Adapting Traditional Logic Modeling Techniques to Address Cyberattack

R. Youngblood[a*] and K. Le Blanc[a]
[a] Idaho National Laboratory, Idaho Falls, Idaho

Abstract: This paper is concerned with modeling the impact on systems of attacks that corrupt the flow of information in a system, especially control signals, with a view to taking steps to prevent the adverse impacts of such attacks. "Corrupting the flow of information" includes cyberattack, but is not limited to cyberattack. This scope also goes beyond hardware: it includes corrupting the information made available to operators, in order to induce them to contribute to undesirable plant outcomes. In order to clarify the method, we begin with a simple control-system problem first posed by Lapp and Powers. Even in this simple system, there are several ways to mislead the operators; one way is to induce them to take adverse control actions, and another is to conceal from them an adverse system evolution being induced by corruption of information flow. The paper describes (1) certain adaptations needed to appropriately refocus the application of logic modeling to this class of problems, (2) analysis and tests of attack scenarios, and (3) limitations that need to be addressed in future methodology development. We illustrate Top Event Prevention Analysis in the context of cyberattack; and finally, we discuss future work using a plant simulator.

Keywords: PRA, Cyberattack, Information Flow.

1. INTRODUCTION

This paper is concerned with modeling the impact of cyberattack on systems through corruption of the flow of information, especially control signals. This scope also includes corrupting the information made available to operators in order to produce undesirable plant outcomes.

Logic modeling is a tool for understanding how to satisfy certain complicated conditions. Fault-tree analysis is a special case of this: a fault tree is a picture of a set of logical relationships that relate a complex, system-level ("top") event (such as "failure of system X") to combinations of "basic" events (such as "Component P-1 fails AND component V-32 fails AND Tank-3 is empty"). Given the logical relationships in a fault tree, we can process the collection of relationships to obtain the "minimal cut sets" of the system: the list of combinations of basic events that are both necessary and sufficient to satisfy the condition "the top event occurs: the system fails." A minimal cut set is "sufficient" in the sense that occurrence of all of the events in the cut set is sufficient to cause the top event; it is "necessary" in the sense that if one element of a minimal cut set is removed from the cut set, the remaining elements are NOT sufficient to cause the top event.

Logic modeling was being used to understand satisfaction of complex logical conditions even before fault-tree analysis was invented. Nowadays, most of what is done in nuclear safety analysis is fault-tree analysis (perhaps in support of event-tree analysis); but more general applications of logic modeling are of interest. Below, we will use logic modeling in several ways: (1) to understand the conditions under which functional failure occurs, (2) to understand the conditions under which system damage occurs, (3) to understand the conditions under which the system state is successfully masked from the operators, and (4) to understand the conditions under which the system is adequately protected from cyberattack.

