Integration of Graphical Modeling Techniques as a Structural Framework for System-Aware Cyber Security Architecture Selection

---

A Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment

of the requirements for the degree

Master of Science

by

Barbara A. Luckett

December

2013

APPROVAL SHEET

The thesis

is submitted in partial fulfillment of the requirements

for the degree of

Master of Science

Barbara A. Luckett

AUTHOR

The thesis has been read and approved by the examining committee:

Peter Beling

Advisor

Barry Horowtiz

Carl Elks

Accepted for the School of Engineering and Applied Science:

James H. Ayd

Dean, School of Engineering and Applied Science

December

2013

Abstract

As a consequence of increased risks of insider and supply chain attacks, it has become more apparent recently that cyber attacks cannot be completely addressed by traditional perimeter security solutions alone. In order to better protect systems, a new systems engineering focused approach, called System-Aware Cyber Security, has been developed. Previous research efforts have led to the development of an expansive portfolio of System-Aware Cyber Security design patterns, which creates a complex multiple criteria decision analysis (MCDA) problem of how to best allocate and implement the protection to create an integrated system security architectural solution that best shifts the asymmetry from favoring an adversary to favoring the US defense.

While MCDA is a very well developed research area with an expansive literature in existence, there are several critical issues that are introduced when considering the cyber security architecture selection process which prompt the need for the development of a decision support tool. In addition to the vast decision space, the tremendous potential for uncertainty in the initial parameter estimates, and the large, diverse group of stakeholders involved, the most critical difference is the presence of an intelligent adversary. While it's obvious that an attacker's actions could cause uncertainty for the defense's system, it's important to recognize that the defense's choices regarding the system can also cause uncertainty for the attacker. This seemingly simple notion – that the design decisions can affect the attacker just as the attacker's decisions affect the system outcome – became a driving force in the development of the current relational methodology for the System-Aware Cyber Security architecture selection decision process.

This research effort proposes a schematic framework designed to utilize a combination of well-known graphical modeling techniques to provide guidance and insight to the decision makers regarding the overall structure of the system and the impacts of their decisions. This methodology involves multiple iterations of Directed Acyclic Graphs and Attack Trees to create a graphical depiction that formalizes the complex structure of the decision process, captures both the attacker and defensive perspectives, and recognizes potential uncertainty in cost and security benefit estimates by providing a more robust approach than scoring alone. In addition to detailing the need for a decision support tool set and describing the developed relational methodology and the graphical modeling techniques it utilizes, this thesis outlines a series of case study workshops conducted on an initial example application. Working through the methodology with a project team provided insight about the usefulness of the framework in a real-word project scenario and provided feedback which has been used to refine the methodology.

Table of Contents

I. Introduction

A. General Problem Statement

Every day, people are faced with making decisions. Whether it is an individual deciding what entrée to order at a restaurant or a corporation deciding on a new marketing initiative, decisions are common occurrences and, more often than not, they involve multiple objectives and/or uncertainly involving an alternative's outcome. "Consideration of different choices or courses of action become a multiple criteria decision making problem when there exist a number of such standards which conflict to a substantial extent." [1] Using this definition, any number of decisions that an individual encounters on a daily basis could be classified as multi criteria decision analysis (MCDA) problems, but since the implications of these decisions are usually fairly minor, the decision maker can typically rely on intuition as opposed to a formal method of preference assessment. However, in situations where the stakes are higher – whether the decision affects a larger number of people, the consequence are more serious, the impact will be felt for a longer period of time, or the decision at hand is a part of a multi-stage decision – it becomes important to formally structure the decision problem in order to accurately evaluate each alternative's performance against each objective. There are multiple techniques in existence for formally structuring a decision problem, but they can generally be categorized as decision analysis methods. The term decision analysis has grown out of the fields of operations research, systems analysis, management sciences, control, and cybernetics. Decision analysis "provides tools for quantitatively analyzing decisions with uncertainty and/or multiple conflicting objectives" [2] and "helps the decision maker clarify in his own mind which course of action he should choose." [3]

In these more complex decision problems, it is vital that the decision maker (or a separate analyst, depending on the situation) takes the time to formally define the elements of the decision system and construct an appropriate frame around it. This step often gets overlooked but it is crucial, especially in a group decision making scenario where individual decision makers may all have a distinct knowledge base and background [4]. By explicitly defining the scope of the system and the relationships between the

random variables considered, a structured frame ensures that the problem can be understood by everyone involved so that an optimal decision can be reached. Since graphical representations are generally more easily understood than algebraic formulations [2], graphical modeling tools have gained considerable popularity for a variety of decision making applications, especially in large group environments.

B. Application-Specific Problem Statement

As cyber attack threats have evolved, system protection strategies have been forced to as well. In order to better protect systems from insider and supply chain attacks, a new systems engineering focused approach, called System-Aware Cyber Security, has been developed by Jones and Horowitz [5]. System-Aware Cyber Security is defined as the utilization of reusable security techniques that are integrated into the system, creating a solution architecture that is designed with a specific application in mind and thus is able to provide unique security capabilities. This protection can be implemented through a variety of techniques – e.g., capabilities to deter potential attackers, detect when the system has been compromised, isolate the sub-systems that have been compromised, or restore the system to an original, uncompromised state – each of which not only has an impact on the attacker regarding the overall security of the system, but also a cost for implementation and set of collateral system impacts felt by the defense. This expansive portfolio of techniques means that multiple System-Aware security design patterns could be candidates for integrated security system architecture, and deciding how to best allocate and implement the protection becomes a complex MCDA problem.

At the highest level, this question involves identifying the system functions requiring more protection, the design patterns that can effectively be implemented on the selected system functions, and finally, the combinations of system functions and design patterns that best reverse the asymmetry of the attacks from favoring potential attackers to favoring the US defense. The cyber security architecture selection process introduces several challenges that make it unique from other MCDA problems, which have prompted the need for the development of a decision support tool.

II. Background and Motivation

System-Aware Cyber Security focuses on developing security architectures that are specific to a system application but are based upon reusable system security services. The architecture includes services that: (1) collect and assess real-time security relevant measurements from the system being protected, (2) perform security analysis on those measurements, and (3) execute system security control actions as required. A variety of these services have been developed thus far [5], including:

1. Significantly increasing the difficulty for adversaries by avoiding a monoculture environment through the integration of a diverse set of redundant subsystems involving hardware and software components provided by multiple vendors
2. The development of subsystems that are capable of rapidly changing their attack surface through hardware and software reconfiguration (configuration hopping) in response to perceived threats
3. Data continuity checking services for isolating faults and permitting moving surface control actions to avoid continuing operations in a compromised configuration
4. Forensic analysis techniques for rapid post-attack categorization of whether a given fault is more likely the result of a cyber attack than other causes (i.e. natural failure)
5. Additionally, although not part of the preliminary portfolio of design patterns, parameter assurance has since been added as a response to the discussions at the initial meetings and workshops.

These security services are all intended to decrease the attractiveness of an attack to the adversary – i.e.: alter the attack steps to make the attacker less likely to succeed or make the attack more expensive or technically difficult for the attacker, lessen the consequences of a successful exploit, or make it easier for the defense to detect or recover from an attack. As mentioned previously, the design patterns can be applied to various features within the system and these implementation strategies can be combined to

create an almost infinite number of possible architectures. In order to be able to assess the value of these various architectural solutions, research focus shifted to developing a scoring-based methodology.

Since MCDA is a very well developed research area with an expansive literature in existence [4, 5], it is only natural that it provided a starting point for the challenge of deciding how to best implement System-Aware Cyber Security protection services. As a first iteration by Rick Jones [6], it was determined that the decision makers are concerned with six factors for any given architectural solution candidate under consideration: (1) Deterrence, (2) Real-Time Defense, (3) Restoration, (4) Implementation Cost, (5) Lifecycle Costs, and (6) Collateral System Impacts.

In a very pure MCDA approach [4], each of these factors is assigned a weight representing its relative importance to the decision maker and each alternative solution is assessed and assigned a numerical, discrete score between 0 and 5 inclusive regarding its performance for each criterion individually. These scores are then aggregated with the weights to provide a single cohesive score that represents the solutions' overall value, and this score can be used to justify a final recommendation (where the highest scoring alternative is understood to be the best performing).

While this weighted scoring method created an objective way to evaluate two alternatives, it failed to provide guidance to the decision maker in determining which alternatives should be scored. A commonly recognized shortcoming of MCDA techniques is that they assume a "given the state of the problem" approach, which means they rely on a predetermined set of alternatives as an input and then focus on methods for comparing them [4]. While this strategy certainly has its uses in a wide range of real-world applications from identifying critical infrastructures to pharmaceutical layout decisions [7,8], it falls short for the System-Aware Cyber Security decision problem which doesn't have a concrete set of alternatives initially. Instead, the architecture selection decision process begins with an expansive set of existing design patterns, which can be applied to variety of system features and combined to create a near infinite number of possible alternatives. Due to the vast size of the decision space, it is simply not feasible to

enumerate and score every possible solution. Since there is obviously more value in scoring alternatives that are expected to perform well as opposed to those that are not, research focus shifted to the development of a decision support tool to assist the decision maker in filtering the solution set and eventually designing an appropriate solution.

