How Secur*IT*ree Deals with Random Events

Secur*IT*ree is primarily an attack tree analysis tool. However, it has significant fault tree functionality as well. Even more notable is its ability to analyze situations that arise partly due to adversarial actions and partly due to random factors. Let's see if I can explain this.

Consider first the capabilistic situation. In most cases we do not know the probability of an attack. Seldom do we have statistics to guide us. Fundamentally, even if statistics exist they may not apply to our situation. The likelihood of a given adversary performing a specific attack depends on three factors:

1. The capabilities of the adversary (money, technical ability, time, opportunity, tolerance for discovery, etc.)
2. The goals of the adversary (i.e., will the attack provide them the benefit they seek)
3. The exploit requirements of the attack

These factors determine how likely (probable) it is that a given encounter (opportunity) between the adversary and the target will result in the attack being realized. If we are looking to understand the frequency of the attack, we also need to know the number of attackers in the threat agent pool, the number of targets competing for their attention, and some additional information about the type of attack. I won't go into that now, but Secur*IT*ree supports all of this.

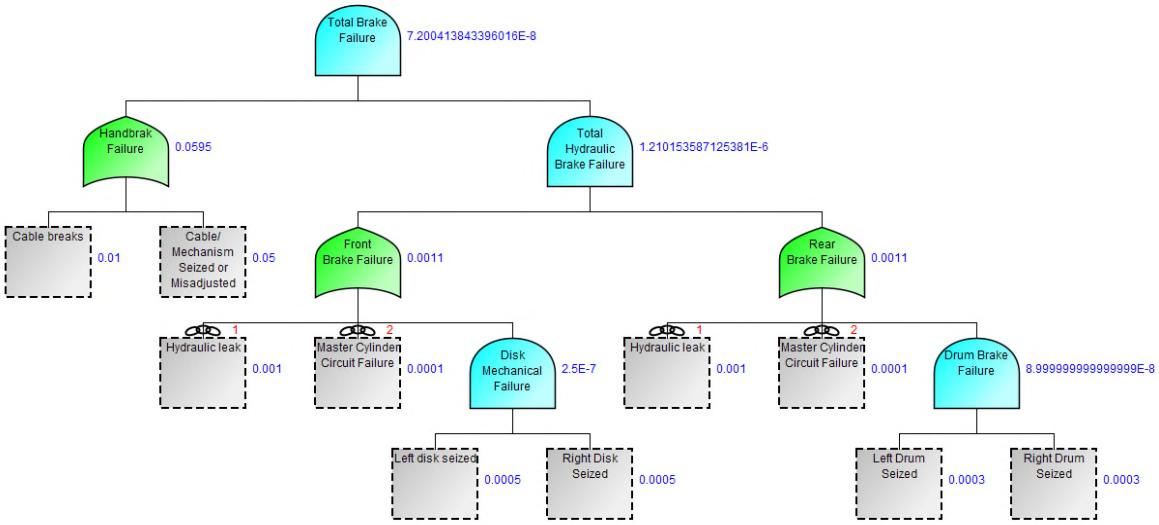
Secur*IT*ree supports two levels of capabilities-based analysis - basic and advanced. Basic is very simple and binary. You create a tree and define certain capability indicators (such as financial cost, technical difficulty). You populate the leaf node exploits with values representing the amount of each resource that needs to be expended to perform the exploit. Using aggregation formulas defined for each indicator Secur*IT*ree computes the resource costs of each scenario in the tree. Next, you create a profile of your threat agent. This profile defines the capabilities of the adversary. It is applied to the tree and nodes (and paths) pruned away if they are beyond the adversary's capabilities. A bit rudimentary but sufficient for quick and dirty analysis.