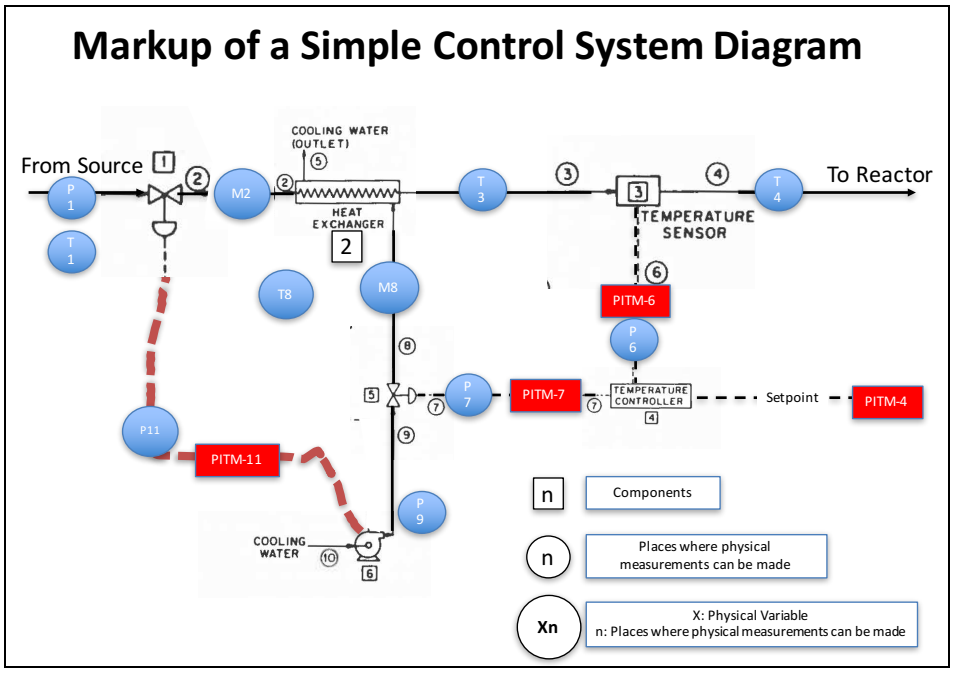
* robert.youngblood@inl.gov

2. EXAMPLE BASED ON SIMPLE CONTROL SYSTEM

For illustration, we begin with a simple problem first posed by Lapp and Powers [1], the execution of which allows us to clarify how to analyze cyberattack from a particular point of view. This simple system has functional failure modes that are distinct from "damage" failure modes, and even though the original problem did not explicitly involve operator action, we can analyze how to spoof operator-observers in order to conceal what is being done to the system, by taking a few liberties with the original problem formulation.

Figure 1 shows a markup of the Lapp-Powers example. The system is intended to regulate the temperature of HNO_3 delivered to the reactor via the path shown in the upper portion of the figure. In normal operation, hot HNO_3 enters at the top left and flows to the right through an isolation valve [1], a heat exchanger [2], and a sensor meant to read the temperature of the HNO_3 as it exits from the heat exchanger. The sensed temperature is fed to the controller [4], which compares it with an internal setpoint. If the temperature is high, the controller responds by causing valve [5] to admit more cooling water into the jacket of the heat exchanger; if the temperature is too low, then of course the reverse is meant to occur (cooling flow should be reduced). In order for this process to work, pump [6] must actually be working; so signal 11 is provided in order to confirm to valve [1] that [6] is working. If [6] is not working, signal 11 will tell [1] to close, since, otherwise, hot HNO_3 will be fed to the reactor, and the damage state will occur.

Figure 1: Markup of the system in the original Lapp-Powers example



In Figure 1, bold dotted lines correspond to information flowpaths addressed in the modeling. The "setpoint" line was only implicit in the original example. Each information flowpath is labelled "PITM-n" for "person in the middle-n," corresponding to a hypothetical intervention.