In addition to the initially large decision space, there are several other critical issues that exist with the System-Aware Cyber Security architecture selection problem that differ from a standard MCDA problem. First of all, since the decisions being made relate to both the design of the system being protected and forecasts of potential future cyber-attacks, there is tremendous potential for uncertainty in the initial parameter values used to derive the scores. The estimates for cost, collateral system impacts, and security benefits associated with any given design pattern will not necessarily be known for certain until the production of the system is completed and the system is deployed. Also, with a cyber security application, there is also additional uncertainty introduced since the system's outcome is dependent on decisions made by the attacker. In this scenario, the attacker is an active player in the system and one whose choices are unknown to the defensive team at the time when their choices must be made.

Additionally, with the presence of an intelligent adversary, as opposed to purely random natural events/failures which can be assumed to be independent, it becomes obvious that there is a higher probability of encountering multiple smaller, although dependent attacks. As such, an individual exploit may not be critical on its own (at the system component level) but it could become critical if executed along with another attack action. This becomes especially vital in applications that can be cast as System of Systems scenarios where cyber-attacks are based on a collection of concurrently executed exploits across geographically and/or organizationally distinct subsystems.

Finally, for the majority of System-Aware Cyber Security applications, the problem involves a large group of stakeholders and decision makers – all of whom bring a unique knowledge and experience base to the problem. The process is further complicated by the desire to include everyone in the decision

making process and ensure that each individual understands the problem and is able to contribute as new information is gained through each iteration.

While all of these issues are obviously important and should be used to drive the direction of research, the presence of an intelligent adversary is what makes the cyber security problem unique and really sets it apart from existing fields such as safety or reliability. It is important to note that the original weighted scoring method did recognize the presence of an attacker in the sense that it relied on estimates of the expected change in security benefit for an implemented design pattern. However, while it recognized that the attacker's actions could cause uncertainty for the defense's system, it failed to recognize another important point: the defense's choices regarding the system can also cause uncertainty for the attacker. This seemingly simple notion – that the design decisions can affect the attacker just as the attacker's decisions affect the system outcome – became a driving force in the development of the current relational methodology for the System-Aware Cyber Security architecture selection decision process.

III. Existing Methodologies

This section is intended to provide an introductory overview to three key areas of the literature review that was conducted as a portion of the research effort: (1) the existing graphical modeling techniques that were used as building blocks for the relational methodology described here, (2) the incorporation of game theory concepts to address the presence of an intelligent adversary, and (3) several other methodologies in existence that were developed to serve similar purposes in evaluating potential security architectures.

1. Graphical Modeling Techniques

In general, it's recognized that models are simplified representations of complex real world problems [9]. While this means that some complexity is lost in the creation of a model, effective modeling seeks to make appropriate subjective assumptions and find a balance between the level of detail necessary to be representative and that to be useable [10]. The use of a model makes a complicated system or decision more understandable to the user, which in turn allows them to make better decisions. Specifically, the use of graphical models has gained popularity over models involving algebraic formulations for several reasons: graphical representations are "natural for the novice, convenient for computation, and yet powerful enough to convey difficult concepts among analysts and researchers." [11] Additionally, graphical models are beneficial in showing which decisions are best over a range of possible parameter values, rather than only offering point solutions [2]. Due to the range of people involved in the decision process and the high potential for uncertainty in the initial parameters, this research effort has chosen to focus on utilizing graphical methods as a decision support tool for the cyber security architecture selection process. There are multiple graphical modeling techniques in existence, two of which have been outlined below along with their history, notation, and unique benefits and drawbacks when applied to the System-Aware Cyber Security architecture selection process.

a. Directed Acyclic Graphs

A Directed Acyclic Graph (DAG) is a graphical model consisting of two parts: a set of nodes and a set of directed arcs (or edges, as they are known in graph theory) connecting those nodes. DAGs are common in the fields of mathematics and computer science and can be used to model a variety of systems, structures, and processes. DAG is a fairly broad term, encompassing several other more specific classes of graphs, so it is difficult to accurately assess when DAGs first came into use. However, Bayesian networks and influence diagrams (two well-recognized modeling techniques that utilize DAG notation) were developed in the late 1980s and "have become increasingly popular within the AI community since… due to their ability to represent and reason with uncertain knowledge." [12]

Figure 1 shows a very basic DAG to demonstrate how the notation can be used to depict influence relationships within a system.

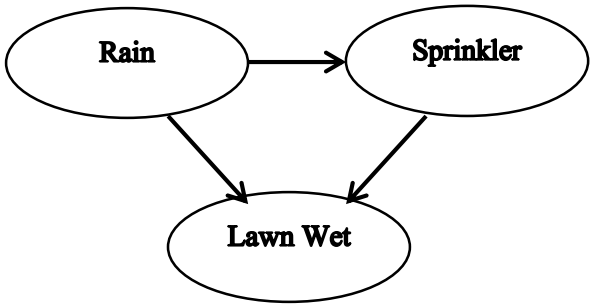

Figure 1. Simple Example Directed Acyclic Graph to Demonstrate Notation

The three nodes in Figure 1 represent three variables in the system: "rain," "sprinkler," and "lawn wet." Each of these variables has two possible values it can take; for instance, it can be raining or not raining, and the sprinkler can be turned on or off. The arcs connecting the nodes represent influence relationships between the variables. Since there is an edge from "rain" to "sprinkler," we know that whether or not it is raining will influence whether or not the sprinkler is turned on. Additionally, the graph shows that the status of "rain" and "sprinkler" both have an influence on whether or not the grass is wet.

As utilized in the relational methodology, each node of a DAG represents a random variable and the edges between nodes specify the influence relationships among those variables; nodes that are not connected are assumed to be conditionally independent of one another. It is important to stress that a DAG is not necessarily a data flow diagram and it is not intended to depict the transfer of information throughout the system. Rather, each node here is a variable that has an expected functionality and has the potential to compromise the overall performance of the system if it doesn't perform in that manner.

An additional concept that will be utilized regarding DAGs, and Bayesian networks in particular, is the notion of a Markov blanket. For node A in a Bayesian network, the Markov blanket of A is defined as the set δA, composed of A's parents, children, and children's other parents (commonly referred to as spouses). The term, which was first coined by Judea Pearl in 1988 [13], can be represented as:

$$P(A|\delta A, B) = P(A|\delta A)$$, where B is a node not included in δA

As defined, the Markov blanket of a node is the only knowledge needed to predict the behavior of that node. Markov blankets are utilized in machine learning pattern recognition, statistics, and data mining communities, and the goal is to improve prediction performance while reducing storage requirements and the time needed for training/inference processes. For the System-Aware Cyber Security problem, if an attacker wants to alter the behavior of a particular node, they need to understand the behavior of that node's Markov blanket. If the defensive team is able to insert more nodes into the Markov blanket (through effective implementation of appropriate design patterns), it increases the uncertainty associated with the attack and the difficulty for the attacker

DAGS (and their subcategories of Bayesian networks and influence diagrams) have been applied to a range of applications over the years including multiple medical diagnosis scenarios, the grammar checker developed for Microsoft Office, and more recently, expansion into product design and development [12]. They are particularly useful for the cyber security selection process because they allow decision makers to represent the causal relationships among a very large number of variables [14]. The existing UAV system

in combination with the expansive portfolio of previously developed System-Aware Cyber Security design patterns creates a very large set of variables that all influence the overall success or failure of an attack on the system. By constructing various iterations of DAGs, the decision makers are able to capture the system variables, structure the relationships among them, and illustrate the impact the possible defensive solutions have on the attacker. However, the use of DAGs alone does not capture a particularly important aspect in the cyber security application: the presence of an intelligent adversary and how their actions impact the system, just as the design team's decisions do.

b. Attack Trees

The second graphical technique utilized in the relational decision support methodology is an attack tree. The attack tree technique first originated in the intelligence community under the term threat logic trees in the late 1980s or early 1990s. The technique was popularized several years later as attack trees by Bruce Schneier who recognized the need for a technique to model threats against systems. By understanding the various ways that a system could be attacked, Schneier envisioned that design teams could better design countermeasures to those attacks. Additionally, by understanding who the potential attackers could be (and their associated resources, technical abilities, motivations, and goals) he recognized the potential to allocate defensive measures to the most likely attacks [15].

Similarly to decision trees and fault trees, attack trees are represented by a diagram with a single root node at the top that represents the overall goal or objective of the adversary (i.e., a successful attack) [16]. On its own, the root node is so broad that it doesn't provide much information as to how an adversary might execute an attack, so the tree continues to branch down, breaking the root node into smaller steps. This process of decomposing the intermediate goals continues until the leaf nodes (which cannot be further broken down) are reached. The leaf nodes in an attack tree represent specific, concrete actions that an adversary could take and that could lead to a successful attack if executed in combination with others

in the tree. Figure 2 shows a very simple attack tree, with the purpose of demonstrating the notation and illustrating the different types of nodes.