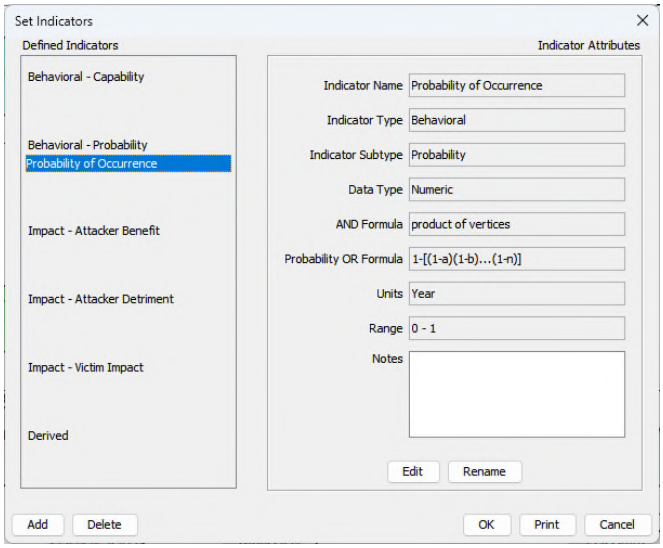
Advanced analysis is more sophisticated. It provides a more accurate depiction of the adversary. It recognizes that the adversary's willingness to expend resources is not binary. Rather it varies on a continuum. The adversary will be very willing to spend small amounts of a resource (compared to what the adversary possesses). They will be less willing to spend large amounts of the resource but may do so if the reward is sufficient. This requires the model to describe the benefits that an attack may accrue. In Secur*IT*ree , this is done by defining impact indicators reflecting attack benefits (money, confidential information, causing an outage). Both the willingness to expend each resource and the value the attacker puts on each benefit (there may be multiple) are described using utility curves. All of this is used to derive the probability of an attack scenario being realized in an encounter between the target and the adversary.

To understand risk requires an additional component to be introduced. Risk (to the defender) is the combination of the probability of the attack and the impact on the victim. Additional impact indicators can be defined reflecting the type of damage attacks will cause to the defender. A profile of the defender can be defined showing how they perceive given amounts of each type of

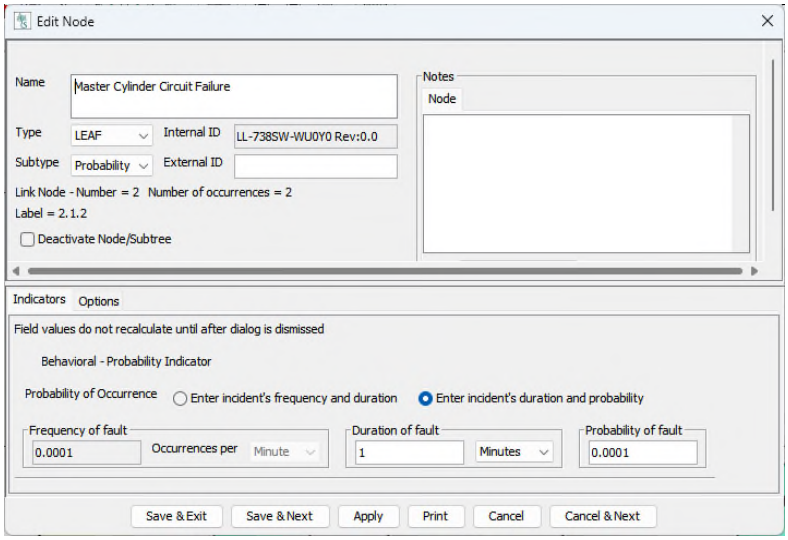
damage. Again, utility curves are defined for the defender translating raw amounts of damage into perceived impacts. Combined with the previous probability calculation it is now possible to assess the risk of an encounter between the adversary and target. If the additional information I mentioned earlier is also available, then (using frequency) it is possible to compute the cumulative risk - risk over a time period.

Now, what about fault tree analysis. Secur*ITree* also supports fault trees. Note that in a fault tree there is only a single indicator function determining the likelihood of a series of events. This indicator is probability of occurrence. Why only one indicator? Because probability (whether defined from basic principles (like tossing a dice) or from statistics) includes all of the factors that influence the event. An example I frequently use involves the total failure of the braking system on an automobile.