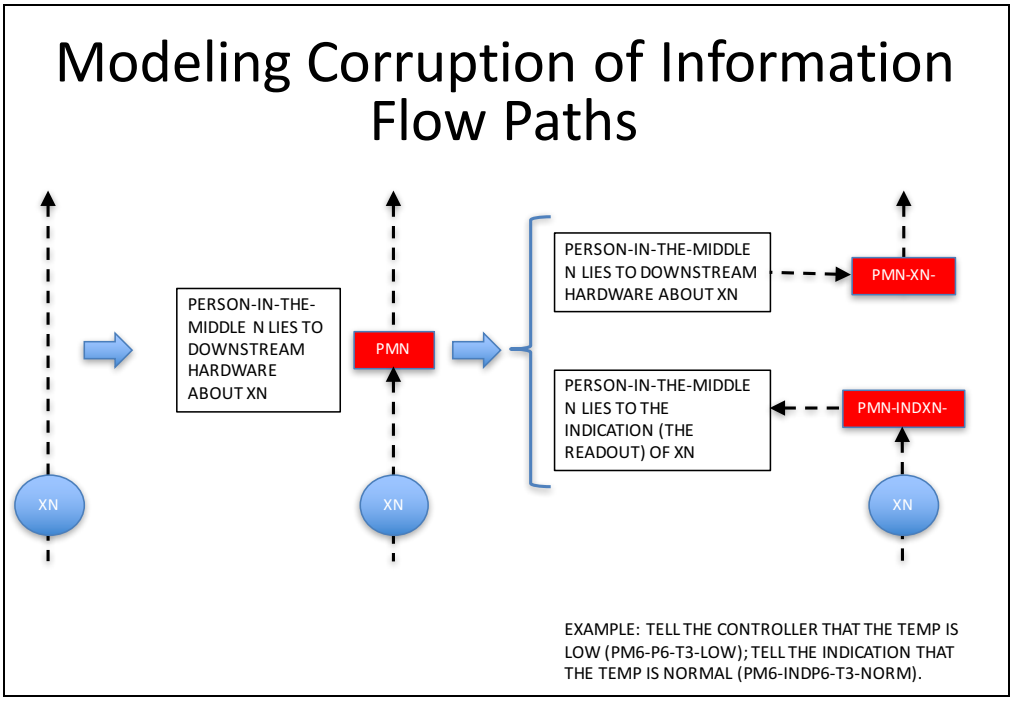
In the original Lapp-Powers example, some of the dotted lines were not necessarily electronic, and were certainly not wireless. They may have been things like air pressure, to control downstream air-operated valves. For present purposes, all dotted lines are treated analogously as carrying "information" and as being subject to intervention.

Here, "intervention" means some corruption of information flow somewhere on a dotted line, by a "person in the middle." The person in the middle can do either or both of two things (and perhaps others as well):

1. interfere with the information conveyed to the downstream component(s),
2. interfere with the information conveyed to indications available to operators.

These possibilities are suggested in Figure 2.

Figure 2. Corruption of Information Flowpaths



This diagram assumes that there are two classes of applications of the information: use by a downstream component or control system, and use in a control room indication.

Following are suggested steps for identifying conditions under which the "Person in the Middle" can cause damage AND conceal that fact from the indications available to the operator.

1. Identify all of the "Dotted Lines"
   Identify all the places on the system diagram where process control information is transmitted from one component to another.
   Each of these can either be in a "normal" state, or in a state where a person in the middle intervenes to tell the downstream component something other than what is correct.
   Identify places where information is transmitted to some kind of "indication" where information, correctly displayed, could cue the operators that their information is inconsistent, leading to thwarting of our sabotage. Some of these will be local indications, some will be control room indications, some will be indication of what signal is being sent from one component to another.
   Each of these can either be in a normal state, or in a state where a person in the middle intervenes to lie to the indication.
2. Model a "damage" state.
   In the Lapp-Powers example, the damage state is "flow is on AND T> Tmax."

3. Substitute in all the basic events, reduce the resulting expression, and condition on whatever events are to be presumed.
4. Model the conditions under which all of the indications will look "normal."
   This is an AND of OR expressions:
   (Indication 1 looks normal either because the sensed variable value actually is normal, OR because we are spoofing it) AND
   (Indication 2 looks normal, either because the sensed variable value actually is normal, OR because we are spoofing it) AND …
5. Form "damage state AND normal indications." Reduce. Get rid of complemented events.

Table 1 shows cut sets for the modified Lapp-Powers problem corresponding to "Flow at Temperature > Max Allowed Temperature," including at least some of what is needed in order to spoof the operator.