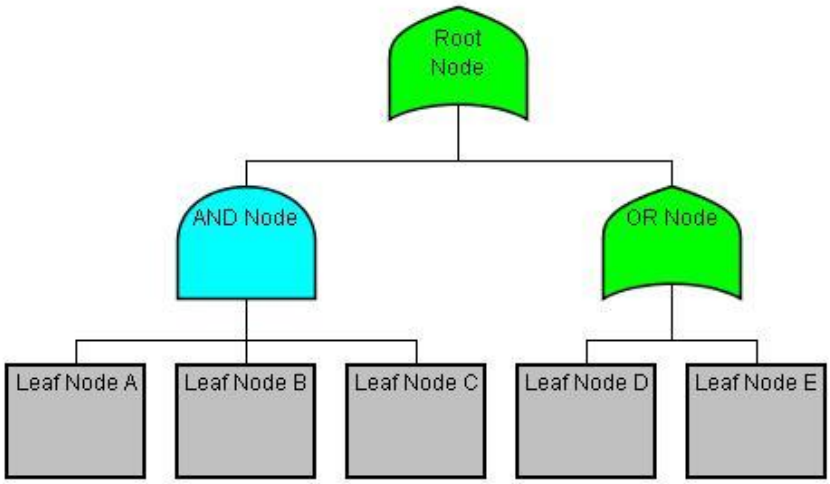

Figure 2. Simple Example Attack Tree to Demonstrate Notation

Since the root node in Figure 2 is an OR node, the attacker only needs to execute one of the subsequent nodes on the following level: completing either the AND node or the OR node will be sufficient to accomplish the overall goal. If the attacker chooses to attempt the AND node, they must complete all three of its associated leaf nodes (A, B and C). On the other hand, if they choose to take the right hand path of the tree, they only need to complete one action (either leaf node D or E), since they are joined by an OR node. This means that the attacker has three different paths to traverse the tree (known as attack scenarios): (1) nodes A, B and C, (2) node D, or (3) node E. The successful completion of any one of those sets will allows the adversary to reach the root node at the top of the tree and accomplish their overall goal.

After the tree has been constructed and the set of leaf nodes has been identified, analysis can be carried out to identify a subset of the most likely attack scenarios. This includes making assessments for several different adversary profiles regarding their preferences and the capabilities they are expected to possess and assessing each attack action in regards to those same behavioral indicator variables. This information

can then be used to prune the attack tree for a specific attacker to identify the subset of their most preferred attack actions. By incorporating information regarding factors that would influence the adversary into the analysis, judgments can be made about which attack actions are more desirable to the adversary [15]. Attack scenarios which are near or beyond the attacker's capabilities/resources are less preferred than attacks that are perceived as simple and inexpensive. Additionally, the extent to which an attack satisfies the adversary's objectives also affects their choices: actions that are both within the adversary's capabilities, and which satisfy their goals, are more likely to be perused than those that do not [16].

Attack trees are constructed from the perspective of the adversary. When building the tree, the modeler must focus on what the attacker wants to achieve and the various ways to accomplish it, and not how to best defend the system [15]. For this reason, attack trees provide valuable information but are not sufficient on their own in the cyber security application since they are created with a fixed system design in mind and they do not capture the implications of possible defensive design choices.

2. Incorporating an Opponent or Adversary

It is obvious that there have been many instances in past military or defense decisions that involved an intelligent adversary. However, the speed at which a cyber attack can now be developed and executed as well as the hidden nature of the development of such exploits introduces some complexities that haven’t existed in the past.

There are three conditions that are recognized as perquisites for an attack (cyber or otherwise) to occur [9]:

1) The defense’s system must have vulnerabilities
2) The attacker must have the capability (e.g.: knowledge, technical ability, and resources) to exploit the existing vulnerabilities

3) The attacker must believe they will benefit from executing the attack; this belief drives the motivation

The first condition is completely within the defender's control, condition two depends on the both the defender and the adversary, and condition three is primarily dependent on the adversary. The fact that the attacker and defender jointly determine whether or not an attack will occur is a driving factor for the decision analysis process here. By only considering the defense's design choices and not the attacker's, the decision maker is ignoring information that, although it is not known with certainty, is still tremendously valuable.

The increase of applications in counterterrorism or even corporate competition over recent years has led to an increase in research and literature focused on analyzing decisions where there exists an intelligent adversary and uncertainty in outcomes. This area, sometimes called Adversarial Risk Analysis, draws upon the knowledge and techniques of both probability and game theory [17]. There has been considerable work done in this field, ranging from examining the tradeoffs between vulnerability and secrecy concerns [18] to the tradeoffs between utilizing warnings or physically deploying defensive resources [19]. This work makes an assumption that the attacker has perfect information about the target system and uses that to make their decisions [20]. This structure allows the problem to be modeled as a signaling game where one player (the attacker) has information the other (the defender) does not. In this case, the attacker can determine their best attack, given the (known) defense:

$$a^*(d) = argmax_{a \in A \psi A}(a, d)$$

where $a^*(d)$ is the attacker's optimal strategy given the defense's strategy

And we can assume the defense is aware that this is the attacker's strategy so they must solve:

$$d^*(a) = argmax_{d \in D \psi D}\big(d, a^*(d)\big) \qquad [17]$$

While these formulations will not be explicitly solved as stated in the optimizations above, this strategy will be incorporated into the proposed methodology on a more conceptual level: the framework will first aim to identify the actions that are most preferred by the attacker (given knowledge of the target system), and then use that information to drive the design team's choices.

3. Methodologies Used for Similar Applications

a. Architectural Scoring Framework

The Architectural Scoring Framework (ASF) was developed to address some of the recognized limitations of the previously described weighted scoring method (outlined in Section II). While the weighted scoring method created a structure to allow two architectures to be objectively compared, it did not provide guidance for selecting a subset of architectural solution candidates to be evaluated. This point led to the desire to expand upon the scoring structure to gather information regarding the preferences and priorities of the decision makers and use that information to guide the selection of potential architectures. At a very high level overview, the ASF methodology "includes the selection of critical system functions to be protected using System-Aware security, identification of asymmetric attack vectors, selection of design patterns to protect those system functions from potential cyber attacks, the integration of those system functions and System-Aware design patterns into candidate architectures, a separate evaluation process to determine which of those candidate architectures should be evaluated using the [previously described scoring process], and finally an evaluation of how the architecture affects the asymmetry between potential attackers and the system being protected." [6] The framework was segmented into two distinct phases:

1. Design and selection of architectural candidates for evaluation, which includes eliciting information from the decision makers regarding the critical system functions to be protected using System-Aware security and selecting design patterns that could be applied to protect those system functions from potential cyber attacks.

2. Use of the rigorous weighted scoring methodology for providing a more detailed assessment of the benefits and costs of the proposed architectural candidates from step (1) in order to aid in the selection of a final architecture.

Although the addition of the first step addresses the need for a way to determine which solutions to score, the ASF methodology still has some limitations [6, 21]. First, the initial prototype assumes that the set of potential design patterns are independent; i.e., the methodology relies on a simplifying linearity assumption where security effectiveness scores and costs are elicited considering each design pattern individually and not in combination. Furthermore, the decision to implement any one design pattern does not preclude any other pattern from being implemented. While these assumptions make the methodology easier to implement and simpler for decision makers to understand, it is not realistic as it ignores possible economies or diseconomies of scale that could arise from shared knowledge and dependencies among available design patterns. Additionally, it is well recognized that the cyber security application introduces tremendous potential for uncertainty in the initial parameter values used by the scoring methodology: for instance, the cost and security effectiveness values for any given architecture are not know for certain. If a single input parameter changes, the final numerical score for a solution that depends on that value would change, and this could impact the final recommendation produced by the methodology. Scoring alone is fairly rigid, so the methodology could be improved and made more robust by showing the solutions' sensitivity around uncertain values.

The ASF methodology, and the Excel prototype system developed for its use, was applied to a case study involving implement System-Aware Cyber Security protection services on an unmanned aerial vehicle (UAV) platform in 2012. The UAV project also serves as the application for the relational methodology case study described in Section V.

b. Mission-Oriented Risk and Design Analysis

The Mission-Oriented Risk and Design Analysis (MORDA) methodology is "a risk and design method developed by NSA and IDI for designing functional and secure systems." [22] Similarly to the ASF, MORDA was designed for use by multi-disciplinary teams with multiple stakeholder perspectives. The framework begins by developing a list of attack scenarios, each of which gets assigned a numerical value between 0 and 1 (where an attack with a value of 1 is "very likely to succeed, unlikely to be detected, and inexpensive to conduct" and a value of 0 means the reverse). The attack values are summed into a portfolio score, and the change in this total score is measured as different design options are implemented.