You may well be puzzled by the way Secur*IT*ree requires you to enter probability data at the leaf nodes.



Oddly, it seems concerned not just about how often the event occurs, but rather how long it lasts. It turns out that this is required to analyze a very interesting situation.

First, let me mention that fault trees differ from attack trees in some very fundamental ways. In a fault tree, the leaf level events are assumed to be statistically independent. In the model above, the likelihood of a brake cable breaking is completely unrelated to whether or not a hydraulic leak occurs in the master cylinder. The likelihood of both happening at the same time is the product of their individual probabilities.

This has several interesting consequences. It means that, in addition to the leaf level probabilities, it is also possible to directly calculate the probability of every scenario (called a cut set in fault tree parlance). In the attack tree, we had to infer the likelihood based on capability and desirability of the scenario. It is also possible to calculate the probability of intermediate nodes being reached in the fault tree and the overall probability of the root node being attained. To do this in an attack tree is much more involved and requires some other processes I haven't discussed.

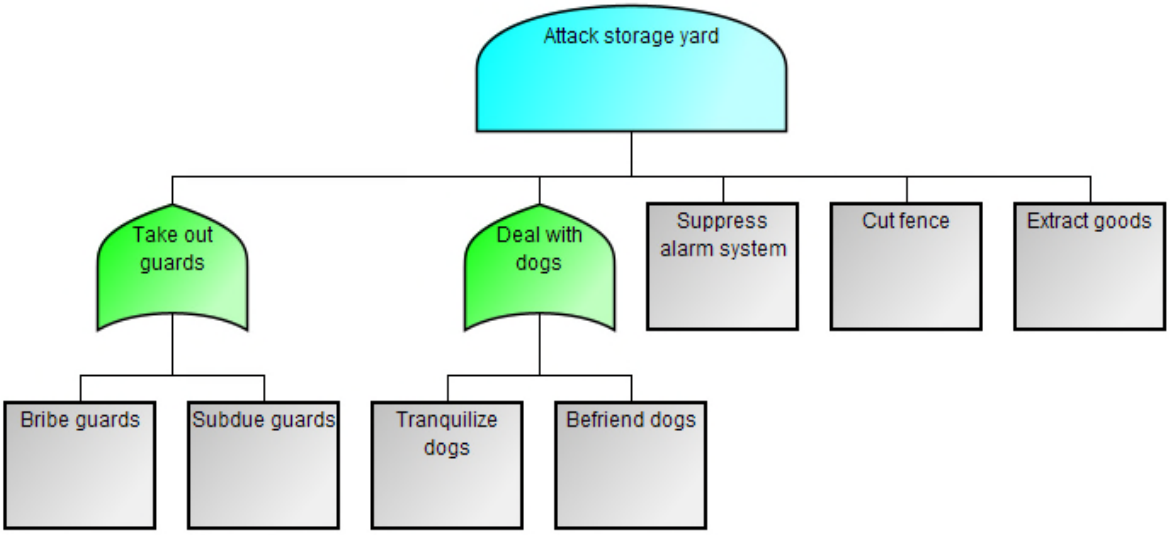
It should be noted that, in an attack tree, the leaf level events are not independent. In fact, they are highly interdependent. Whether or not a particular leaf node occurs may well depend on whether some other leaf node has been previously performed. Interestingly, a single leaf node may have several probabilities depending on the attack scenario that is being considered! This is very different from a fault tree.

So, back to the question of why Secur*IT*ree wants a duration value for probabilistic events.

In some cases attack scenarios may result from a combination of random events and attacker actions. Let me give a simple example involving physical security.

There is a factory located on the American Gulf coast that produces some type of widget. These widgets are stored in an outdoor yard that is protected by a chain link fence equipped with motion sensor detectors. Guards regularly patrol the perimeter. So, to get into the yard, an attacker must perform several tasks.