Table 1. Cut Sets Giving "Flow at Temperature > Max Allowed Temperature"

Failure or Causal Spoof	Indication Spoof	Indication Spoof	Indication Spoof
V5-INT	**PM6-IND-T3-NORM**		
(Failure) Coolant control valve just fails	PM indicates outlet temperature is normal		
PM7-V5-CL	**PM6-IND-T3-NORM**	**PM7-IND-P7-NORM**	
(Spoof) PM tells coolant control valve to close	PM indicates outlet temperature is normal	PM tells the observer that the control valve is getting a "normal" signal	
TC4-V5-CL	**PM6-IND-T3-NORM**	**PM7-IND-P7-NORM**	
(Failure) Temperature controller just tells control valve to close	PM indicates outlet temperature is normal	PM tells the observer that the control valve is getting a "normal" signal	
PM6-TC4-LO-T3	**PM6-IND-T3-NORM**	**PM7-IND-P7-NORM**	
(Spoof) PM tells the controller that outlet temperature is LOW	PM indicates outlet temperature is normal	PM tells the observer that the control valve is getting a "normal" signal	
SENS-3-LOW-T3	**PM6-IND-T3-NORM**	**PM7-IND-P7-NORM**	
(Failure) The Sensor reads a low outlet temperature	PM indicates outlet temperature is normal	PM tells the observer that the control valve is getting a "normal" signal	
PM4-TC4-STPT-HI	**PM4-IND-TC4-NORM**	**PM6-IND-T3-NORM**	**PM7-IND-P7-NORM**
(Spoof) PM feeds the temperature controller a high setpoint	PM indication to the observer is a normal setpoint	PM indicates outlet temperature is normal	PM tells the observer that the control valve is getting a "normal" signal

Notes:

1. The above is a tabulation of "conditions under which there is flow with T> Tmax, AND the indications available to operators are spoofed so as to conceal that fact."
2. In this simplified version of the model, Cooling Water Pump P11 is always on. If P11 is off, the system should shut down, and if there is no flow, then there is no flow at T>Tmax. We could wait for P11 to fail, and spoof the signal that is supposed to tell the shutoff valve to close, and spoof signals needed to hide the hot outlet temperature and the fact that the rest of the system is struggling to deal with it; or we could CAUSE P11 to fail, and spoof the other signals.
3. PM is always "person in the middle." The numbers on PM's in the cut sets indicate which dotted line is being interfered with by the PM. Note that there are two classes of PM's: Sometimes the PM's are lying to the downstream hardware, and sometimes they are providing false INDication to the operators. The latter PM events are all of the form "PMn-IND-… ." The former are of the form "PMn-[hardware]-… ."
4. There are four dotted lines to work with. In this simple model, one of them is always functioning normally. That leaves 3. There are 6 cut sets, 2 per dotted line, corresponding to 2 ways of messing with the 3 dotted lines: either a component failure, or a PM intervention, accompanied by "indication" spoofs needed to hide the intervention from the operators.

The steps are meant to search for ways in which a Saboteur could attack a particular system by interfering with signals (as opposed to directly (physically) tampering with system hardware component states). This is done for purposes of method refinement; the extension to consider a broader scope of Saboteur activity (also physically tinkering with components) and/or a broader scope of component failures occurring stochastically (as opposed to being due to physical or cyberattack) seems (at this point) straightforward in principle, and is discussed below.

3. TOP EVENT PREVENTION ANALYSIS

Top Event Prevention Analysis (here called "Prevention Analysis" for short) was formulated in the late 1980's [2, 3], based on prior work involving Boolean Optimization [4]. The original point of Prevention Analysis was to help choose a subset of all systems, structures, and components (SSCs) at a given facility, such that special treatment of just those SSCs would suffice as the backbone of the safety case for that facility.[*] More generally, given a cut set expression for a particular undesirable outcome (such as "system failure"), Prevention Analysis can be used to select a subset of basic events whose prevention most efficiently accomplishes prevention of the undesirable outcome. In many cases of practical interest, it is possible to choose a relatively small fraction of the overall population of basic events for special treatment, and still achieve a very high level of system reliability, even if the rest of the basic events are assumed to have high probability. Blanchard and co-workers [5, 6] have carried out many applications of Prevention Analysis, including application to vital area analysis, which is somewhat related to the present application.