Each Countermeasure and Design Option (CDO) is assessed with respect to the three adversary value model measurements (likelihood of detection, likelihood of success, and adversary resources required) and the cost to the US to implement. Additionally, the framework recognizes the diseconomies of scale associated with implementing multiple CDOs and does similar assessment for the set of all possible CDOs together. The methodology uses an Excel-spreadsheet based optimization tool to determine the best CDO in terms of the trade-off of performance vs. cost and a Net-Value Added model to determine a stable ordering of CDOs to implement. The Net-Value Added structure implements the best CDO first, and the values of subsequent CDOs are determined given that the first CDO is already in place.

The MORDA framework better deals with the (dis)economies of scale issue that the ASF had. However, MORDA has other challenges, including (1) a dependence on quantitative data and the lack of sensitivity analysis and (2) difficulty getting system owners and security analysts to understand each other, the architectures, and the results of the analysis.

As of publication in 2007 [22], MORDA had been used on ten different projects, with varying numbers of attacks, adversary classes, CDOs and architectural solutions.

c. Network Risk Assessment Tool

The Network Risk Assessment Tool (NRAT) uses the two fundamental components of risk to make judgments regarding the protection of information systems: probabilistic risk analysis to estimate the likelihood of an attack and an evaluation of the potential severity of the attack's impact on the operational missions the system supports [23].

The framework begins by developing potential threat actor profiles and potential attack scenarios. It then compares some basic attributes of the attack's requirements with the same basic attributes of the potential threat actors (the software provides general attack traits to avoid bias and subjectivity of individuals using the tool). These comparisons are aggregated with "standard logical and mathematical functions" to produce percentage values representing attack success likelihood, and the net likelihood that an attack is successful is calculated by obtaining the product of the two individual likelihood values representing attacker competency and system vulnerability. Additionally, the NRAT framework assesses the consequences of the potential attacks on the services the system provides and maps those to the associated missions. The software then uses a series of algorithms to calculate each attack's impact on each mission. The net risk of each attack is determined using both the probability and severity information. By determining the risk for both a baseline system and an enhanced protection system, the user can determine the percentage improvement associated with the protection and compare that percentage of the mission value to the cost of the implemented protection.

Similarly to both the ASF and MORDA methodologies, the NRAT also has a firm quantitative focus and does not include any type of sensitivity analysis for dealing with the potential for uncertainty. Also, while the NRAT framework does compare two architectures to weigh the cost-benefit trade-offs, it is not specifically geared towards a design situation. Rather, it focuses on comparing two fixed system designs instead of providing guidance in developing an architectural solution.

At the time of the article's publication in 2008 [23], NRAT was being evaluated for its use for application for the defense of military information systems.

IV. Proposed Framework

The relational decision support methodology described here was designed to be an iterative process that relies on inputs from a range of stakeholder communities. In order to ensure that the information being used is as accurate and certain as possible, it was imperative to ask individuals questions that were appropriate to their backgrounds and areas of expertise. This was accomplished by initially dividing the stakeholders into three distinct groups: the blue team, red team, and green team. The blue team consists of designers and users of the system being protected, and their responsibilities include identifying/prioritizing the critical system functions to protect and determining which design patterns can be implemented on which system functions. The red team is made up of individuals with knowledge of cyber-attacks and potential threat agent classes, and their work is focused on developing candidate attack vectors and assessing the effectiveness of the proposed design patterns. The green team, which is comprised of experts in system cost analysis and adversary capability, will analyze costs, to both the attacker and defender, for candidate architectural solutions.

The relational methodology in its entirety is composed of six steps: each of which is well defined with the goal, required deliverables, and the responsible team(s) for that stage. This information is presented at a high level here intended to capture the general flow from one step to another. The scope at this point is fairly general; two full examples from the case study workshops conducted are outlined in Section V to demonstrate the usability of the approach in more detail and explain refinements that were made based on the case study results.

A. Step 1: Define the Variables and Relationships Within the System to be Protected

The initial step of the relational methodology is focused on framing the problem to ensure that all participants in the process are on the same page regarding the system to be protected. The process begins by identifying the most critical function of the system and defining the variables and influence relationships within that portion. Obviously, most, if not all, systems that would require protection with

System-Aware Cyber Security services are not simple systems; instead, they have multiple functions that are all part of the overall concept of system functionality. The rank ordering of the system functions and the identification of a single, most critical function serves to provide initial scoping for the process. After the completion of the framework for the most critical function, the methodology can be completed on additional functions as deemed necessary by the project team.

Step one is to be performed by the blue team and is intended to outline the expected functionality of the system "as-is" with minimal defensive strategies implemented. At this point, a system influence relational diagram is constructed using DAG notation. This diagram is created for the system without the consideration of a cyber attack to ensure that everyone involved in the process is in agreement on the most basic structure and components before the additional complication of an adversary. DAGs provide immense value in situations where a system is characterized by a large number of variables and where rationally and successfully making decisions is dependent on understanding on the interrelationships among those variables. For this reason, they work well for considering a system of this scale and have been used for a variety of applications in the safety and reliability fields.

As explained in Section III, a DAG includes a set of nodes and a set of edges connecting the nodes. In the system influence relational diagram constructed in step one, nodes represent random variables within the system. These can be hardware or software components, interfaces, or external factors, all of which have an expected functionality and can influence the outcome of the system. The edges connecting the nodes represent the influence relations between the variables. If two nodes are connected, that means the value one node is dependent on the value of the other (value here being whether or not the node functions as expected). Similarly, if two nodes are not connected, the functionality of one does not have an influence of the other. While a DAG alone overlooks a critical aspect of the problem at hand (the presence of an adversary), its construction enables the team to reach a common understanding of the system before advancing in the process.

There are two deliverables for this step: (1) a rank ordered list of system functions to provide initial scoping for the process and (2) a system influence relational diagram, using DAG notation, constructed for the most critical function.

B. Step 2: Recognize the Possible Paths an Attacker Could Take to Exploit the System

Step two introduces one of the issues that make this specific problem unique: an intelligent adversary. While the system influence relational diagram represents a system where success may be compromised by random failures, the cyber security architecture selection problem introduces concerns where the decisions made by an active player in the system can also compromise mission success. In step two, the red team is tasked with constructing an attack tree for the specific system function considered in step one. By looking at the system from the perspective of an adversary, attack trees can be utilized to understand the possible paths an attacker could take to exploit a specific feature of the system.

As outlined in Section III, the process of developing an attack tree requires the decision makers to think like an attacker and identify different strategies that an attacker could take to accomplish the overall attack goal. An attack tree graphically captures the decomposition of these strategies to individual root actions; minimal cut sets of the leaf nodes represent various attack scenarios that the adversary could execute to successfully reach the root node. The attack tree provides obvious value here by graphically capturing the interplay between the defender's system and the adversary, but is not a sufficient technique on its own. Since the attack tree is created considering a fixed system design (the system laid out by the DAG produced in the first step), it does not capture the back-and-forth nature of the problem or the design decisions under the consideration of the defense team.

There are two deliverables required for the completion of the second step: (1) an attack tree capturing how an adversary could execute an attack on the system feature and (2) the complete set of possible attack actions (which are the leaf nodes identified from the construction of the tree).

C. Step 3: Determine the Subset of Attack Actions Most Desirable to an Attacker

Considerable analysis can be conducted after the construction of an attack tree. However, rather than focusing on quantitatively calculating the probability of success for a specific attack scenario as is

typically done in attack tree analysis, the analysis included in this framework considers a more qualitative, abstract metric space. In step three, the green team develops a set of variables that can be used to assess the difficulty of a particular attack action. These variables are called behavioral indicators and can include, but are certainly not limited to resources such as technical ability, time, manpower, money, equipment, facilities, presence of an insider, and access to system design information. These variables are used to make two separate types of judgments: leaf node assessments and adversary profile construction.

While there is a substantial set of possible behavioral indicators in existence, they are not all necessarily well-defined. The selection of a set of variables to use for a particular application, and the clear definition of the meaning of those variables in the context of that specific application is a critical part of the process. Rather than expanding on specific hypothetical behavioral indicator variables at this point, that discussion will be held off until the case studies in Section V where it will be valuable to elaborate on the actual variables used in the process of assessing the tree and the potential adversary classes.

For the leaf node assessments, the green team goes through the entire set of possible attack actions identified by the red team in step two and assigns a value to each node in regards to each behavioral indicator variable. These assessments are not quantitative calculations of likelihood of success, but rather broad classes of knowledge and effort required by an adversary in order to be willing to attempt the attack and able to successfully execute the exploit. Similarly, the same behavioral indicator variables can be utilized to construct an adversary profile; meaning for a specific threat actor that the team is concerned with, the green team can assign a value in regards to each variable for the level that the adversary is expected to possess. These two sets of behavioral indicator assessments (the leaf node assessments and the adversary profile) can then be used in combination for a process referred to as pruning an attack tree. Pruning essentially compares the levels of various behavioral indicators to create a reduced attack tree with a subset of the original leaf nodes that a specific adversary would be capable of.

