicks on a node, the visualization could zoom itself in to cover the whole available space with this specific node. This makes the children of this node better visible. The user could then zoom out again to a parent node.

In theory it should be possible to visualize attack trees with circle packing, but this requires a closer inspection. This paper explores this idea and proposes how this could be done in practice. In short:

Figure 4. Example of circle packing.

RQ 1 How can attack trees be visualized using circle packing?

2. METHODOLOGY
First, a method of visualizing attack trees using circle packing is proposed. Then, a proof of concept is built and shown. Finally, possibilities for expanding this visualization to show more information / properties will be shown. This could be used to extend this visualization to show data present in different kinds of attack trees.

3. COMBINING ATTACK TREES WITH CIRCLE PACKING
The basic algorithm to visualize attack trees with circle packing is fairly simple because the circle packing visualization is made to visualize tree structures. This algorithm will be as follows:

1. Draw the root node as a circle covering as much space of the available space as possible.
2. Draw the children of this node inside the circle of this node as circles, filling up as much space as possible, while making the leaf nodes all equal in size.
3. Repeat step 2 for every non-leaf node until all leaf nodes are drawn.

This way, there is one property not yet shown: If the nodes are conjunctive or disjunctive. This is done by giving disjunctive node circles a thin border and conjunctive node circles a thicker border. This communicates the idea that the nodes with thick borders have a special meaning, namely that all nodes inside have to be compromised in order to compromise this node.

Using these simple principles, the diagram shown in Figure 3 will look like shown in Figure 5. In this visualization, the red circles are the leaf nodes of the tree. The blue circles represent the non-leaf nodes of the tree. The largest blue circle represents the root node.

4. PROOF OF CONCEPT
In order to build a proof of concept which will be easy to reuse and build extensions upon, it is a good idea to choose
a well known drawing library which can easily be implemented by a lot of applications. For this, D3js [2] has been chosen because it is a well known drawing library for web in JavaScript. Because it uses JavaScript and HTML, it can be implemented in any kind of web application. It also means that with just JavaScript and CSS the whole visualization could be changed and adapted to specific needs. The advantage of this is that is is lightweight, almost every platform supports it and it is easy to develop with.

Mike Bostock[3] built two examples[4, 5] of circle packing with D3js. Those are very useful, because they convert a JSON file, containing a tree structure, to a spectacular visualization. This can then be adapted to support all kinds of features needed for specific attack tree visualizations. Those examples have been used to provide the images in this paper.

The JSON file needed for the visualization is quite straightforward and can easily be extended. To demonstrate this, a small tool in Python has been created to convert an XML file exported from ADTool[1] (which is a well known tool to draw attack trees with) to a JSON format (See appendix A). This means that the data from ADTool can easily be parsed into this visualization. This could make it fairly easy to visualize attack trees created in ADTool.

Figure 5 shows an example of an adaption of Mike’s work to visualize an attack tree which has been exported from ADTool. The outer circle represents the root node. When a circle is drawn within another circle, it shows a child-parent relation. The blue circles represent non-leaf nodes and the red circles represent the leaf nodes. The leaf nodes are drawn equally in size.

5. EXTENSIONS FOR VISUALIZATION

There are many extensions and adaptions of the classic attack trees[8]. They all have their own properties connected to nodes and/or work a little bit differently. These properties include but are not limited to: the cost of compromising a node, the difficulty of compromising a node, the time it takes to compromise a node, how a node is defended and how these defences could be broken by an attacker. This section provides an overview of properties of the visualization which could be modified to communicate more properties and thus visualizing and communicating more information to the user.

5.1 Text in circles

The text in the circles can display a lot of extra properties. The properties do have to be quite small to display, otherwise the visualization gets unreadable quite fast. It is also possible to give text different properties like making it bold, making it italic, underline it, strike it through, change the color, change the shadow, change the size. This could communicate extra information to the user.

5.2 Size of circles

The size is an important property within the visualization, because it is possible to change the complete look and feel of the visualization. For example, Figure 4 uses different sizes of leaf nodes to communicate a difference in importance. Figure 5 uses the same (small) size for every leaf node, which looks completely different. To use the full potential of this visualization, it is a good idea to give a size to the nodes according to a property they have. This could be for example the cost to compromise this node, the difficulty of compromising this node or the time it takes to compromise this node.

5.3 Color of circles

Color plays an important role in visualization because it can communicate ideas and feelings with just this property alone. It is important to select clear colors for the circles such that the text is still readable. When used correctly, colors can be used to great extend to visualize a property, for example shades could be used to indicate the difficulty to compromise this node, the time needed to compromise this node or how good the defenses are for this node.

5.4 Borders of circles

If the border color is different from the color of a circle, the border can also communicate extra information. This can be done using the border color and the border thickness. For example, the border thickness could communicate how good or costly the defences are around a node. The color could for example provide information about the difficulty of compromising this node or how good the defenses are for this node.