Although the Lapp-Powers example discussed above has a control loop, it is actually too simple to be used to illustrate Prevention Analysis. A "flow loop" model that is slightly more complicated than the Lapp-Powers example has also been executed consistently with the above steps; that model will be presented separately. Here, we take the cut sets of the flow loop as given, and illustrate the process of preventing the top event.

As it turns out, each one of the 96 cut sets[†] in the flow-loop example contains 3 elements:

[*] In traditional ("deterministic") reactor licensing, this is done simply by identifying SSCs credited in the Safety Analysis, including the SSCs in the safety trains that are incapacitated by the postulated limiting failure. But Prevention Analysis was originally formulated for non-reactor facilities, for which a comparable Safety Analysis analog was then lacking, and it could be of significant benefit in future reactor licensing work.

[†] The reason some of them are highlighted in green is discussed below.

FLOW-LOOP-EX =

1	3	XX-5-A * XX-8-A * XX-11-A +
2	3	XX-5-A * XX-8-A * XX-10-A +
3	3	XX-5-A * XX-8-A * XX-9-A +
4	3	XX-1-A * XX-5-A * XX-8-A +
5	3	XX-5-A * XX-7-A * XX-11-A +
6	3	XX-5-A * XX-7-A * XX-10-A +
7	3	XX-5-A * XX-7-A * XX-9-A +
8	3	XX-1-A * XX-5-A * XX-7-A +
9	3	XX-5-A * XX-6-A * XX-11-A +
10	3	XX-5-A * XX-6-A * XX-10-A +…

In order to prevent the top event, we need to prevent every cut set. This can be accomplished by formulating the negation of the cut set expression to obtain the success paths; if every element of some success path succeeds, then that path succeeds, and the top event is prevented. However, for reasons discussed elsewhere, since "prevention" is not absolute, we may wish to prevent more than just one element in every cut set. In the present exercise, we wish to prevent at least two elements in every cut set, which is notionally analogous to applying a "single-failure" criterion. When the machinations needed to formulate Level 2 prevention are carried out for this example, we get a handful of prevention sets. The first is

Prevention Set 1:
XX-2-A, XX-3-A, X-4-A, XX-5-A, XX-6-A, XX-7-A, XX-8-A, XX-13-A, XX-14-A, XX-15-A, XX-16-A, XX-17-A, XX-18-A, XX-19-A.

The events highlighted in green appear in the first 10 of the FLOW-LOOP-EX cut sets. In fact, whatever prevention set we choose, there are always at least two elements of that prevention set in every cut set, which was the goal in this example ("Level 2 Prevention"). In fact, we can completely neglect every event in the top event expression that is not part of the chosen prevention set – we can even assume that they all occur – and we will still be protected against the top event by two events. In Prevention Set 1, we are protecting XX-5-A and XX-8-A, but not XX-9-A, X-10-A, or X-11-A; if we assume that the unprotected events (yellow in the markup of the cut set expression above) all occur, the first four cut sets in FLOW-LOOP-EX all collapse to XX-5-A*XX-8-A, with both of these events prevented.

The above discussion is written as if there were exactly one way to corrupt information flow to or from a given component (say, "C"), and that in a given prevention strategy, one either prevents that one corruption mode, or not. Suppose that in fact there are two ways: information flow to or from C can be compromised either remotely (via cyber) or locally (by an insider). Then, in order to achieve a given level of prevention, if we wish to protect C, we have to protect it *both* from cyberattack and from local sabotage. Algebraically, suppose we start with a cut set A*B*C, but identify a need to enhance it by considering both "remote" and "local" corruption. The cut set then becomes A*B*(C-R+C-L), where C-R is read as "C-Remote" and C-L is read as "C-Local." Table 1 below shows how this affects the Prevention Analysis.

In effect, the calculation is a bit like taking the complement of event C=C-R+C-L; we have

$$C=C\text{-}R+C\text{-}L$$
$$/C= /(C\text{-}R+C\text{-}L)=/C\text{-}R * /C\text{-}L.$$

That is, "Prevent-C" has become "Prevent-C-R **AND** Prevent-C-L."