There are several deliverables for this step: (1) a well-defined list of behavioral indicator variables to be used, (2) an assessment of each attack action in the tree in regards to these variables, (3) an adversary

profile defining a particular threat actor's expected resource levels, and (4) a pruning tree constructed for the particular adversary.

Step 4: Identify Appropriate Defensive Actions and Their Impacts on the Attacker

After the red and green teams have identified the actions that an adversary would need to take to successfully execute an attack and the subset of those that are most attractive to a particular adversary, the blue team can then determine which of their existing defensive actions may be appropriate. The relational methodology relies on the assumption that a portfolio of design patterns has already been developed -- either by the blue team previously or by an external group no longer involved in the process. If the blue team was not responsible for developing the set of design patterns, it is assumed that they have access to the portfolio and the have the necessary knowledge regarding the meaning of each design pattern.

The goal in step four is to select design patterns from the existing portfolio that could be implemented to make the actions captured in the leaf nodes of the attack tree less desirable to the attacker. This can mean increasing the difficulty, cost, or noticeability to the adversary, or lessoning the consequences felt by the defense in the case of a successful attack. The blue team begins by comparing the set of attack actions from the pruning tree to the existing portfolio of design patterns and brainstorming possible design patterns of value. At this point, each potential defensive solution can be added to the system influence relational diagram created in step one so that the diagram represents the system with the identified defensive alternatives implemented and shows their value integrated into the holistic system for context. In this second iteration, the DAG notation is used to model an understanding of the complexity added to an attack.

This concept of increasing the complexity of an attack is directly related to the notion of a Markov blanket for Bayesian networks. Adding each of these possible choices into the system relational diagram also adds additional nodes that are related to specific defensive alternatives. The success of a particular implementation choice depends on other variables in the system, and these nodes become other factors that, if that strategy is chosen, the adversary must now consider in order to still successfully exploit the system. Returning to the terminology introduced in Section III, implementing a design pattern effectively

inserts new nodes into the attacker's Markov Blanket, because there are now additional factors the adversary needs to account for.

The updated system influence relational diagram serves two purposes. In addition to graphical representing a strategy's impact on the attacker through the addition of nodes in the attacker's Markov blanket, it also aims to capture the benefit of an action in the context of the system as a whole. By adding defensive strategies to the system influence relational diagram, the diagram can provide insight regarding the breadth vs. depth of the protection provided, which works to ensure adequate coverage of the system and recognize possible (dis)economies of scale that occur because of combinations of design patterns.

After constructing the updated system influence diagrams, the new nodes that have been inserted can be evaluated in regards to the same behavioral indicators defined in step three. These new nodes represent additional actions that the adversary will have to complete in order to successfully complete their original exploit so the level of difficulty and resources required can be assessed in a similar manner as the leaf node assessments conducted in the third step.

There are three deliverables required for the completion of the fourth step: (1) a list of design patterns selected from the existing portfolio that may increase the difficulty for an attacker to execute the actions considered most preferable, (2) updated system influence relational diagrams with the potential defensive actions included, and (3) assessments on the new nodes in the system influence relational diagrams regarding the levels required for each behavioral indicator variable.

D. Step 5: Evaluate the Impacts of the Selected Potential Actions on the Defense

While step four captures the design patterns' impacts on the adversary, step five transitions to evaluating the impacts those same choices have on the defensive team. The green team is able to apply their second class of intelligence information here: cost analysis estimates for the defensive solution choices. At this point, each of the design patterns selected in step four is evaluated in regards to implementation cost, lifecycle cost, and collateral system impacts. The green team is responsible for estimating the monetary cost of a solution, but the blue team also adds input on a solution's collateral system impact here – since they have knowledge regarding the system, how it will be used, and what

impacts are unacceptable. Any solutions that are deemed to be beyond budget or have unacceptable impacts on system performance can be eliminated from further analysis at this point.

There is one deliverable for this step: a reduced list of possible defensive choices, filtered down from the original existing design pattern portfolio to only those that increase the difficulty for the considered attacker while still remaining at an acceptable impact to the defense. This subset of the original portfolio of design patterns is carried into step six and discussed to select the final defensive strategy(ies) that should be implemented.

E. Step 6: Weigh the Security Trade-offs to Determine Which Architectural Solutions Best Reverse the Asymmetry of a Potential Attack

The goal of the sixth and final step is for all three teams to return together and participate in a collaborative discussion regarding the security trade-offs that exist with the potential choices determined in step five. While each defensive strategy remaining after step five has an acceptable impact on the attacker and on the defense, some are obviously better choices than others. Figure 3 below demonstrates this point through a simple diagram. It's important to note here that this graph is not a deliverable of step six (because neither the impact to attacker nor the impact to defense is consolidated into a single quantitative value), but rather just a tool for explanation of the bigger picture concepts that should be discussed at this point.

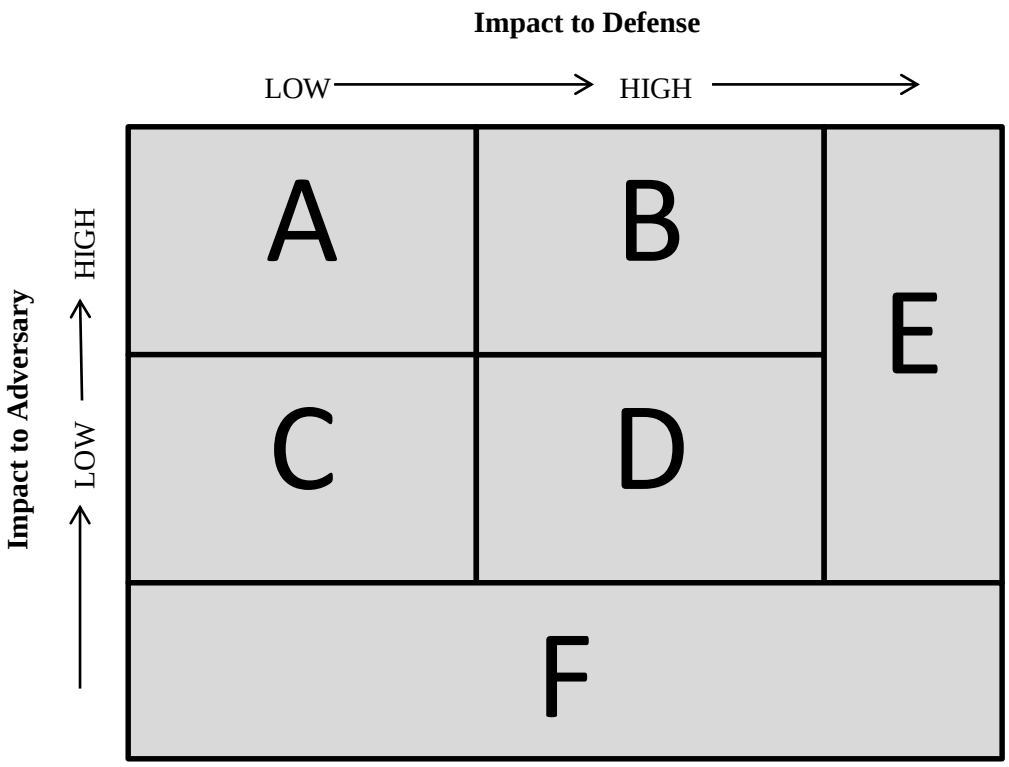

Figure 3. Quadrant Chart Representing Different Types of Strategies Remaining at Step 6

At the conclusion of step five, there are a variety of possible solutions remaining for consideration and these are represented by quadrants A, B, C, and D in Figure 3. Point A is the most obvious choice for implementation, with its high impact to the attacker and low impact to the defense. Points B and C are less obvious but still may have considerable value. Point D passed through the filtering processes in step four and five because the defensive strategy it represents does have an adequate impact on the attacker and the impact it has on the defense team is acceptable. However, when these impacts are compared as a pair, one can easily see that the asymmetry is not in the defense's favor: it requires the defense spending a considerable amount of money to only have a minor impact on the attacker's actions. This design pattern should not be ruled out until it's been discussed and considered because it could provide value in an area of the system influence relational diagram that otherwise is not affected, but it should be recognized as a less obvious choice.

Neither points E and F are included in the set of possible actions going into step six because they were both eliminated by the analyses in earlier steps. Point E was eliminated at the end of step five because it had an unacceptable impact to the defense – either because it's over budget or has some other

collateral system impact that the team has deemed unacceptable. Point F was eliminated at the end of step four because it did not increase the difficulty or uncertainty for the attacker to complete one of their most preferred attack actions. While these points do not need to be discussed in step six, it's important to note that they have not been fully eliminated. Records should be kept of defensive strategies that are eliminated throughout the methodology, so these design patterns can be revisited down the line if something changes. For instance, if the defensive budget changes, the team may want to consider the solution associated with point E.