Some of these activities (taking out the guards and dogs, and suppressing the alarms) might be complicated, prone to detection and require significant resources. These steps are shown below.

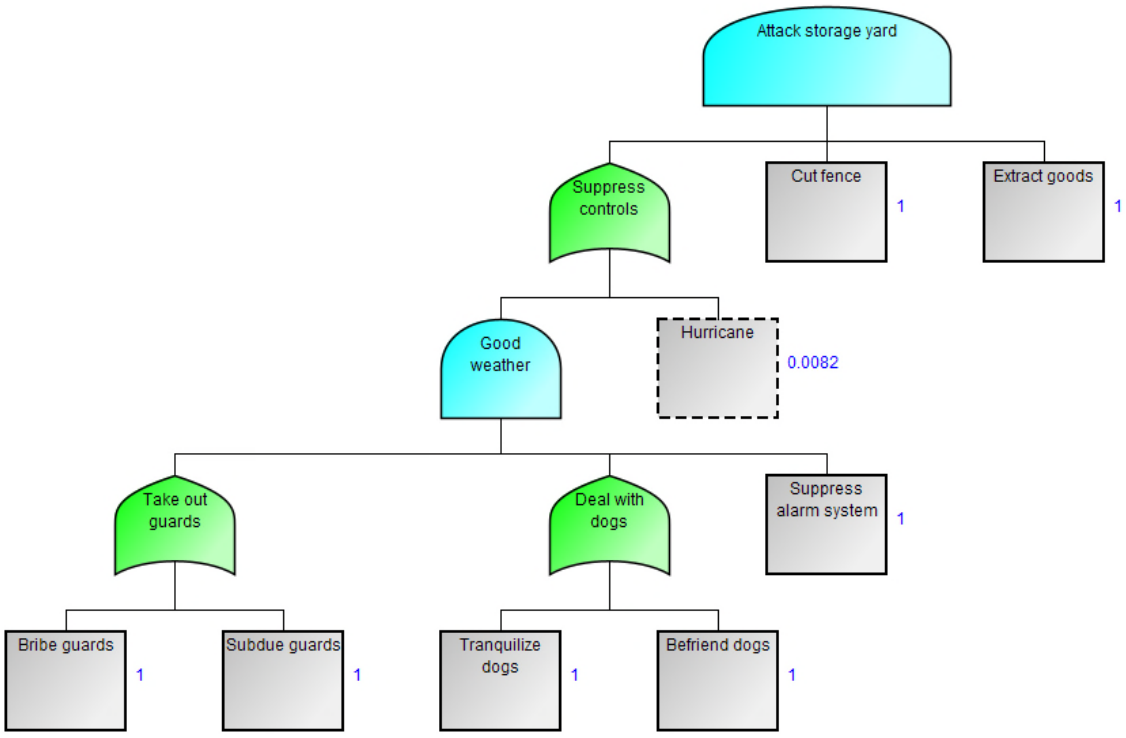


However, it turns out that, on average, for about three days per year, the New Orleans area is in the midst of a hurricane. During this type the guards huddle in their shacks, the dogs are cower in their kennels and the motion sensors are useless (triggering constantly). In this case, a brave (foolish ?) attacker might only need to cut the fence and extract the goods.

So, there are two cases (and probabilities). One for good weather. Another for hurricane weather. How do we analyze this for the hurricane situation?