Why bother with this elaboration? Suppose that each of A, B, C has hypothetical "local" and "remote" corruption possibilities. Then the cut set A*B*C maps into 2^3 cut sets, comprising all unique

combinations of remote and local corruption modes for each of A, B, C. Suppose further that the difficulties of "local" and "remote" corruption vary widely, but differently, for each of A, B, C. We may find that there is a fairly easy path to corruption of all three elements, involving a mixture of local and remote attacks, and steps should be taken to make those specific attacks harder. It is straightforward to generalize the above analysis to address this point by ranking prevention sets according to the effort needed to implement them. If all acts of prevention required the same effort, then the prevention sets could be ranked according to effort required, just by counting the number of prevention acts in each prevention set. In prevention analysis for pure internal events scenarios, this can work surprisingly well, for reasons discussed elsewhere; but if certain corruption events are essentially infeasible, this is valuable information. If certain local modes are essentially infeasible for the attackers, then their appearance in the prevention set is moot: yes, we need to be sure they are prevented, but they are inherently prevented, and we don't have to do anything.

Table 2: Prevention of both local and remote corruption events

Level 2 Prevention * = AND + = OR	
… of A*B*C	… of A*B*(C-L+C-R)=A*B*C-L+A*B*C-R
Requires:	Note that one cut set has become two, as the result of substituting C-L+C-R for C; so we need to prevent ***each*** of the two cut sets. This requires: (Prevent-A*Prevent-B+ Prevent-A* Prevent-C-L+ Prevent-B* Prevent-C-L) * (Prevent-A* Prevent-B+ Prevent-A* Prevent-C-R+ Prevent-B* Prevent-C-R)
Prevent-A * Prevent-B + Prevent-A * Prevent-C + Prevent-B * Prevent-C.	… which reduces to Prevent-A* Prevent-B+ Prevent-A*Prevent-C-L*Prevent-C-R + Prevent-B*Prevent-C-L*Prevent-C-R.

4. ACCOUNTING FOR THE OPERATORS

In the simple examples discussed above, the operators were treated at a high level of abstraction, and we did not try to cause the operator to take actions having adverse results. In the Lapp-Powers example, operators were presumed capable of realizing that the hypothetical indications were mutually inconsistent, unless the Saboteur went to some trouble to mask the intervention by spoofing the indications, but we did not explicitly consider cases where we spoofed enough indications to cause operators to cause problems.

A next step is to focus on operator response in the overall analysis. This will include cases where the attacks are meant to have adverse consequences, and we challenge the operators to recognize and thwart the attacks given valid indications, as well as cases in which the indications are also spoofed.
We will develop a series of scenarios that progressively increase the level of process knowledge and sophistication of the Saboteur, and study the capability of the human operator to detect and mitigate cyber manipulation. Systematically varying the sophistication with which information is manipulated will help the researchers to pinpoint when and if an operator is no longer able to utilize redundant indications and additional information to identify discrepancies between individual indicators and

system behavior. With this information, we can start to characterize how the operator contributes or mitigates risk due to cyber manipulation.

This will be done using the generic pressurized water reactor (gPWR) from GSE systems. Once the system has been modeled using the methodology described above, the system response and behavior will be verified by implementing the scenarios in a dynamic simulation.

5. CONCLUSION

5.1 Essential Characteristics of the Method

1. Model Information Flow

The example involved a control loop. For present purposes, control loops are of general interest, because even if a system does not have physical controllers (as ESTEC does not), a controller may exist virtually, if Operators are expected to respond to indications of system state. We have not yet illustrated the potential to trick the Operator into causing damage, but the potential appears to be there. We have illustrated a process by which a disturbance injected by the Saboteur at one point propagates through the control loop to accomplish the Saboteur's intent.