The final reduced set of possible defensive strategies produced in step five will include actions in quadrants A, B, C, and D as described above – all of which increase the difficulty for a specific adversary to complete one of their most preferred attack actions while remaining at an acceptable impact to the defense. In the ideal situation, the design team would have the budget to implement all of the remaining alternatives, but more likely the combination of these actions is still more than what the team can afford to implement. Thus, step six is intended to guide the group in weighing the security trade-offs that exist among these remaining alternatives in order to select a subset to be implemented as part of the final solution. There are four factors that should be considered when piecing together the final cohesive security architecture: (1) budget, (2) coverage, (3) dimensionality, and (4) asymmetry. In the structure provided here, budget and coverage are used as prescreening mechanisms to build a set of sample architectures, and dimensionality and asymmetry are used to evaluate these architectural solutions

Even though each remaining alternative is within the budget on its own, the overall budget obviously needs to be kept in mind when considering combinations of design patterns. The project team should have a rough budget in mind at this point of what they are able to spend on protection for the system. Based on the assessed implementation cost values for each defensive strategy from step five, the team should be able to narrow down the number of design patterns they would expect to implement. For instance, the group is not expected to know at this point that they plan to select one alternative with a Medium Implementation Cost and two with Low-Med Implementation Costs. However, they should be able to

specify if they can only afford a single strategy, plan on picking a majority of them, or plan on picking two or three in the Low-Med to Medium cost range.

The remaining set of possible defensive strategies protects multiple system features from multiple different attack types. From reviewing and discussing the set of alternatives remaining, the team will most likely have certain areas that they are drawn to and want to further protect. Based on their budgetary range and the areas of concern within the system, the team can construct several sample architectures for further evaluation regarding the asymmetric shifts.

Using the information elicited in steps four and five, the impacts on both the adversary and the defense can be presented for each architectural solution considered. At this point, the entire project team can discuss the asymmetry regarding the impact a solution has on the adversary compared to that it has on the defense. The goal is to select an architecture that creates the biggest asymmetric shift: greatly increasing the adversary's difficulty to complete the attack while keeping the cost of implementation to the defense low in comparison. Impact on the adversary is measured by the levels of the behavioral indicators required by the adversary in order to still execute the attack after the implementation of the protection (which were assessed in step four). Multiple behavioral indicators cast the evaluation of a solution's impact on the adversary into multiple dimensions, and an optimal solution can be created by combining strategies that are strong in different dimensions. The most obvious example of two solutions complimenting each other in terms of multidimensionality would be one where two solutions fill in the other's gaps for each behavioral indicator. For instance, if the implementation of one strategy forces the adversary to have a high level of design knowledge but only requires a low manpower/time demand, this strategy could be paired with another than requires a high manpower/time demand but may have a low value for design knowledge. The two solutions work together to force an adversary to expand their capabilities in multiple dimensions to still be able to execute the attack

The deliverables at this point will vary but, at the very least, it should include written documentation describing the pre-filtering process and comparing the alternative solutions given in step five and justifying the final selection.

F. General Improvements from Scoring

Both Bayesian networks and attack trees typically have some associated level of probabilistic analysis, and it should be very obvious that this concept has been excluded from the decision support methodology previously described. This was a conscious decision made due to the general youth of the field. The fact that there is a high potential for uncertainty in the initial parameter values (both regarding cost estimates and expected security benefit) created issues with the original weighted scoring methodology, which didn't show a solution's sensitivity around uncertain values. If a single input parameter changes or the team decides to consider a different adversary profile, the final numerical score for a solution that depends on the value would change, and it could change rather dramatically. Scoring alone is fairly rigid, and there was a desire for a more robust decision support tool that could still provide criteria for evaluating different alternatives but by showing the impact a decision has, rather than just comparing a solution's numerical score to another. By removing the quantitative focus, the relational methodology hopes to combat the inherent, recognized uncertainty and shift attention to obtaining general insights and understanding decisions' impacts on the system as a whole.

Another conceptual improvement for the relational methodology from the weighted scoring methodology was the move to consider design patterns in combination as opposed to in insolation. For the initial scoring approach, the cost and security benefit estimates were elicited for each design pattern individually. While this assumption simplified the approach and made the method useable, it was not realistic and ignored possible (dis)economies of scale that could arise with combinations of defensive solutions implemented. Intuitively, one would expect the security effectiveness of a design pattern to be dependent on previously implemented defensive strategies, so it's very important to consider the solution in the context of the entire system to gauge the value it adds. By adding potential design patterns into the original system relational influence diagram, the updated DAG works to visually depict these relationships.

V. Case Study Results

Currently, the University of Virginia (UVA) and the Georgia Tech Research Institute (GTRI) are involved with a joint research project focusing on applying System-Aware Cyber Security solutions to an existing Unmanned Aerial Vehicle (UAV) platform. This application provided the initial insight into the need for a decision support tool drove the research effort outlined in this thesis. While the UAV project and team served as an initial case study, the methodology presented here is envisioned to be applicable to a variety of different applications regarding the decision support process for the System-Aware Cyber Security architecture selection.

In order to initially demonstrate the validity of the proposed framework and depict the type of information that can be gained through its use, a simple version of the UAV application was developed. This example, which was outlined in the thesis proposal, did provide assurance that the methodology would provide value in scenarios regarding cyber security decisions; however, it was overly simplified and had the real possibility of minor technical errors since it did not utilize the entire team's breadth of knowledge during its construction. For the completion of the thesis, a series of case study workshops were held with members of the UAV project team to evaluate the framework using a more detailed and through, technically sound example. These workshops had two objectives: (1) to provide feedback regarding the methodology and its usefulness in a real-world application setting and (2) to provide value to the UAV project itself by filtering down the initial portfolio of design patterns and providing insights into the impacts of the design team's decisions. The outcomes of these workshops are described below and organized as following:

A. Case Study Workshop 1 – this subsection describes the first workshop attempt. No analysis from this meeting is reproduced here, but several interesting points were raised in this meeting which went on the influence the structure of the future case study meetings in two key ways.

B. Case Study Workshop 2 – this subsection describes a presentation to the project team which detailed the application of the relational methodology to a simple example. The analysis here is

completely unrelated to the UAV project, but the workshop served as a simple demonstration of the framework.

C. Case Study Workshop 3 – this subsection describes a series of case studies conducted over a two month period to apply the relational methodology to the real-world UAV navigation system application. It should be noted that while the overall structure of the methodology is consistent between the simple example workshop and the UAV navigation system application workshop, refinements were made between the two and there are minor differences in several steps. The structure as described in subsection C is the most complete and accurate as far as how the framework would be envisioned to be applied to similar projects in the future.

A. Case Study Workshop 1

The first case study workshop had eight team members in attendance and began with a brief presentation on the framework itself (specifically the need for a decision support tool, the goals for the methodology, and an overview of the analyses conducted in each of the six steps). Following the presentation, the plan had been to begin working through the analyses required by the framework for an example centered on the UAV navigation system; however the group was only able to make it to the very beginning of the second step in the allotted time. This difficulty was due to two factors, which are described below and drove the refinements leading into the second case study workshop.

First of all, it is important to note that everyone involved in the initial case workshop had been involved in the project for a considerable amount of time (most, although not all, team members had been involved since the project's inception). While the point of the case study was to leverage the team members' technical knowledge, this familiarity and almost emotional attachment to the subject matter led to issues in the earliest stages of the framework. Having spent so much of their own time and effort researching and working on the UAV project, individuals were very passionate about the material at-hand and had difficulty focusing on the big picture of the relational methodology. Additionally, the open-ended structure of the workshop left too many opportunities for team members to interrupt one another and go off on tangential topics. Even after just seeing the presentation on the structure of the methodology, it

was obvious that the group hadn't grasped the big picture of the methodology and couldn't trust in where it was going yet. For instance, there were several occasions throughout the workshop where individuals made comments suggesting that the group do activity 'x', even though 'x' was set to occur at a later step in the framework.

Additionally, even though the methodology was created to use three different teams, this condition was relaxed for the initial case study due to the limited number of people involved with the project and the desire to get as much feedback as possible. The entire team was involved in both the development of the system influence relational diagram in step one and the construction of the attack tree in step two. While having the entire team involved provided a lot of valuable insight about the framework, it also caused issues where people had knowledge from previous steps in their minds – even if those tasks were technically supposed to be done by an entirely different team. Specifically, during step two (which, as explained in Section IV, is performed by the red team), some participants kept attempting to refer to the system influence relationship diagram (which is intended to be constructed by the blue team in step one). Even though this was not information the red team was technically supposed to have access to, several participants wanted to frame the attack tree in such a manner that built directly off of the DAG.