5.5 External information

Not all information has to be provided within the visualization itself. It is also possible to provide information outside the visualization, for example as a table next to the visualization. For example, detailed information about a node could be provided as text to the right of the visualization to make it more clear and to remove the need to put (too) much text into the visualization. It is also possible to provide, for example, an explanation for the different shapes and colors in the visualization to make it more clear to the user what it represents.

5.6 Depth selector

Another suggestion is to use a depth selector. This means that the user of the visualization can choose how many layers of the attack tree are shown in the visualization, so the view becomes more relevant and less cluttered. This is useful, for example, when the attack tree has to be presented to someone who does not need to see all the atomic actions, but rather the big picture. This also gives the user more freedom to customize this visualization. When a depth selector is active, it should still be able to zoom in the view.
in to be able to see all nodes eventually, so no nodes are permanently hidden.

5.7 Example

An example of a visualization of an attack trees with circle packing and extensions is given in Figure 6. This image shows the root node and its children. To demonstrate the depth selector, only a single layer is now displayed. The circle with "Learn combo" can now be zoomed in upon to uncover its children. Costs have also been added for every node as a simple text, and the green borders represent the strength of defense for the particular node.

6. CONCLUSION & DISCUSSION

It is possible to visualize attack trees using circle packing with the described method. It has potential; it can visualize attack trees exported from ADTool and the visualization itself can be extended to support all kinds of adaptations of attack trees.

A big question remains: is it better than other visualizations of attack trees? This is hard to tell: For the main part it comes down to personal preference. However it has of course good and bad properties: It uses a fixed amount of space and it gives a faster overview of an attack tree, but it is harder to edit or work on the structure. But this kind of visualization does add something new and fresh to experiment further with to make it even easier to understand, present and share attack trees. This kind of research helps in making these kind of models more approachable both for experts and people new to attack trees, which is important in the process of closing the gap between the studies around attack trees and the use of them in daily life by corporations and organizations to make us even more secure.

7. FUTURE WORK

There are a couple of things interesting to research further about this specific visualization. It could for example be tested and tweaked so it is more in line with what users expect from it. Besides, there are a lot of different types of attack trees, which would all need a little adaption of this visualization to work. It would be interesting if standards could be made to visualize a lot of different types of attack trees at once. If this works correctly, these visualizations could be implemented in tools which work with attack trees, such as ADTool.

It is also important to test this specific visualization with its different target audiences: Security researchers, companies and people with no in-depth knowledge of attack modeling. This way it is tested if this visualization will indeed work in practice, or if it is in need of some adaptations.

Besides the circle packing visualization it is important to keep exploring new ways to visualize to make the structure or other interesting properties even more clear to the user. This could be done by experimenting with other types of known tree visualizations, or come up with completely new visualizations.

8. REFERENCES

APPENDIX

A. PYTHON SCRIPT TO CONVERT XML FROM ADTREE TO JSON FOR THE D3JS VISUALIZATION.

```python
#!/usr/bin/env python

import json, sys, argparse
from collections import defaultdict
from xml.etree import ElementTree as ET

def etree_to_dict(t):
    d = {t.tag: {} if t.attrib else None}
    children = list(t)
    if children:
        dd = defaultdict(list)
        for dc in map(etree_to_dict, children):
            for k, v in dc.items():
                dd[k].append(v)
        d[t.tag] = {k: v[0] if len(v) == 1 else v for k, v in dd.items()}
    if t.attrib:
        d[t.tag].update((('@' + k, v) for k, v in t.attrib.items()))
    if t.text:
        text = t.text.strip()
        if children or t.attrib:
            d[t.tag][ '#text' ] = text
        else:
            d[t.tag] = text
    return d

def xml_to_json(f):
    xml = ET.parse(f)
    root = etree_to_dict(xml.getroot()[0])['node']
    final = {}
    todo = [root]

    while len(todo) > 0:
        cur = todo.pop()
        if cur['@refinement'] == "conjunctive":
            cur['and'] = True
        del cur['@refinement']
        cur['name'] = cur['label']
        del cur['label']

        if 'node' in cur:
            cur['children'] = cur['node']
        del cur['node']

        if type(cur['children']) is list:
            todo.extend(cur['children'])
        else:
            todo.append(cur['children'])

    return json.dumps(root)

def main(arguments):
    parser = argparse.ArgumentParser(
        description=__doc__,
        formatter_class=argparse.RawDescriptionHelpFormatter)
    parser.add_argument('input', help="The XML input file.", type=argparse.FileType('r'))
    args = parser.parse_args(arguments)
    print(xml_to_json(args.input))

if __name__ == '__main__':
    sys.exit(main(sys.argv[1:]))
```