To put it in slightly different words, the example performed above is a search for conditions under which Saboteur intent could be accomplished through the corruption of "signals," understood in a general sense. By "signal," we mean the transmission of information about the system state to be used for some sort of control purpose. This could be information sent to valves, telling them to open or close; it could be information about setpoints; it could be information about sensed conditions, like temperature. It could be information meant to directly cause changes in hardware state, or it could be information meant to trick the Operator either into doing something that ought not to be done, or leaving undone something that ought to be done.

Essential Characteristic #1 is therefore that we should be exhaustively picking up, and reflecting, information paths.

2. Model Control Functions

Not all modern analyses are simple tank-to-vessel, flow-or-no-flow models; some involve more complicated modeling of control. But modeling of control is Essential Characteristic #2. This includes not only the physical control systems, but also the Operator.

3. Think More Broadly than "Failure"

Classical risk analysis methods are about "failure," an important simplification. In the example, we analyzed ways of causing "failure," ways of causing "damage," and ways of masking the ongoing perpetration of failure or damage. "Failure"-centric analyses will not necessarily pick up "damage" scenarios, or vice versa.

5.2 Limitations

As in many instances of classical PRA, the logic structure depends on assumptions made about plant behavior, assumptions that need to be based on quality simulation or on actual plant performance. This comment applies at multiple levels: in principle, we need simulation in order to understand the real effect of a given spoof on the physical plant: How long does it take the system to respond, given a spoof? How long does it take the indications to respond, given a spoof? How long does it take for "damage" to occur in reality? We also need simulation to understand how the plant's physical state manifests itself in the indications available to the operators. In this kind of application, doing a really good job of tricking the operator arguably requires knowing what indications should result from

particular system states. In the example above, we postulated the indications, and told the logic how to treat them; in a more sophisticated application, we would derive this information from considered simulation studies.

Corrupting information flow is not the only way to cyberattack. We have in hand an example of "damage" caused by corruption of the power supplied to large rotating machinery. If we could similarly cause degraded, but not absent, voltage to systems that rely on particular voltages at particular frequencies, we might be able to cause "damage" and/or "failure." Some instances of this may be causable by mechanisms appearing in the example, but others may not.

Acknowledgements

This work is supported through the INL Laboratory Directed Research & Development (LDRD) program under DOE Idaho Operations Office Contract DE-AC07-05ID14517.

References

[1] Lapp and Powers, Lapp S. A. and G. J. Powers, Computer-Aided Synthesis of Fault Tree, IEEE Transactions on Reliability, Vol. R-26, No. 1, pp. 2-12, Apr. 1977.
[2] R. Youngblood and L. Oliveira, "Application of an Allocation Methodology," Proceedings of "PSA '89 / International Topical Meeting / Probability, Reliability, and Safety Assessment," April 2-7, 1989, Pittsburgh, Pennsylvania (American Nuclear Society, Inc., La Grange Park, Illinois, 1989).
[3] R. W. Youngblood and R. B. Worrell, "Top Event Prevention in Complex Systems," Proceedings of the 1995 Joint ASME/JSME Pressure Vessels and Piping Conference, PVP-Vol. 296, SERA-Vol. 3, "Risk and Safety Assessments: Where Is The Balance?" July 1995 (The American Society of Mechanical Engineers, New York, New York 10017, 1995).
[4] D.D. Boozer and R.B. Worrell, 'A Method for Determining the Susceptibility of a Facility to Sensor System Nullification by Insiders,' SAND77-1916C, Feb. (1978).
[5] Risk-Informed Safety Margin Characterization Case Study: Use of Prevention Analysis in the Selection of Electrical Equipment to Be Subjected to Environmental Qualification, D. P. Blanchard and R. W. Youngblood, Proceedings of PSAM-12 (Probabilistic Safety Assessment and Management), 22-27 June 2014.
[6] R. B. Worrell and D. P. Blanchard, "Top Event Prevention Analysis to Eliminate Requirements Marginal to Safety," Proceedings of the 1995 Joint ASME/JSME Pressure Vessels and Piping Conference, June 1995.