These issues affected the future case study workshops in two ways. First of all, a simple example that was not connected to the UAV project was presented to allow the participants to reach a point of understanding regarding the framework itself. The issue of dividing into separate teams was not as straightforward to address. While it was clear that the structure from the first workshop did not work, purely following the designated team structure would mean that a wealth of potential feedback for the methodology and insight for the project would not be collected. To address this, case study steps were separated out into distinct workshops over several weeks to allow time between sessions for participants to disengage from the previous task(s) of a separate team. This, in addition to several smaller more team-specific workshops interspersed between the other workshops sought to find a balance between the separation of distinct teams and the expertise of involving more participants.

B. Case Study Workshop 2: Simple Example

Based on the results of the first case study workshop, a separate example, completely unrelated to the UAV project, was developed and presented at a second workshop. The goal of this meeting was to work through a very simple example to help the team members understand the methodology prior to trying to implement it again on the UAV navigation system example. There were two opposing goals that needed to be balanced in selecting the application for the simple example: (1) it needed to be familiar enough so that individuals could understand the topic without spending valuable time learning application-specific technical knowledge and (2) as noted from the first workshop, it needed to not be overly familiar to the point where individuals were too invested in the problem and couldn't focus on the process itself. The system chosen for the simple example was a single-story residential house, and the results and deliverables from this example are reproduced step-by-step below. The information here seeks to provide a more concrete and less conceptual approach than that provided in the general overview of the proposed framework (Section IV), but not to the same level of detail as that of the full scale UAV navigation system application (Section V, subsection C). For each of the steps of the simple example workshop, the information provided here includes the deliverables required and a brief description to provide a level of interpretation of the analyses conducted.

a. Step 1

The methodology begins by determining the most critical part of the system from a functional perspective. A house has a variety of functions, including but not limited to: (1) shielding its residents from the elements, and providing (2) privacy, (3) climate comfort, (4) light, (5) cooking capabilities, (6) storage space for possessions, and (7) physical security for both the residents and their belongings. For the purpose of this example, the blue team determined that the most critical system function was the physical security provided by the house, so the system influence relational diagram was constructed for this function (shown below in Figure 4).

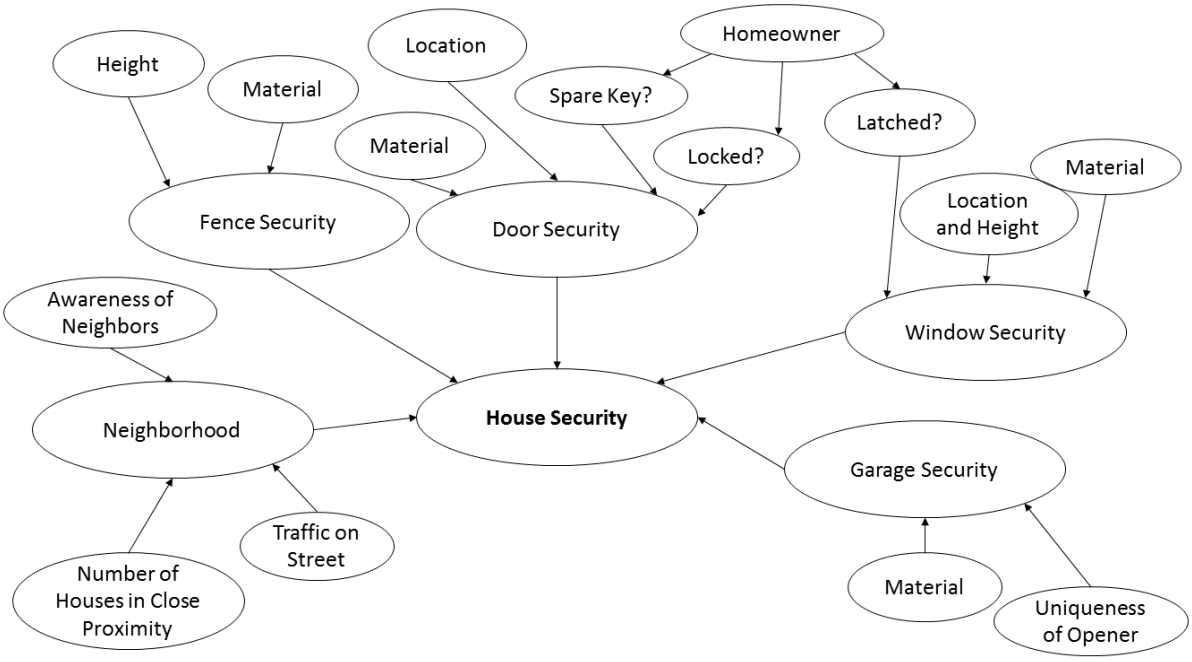

Figure 4. System Influence Relational Diagram Constructed in Step 1

Figure 4 depicts the components within the system that impact the overall security of the house. At the highest-level, the DAG shows that house security is dependent on the neighborhood the house is in, and the security of the fence, the doors, the windows, and the garage. Each of these can be broken down one further: for instance, the security of each door depends on its location (front, back, or internal entry from the garage), the material it is made of, whether or not there is a spare key hidden, and whether or not the door is locked.

b. Step 2

Step two is focused on the red team constructing an attack tree to capture the various paths an attacker could take to exploit the system (represented in Figure 5).

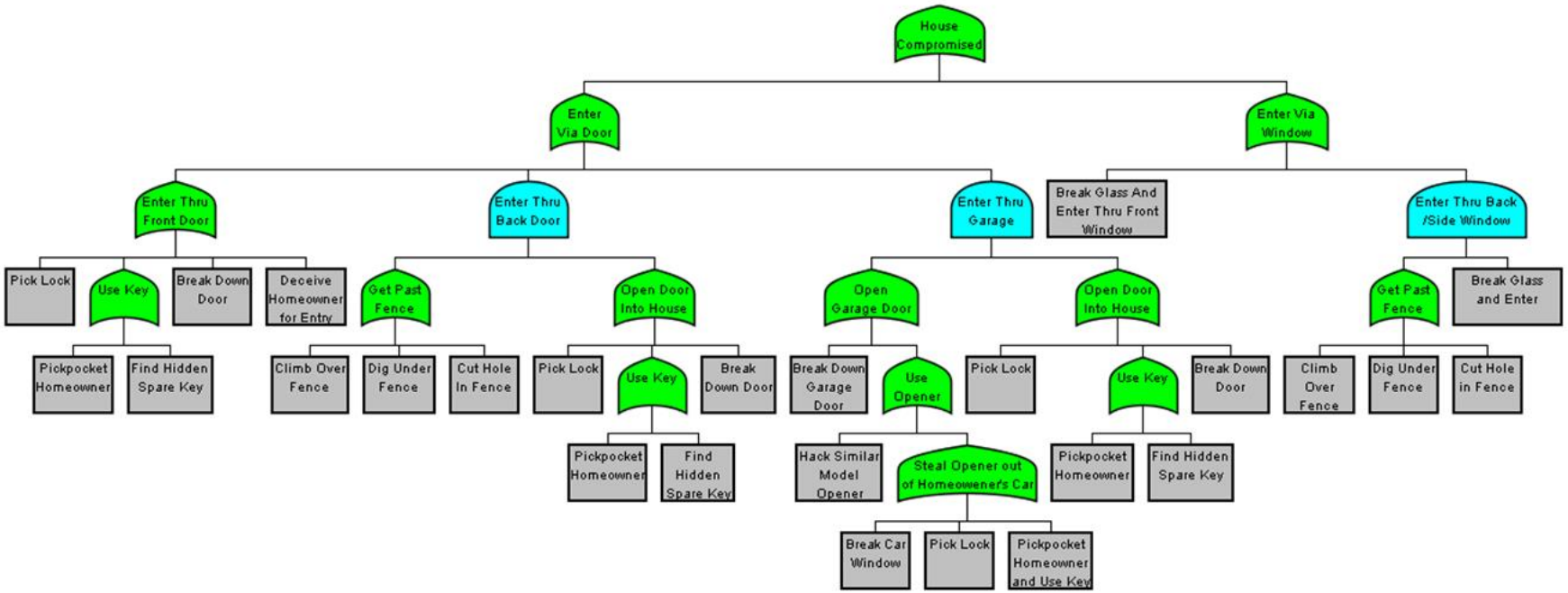

Figure 5. Attack Tree Constructed in Step 2

The root node “House Compromised” is the overall goal of the attacker, and since it is an OR node, the attacker must either “Enter via Door” or “Enter via Window,” but they do not need to do both. On the other hand, “Enter Through Back Door” several levels lower is an AND node, which means that the attacker must both “Gut Past Fence” and “Open Door Into House” (although they have several options on how to complete each of these actions). The tree continues to branch down until the leaf nodes are reached. The final set of possible attack actions is:

1. Pick [House Door] Lock
2. Pickpocket Homeowner [for House Key]
3. Find Hidden Spare Key
4. Break Down Door
5. Deceive Homeowner for Entry
6. Climb Over Fence
7. Dig Under Fence
8. Cut Hole in Fence
9. Break Down Garage Door
10. Hack Similar Model Garage Door Opener
11. Break Car Window
12. Pick [Car Door] Lock
13. Pickpocket Homeowner [for Car Key]
14. Break Glass and Enter Through Window

c. Step 3

It is infeasible for the blue team to defend against every possible attack action identified in step two, so the green team is tasked with assessing which actions are most preferred by specific attackers in step three. In this case, the green team determined that there were three necessary behavioral indicators: (1) cost for attack, (2) attack-specific technical ability, and (3) detectability. For the sake of the example and

demonstration purposes, two adversary profiles were constructed: one for a sophisticated trained burglar group and another for a troublesome neighborhood teenager. The green team assessed that the trained burglar group could afford attacks with high cost and could perform attack actions requiring high attack-specific technical ability, but were not willing to attempt attack actions that had higher than low-medium detectability. However, the neighborhood teenager could only afford attacks with low cost and low attack-specific technical ability, but was less risk averse and willing to try attacks with high detectability.