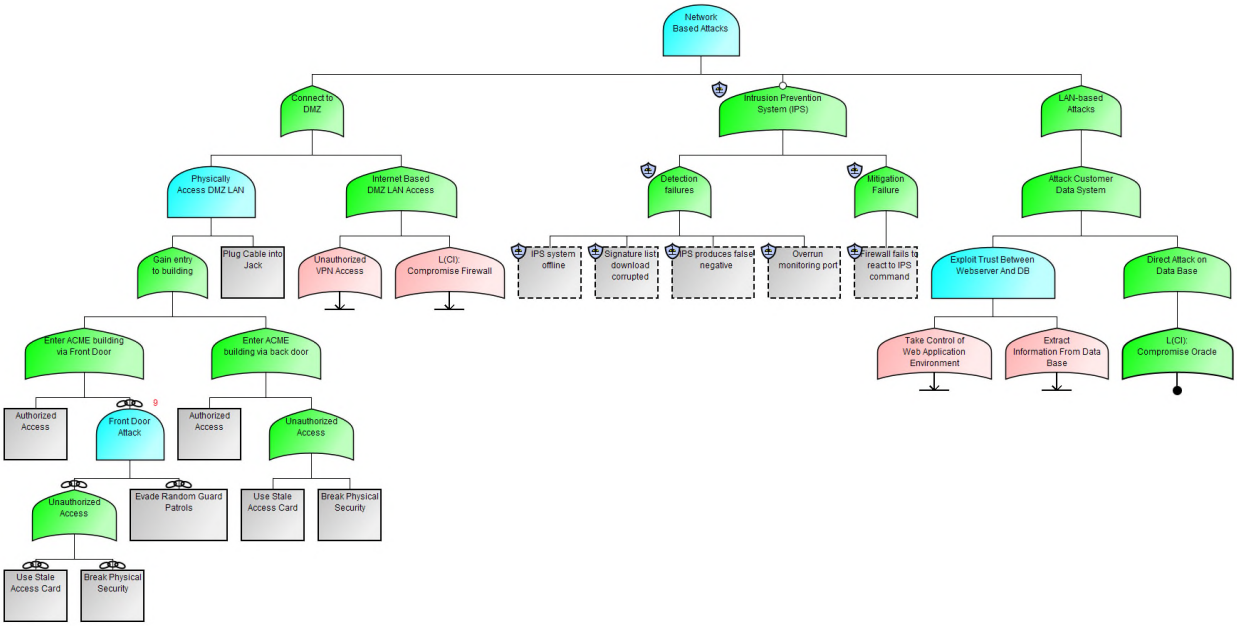
Let's pretend that New Orleans is always in a hurricane state. In that case, simple capabilities-based analysis will tell us what the likelihood would be for the attacker cutting the fence and extracting the goods (both simple, low resource activities). However, since we know that New Orleans is only in a hurricane situation for 3 days of the year (about 1% of the time), the overall average will be 1% of whatever the probability is if the hurricane were perpetual. Interestingly, it isn't the number of hurricanes that influences this but rather the fraction of time spent in the hurricane state that matters!

The whole situation can be depicted as



But, there are two more ways in which Secur*IT*ree deals with probability. The first builds on the probability work above. In some cases, a control or countermeasure can be thought of as a machine. Machines often fail due to random factors. Many years ago I worked with a first generation Intrusion Detection System (IDS). It was horrible. It ran on Windows NT. It often couldn't keep up with network traffic and missed packets. Worse, it tried to update its signatures a couple times a day. Occasionally the update would fail. You might think it would keep using the previous signatures - but oh, no - instead it just continued on with no signatures!

Secur*IT*ree allows the analyst to define a control that fails due to random factors. This really builds on the probability work shown earlier and requires that a probability indicator be defined and the top node of the control subtree has subtype = *Countermeasure*. Here is what that might look like.



Note the o on the top of the countermeasure tree. It conveys the idea that, when the IDS is working perfectly (100% uptime), it is impossible to get past the control. However, when it fails it can be bypassed.

And there is one more way in which probability can be defined in SecurITree. Sandia National Laboratory introduced the idea of attacker effectiveness. The idea is that, even when an attacker performs a step (leaf node or intermediate node) perfectly, sometimes it just doesn't work. For instance, buffer overflows often depend on a code entry point being at a certain location in memory. However, random factors may cause it to be somewhere else in a given case. SecurITree allows you to capture this.