Scalable Fault Tree Construction and Analysis

Student: Will Dower
Advisor: Professor Avi Silberschatz

Abstract

Currently, the best way for a cloud-computing service such as Amazon Web Services or Microsoft Azure to identify likely points of failure in its network is to passively wait for something to break. In a network the size of Amazon's, for example, the only way the company might be made aware of the fact that five servers all depend on the same switch to reach the internet would be if that switch failed. Such companies could benefit from a tool that would allow them to assemble dependency graphs of their network components and use them to identify risk groups, or minimal sets of components which could take down the network if they failed simultaneously. Companies could use this information to strengthen their networks, and end users of cloud services could use it to analyze several cloud services and compare the relative risk in using them.

A vital step in the construction of such a tool is the assembly of the fault tree, a data structure that aims to represent the hierarchy of dependencies upon which a given service runs. My project will be to write code to generate a fault tree from a list of dependency data, and then identify the most prevalent risk groups in the tree.

Details

A fault tree is a tree model of a computer system with a root node which represents the status of the entire system and a node for each of the system’s components and their vulnerabilities, with logical AND and OR gates connecting each child node to its parents. An AND gate connecting child nodes to their parent indicates that each of the child nodes would have to fail to take down the parent node, while an OR gate indicates that any of the children failing would also fail the parent.
is a minimal set of vulnerabilities that must trigger in order to cause the root node to fail. A vulnerability can also have a *weight*, which is some number to indicate the relative likelihood that the vulnerability will occur. An example of a fault tree (without weights) can be seen in Figure 1.

In the above diagram, \{Vul 1\} is a risk group because if Vulnerability 1 triggers, the file system goes down and thus so does the entire data access service. \{Vul 3\} is also a risk group, since both of the service’s paths to the internet depend on a component that Vulnerability 3 knocks out. In other systems, there may be risk groups consisting of multiple components instead of just one.

The goal of the project is to find the biggest vulnerabilities in the system – that is, its highest-weighted risk groups.

**Implementation**

I will write code that will take in a list of dependency data and crunch it into a fault tree, and determine a relative weight for each vulnerability. Finding the minimal cuts in a graph like this is a classic NP-hard problem, but luckily, there is an open source project called Maxino that uses heuristic
approaches to solve satisfiability problems. Since a fault tree features logic gates, it can be expressed as a Boolean formula. For example, the Boolean formula for Path 1 in Figure 1 would be:

\[(\text{Vul } 3 \land (\text{Vul } 3 \land \text{Vul } 2)),\]

since Path 1 depends on the Agg. Switch 1 and DNS components, which are triggered by Vulnerabilities 3, and 3 or 2, respectively. Therefore, I will generate a Boolean formula for the fault tree, which can then be solved as a SAT problem by Maxino. Maxino accounts for node weights, as well, so the relative weights of the vulnerabilities can be factored in as well.

**Deliverables**

The fault tree analysis code will be my deliverable. The idea will be to produce ranked risk groups which can then be used to compare the merits of different cloud systems. If there is time, I may refine the methods by which the dependency data is originally gathered, as well.