The set of leaf nodes was also assessed with regards to the same three behavioral indicators, and these values are shown below in Table 1.

Table 1. Behavioral Indicator Levels for the Set of Attack Actions, Assessed in Step 3

Attack Actions	Cost	Attack-Specific Technical Ability	Detectability
Pick [House Door] Lock	Med-High	Med-High	Low-Med for Front Door; Low for Back Door and Internal Door from Garage
Pickpocket Homeowner [for House Key]	None	Med-High	Low-Med
Find Hidden Spare Key	None	Low	Medium
Break Down Door	Low	Low	Medium for Front Door; Low-Med for Back Door; Low for Internal Door from Garage
Deceive Homeowner for Entry	Medium	Medium	Med-High
Climb Over Fence	None	Low	Low-Med
Dig Under Fence	Low	Low	Low
Cut Hole in Fence	Low-Med	Low-Med	Low
Break Down Garage Door	Low-Med	Low-Med	Medium
Hack Similar Model Garage Door Opener	Med-High	High	Low
Break Car Window	None	None	Low-Med
Pick [Car Door Lock]	Med-High	Med-High	Low-Med
Pickpocket Homeowner [for Car Key]	None	Med-High	Low-Med
Break Glass and Enter Through Window	None	None	Low-Med for Front Window; Low for Back/Side Window

In this example, the green team determined that the adversary of concern was the trained burglar group. The pruning tree created for this adversary is shown in Figure 6 and the reduced subset of most desirable attack actions in that scenario is:

1. Pick [House Door] Lock
2. Pickpocket Homeowner [for House Key]
3. Break Down [Internal Door from Garage]
4. Dig Under Fence
5. Cut Hole in Fence
6. Hack Similar Model Garage Door Opener
7. Break Glass and Enter Through Window

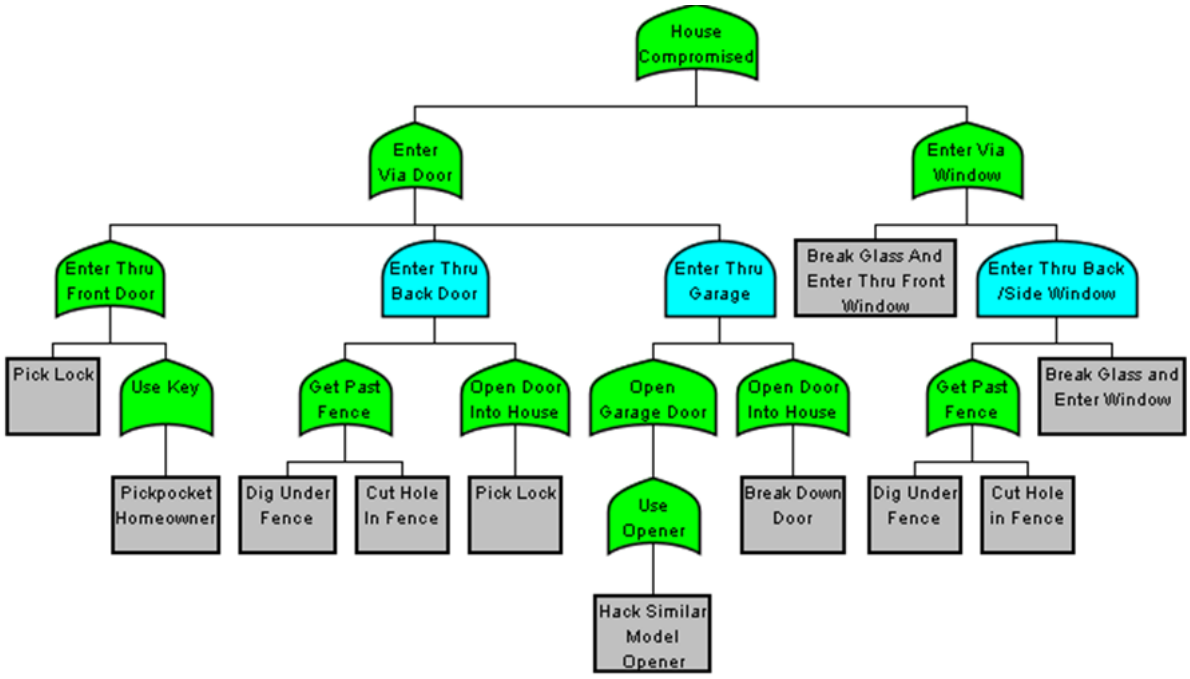

Figure 6. Pruned Attack Tree Constructed in Step 3

d. Step 4

At this point, the blue team is able to refer to the existing portfolio of design patterns to select those that could increase the difficulty of the most desirable attack actions (the leaf nodes of the pruning tree).

Based on the seven remaining attack actions for the trained burglar group in step three, the most applicable design patterns are:

1. Add a deadbolt to any of the doors
2. Reinforce any of the door frames
3. Replace the internal door into the house from the garage with a stronger, thicker one
4. Add an alarm system
5. Recruit neighbors to start a neighborhood watch program
6. Put bars over the windows

Based on the threat actor's preferences and resources, several design patterns were not selected from the portfolio. For instance, increasing the height of the fence was originally a potential option, but would not make sense since the burglar group isn't likely to attempt to climb over the fence, even at its current height. Similarly, the burglar group is unlikely to try to break down a door because that activity puts them at higher risk of being noticed and detected; they are much more likely to attempt to pick the lock or pickpocket the homeowner and use the key instead. The exception is the internal entry door from inside the garage, which is vulnerable to being broken down because of the lower detectability level.

Figure 7 shows these six potential defensive actions added to the original system influence relational diagram constructed in the first step.

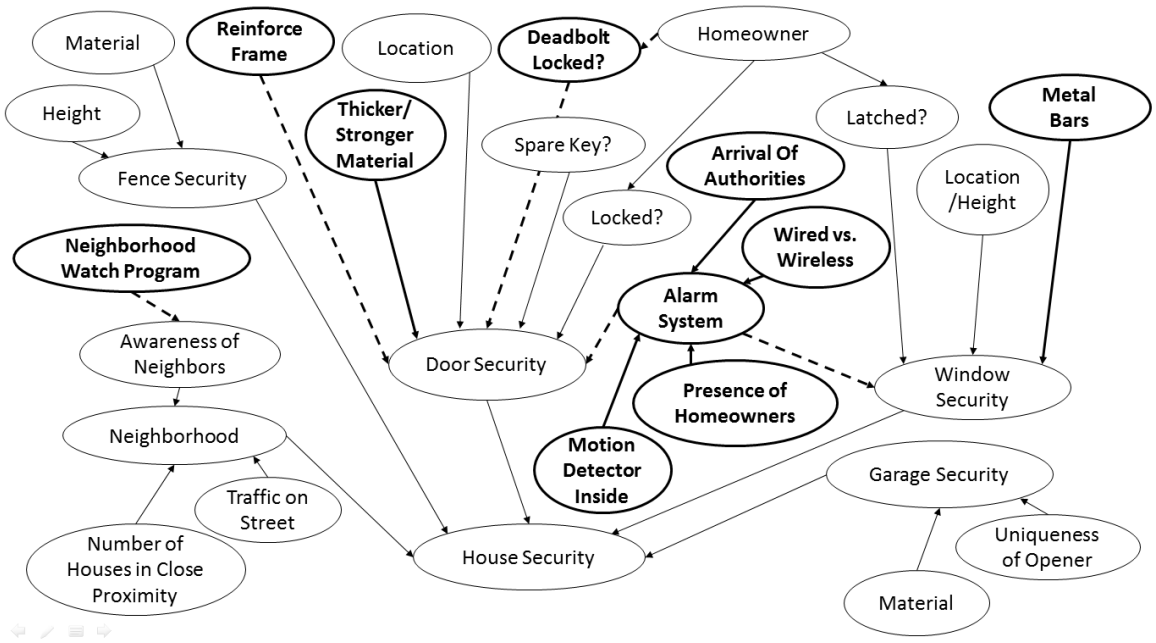

Figure 7. Updated System Influence Relational Diagram Constructed in Step 4

This diagram brings several points to the attention of the team. First of all, the installation of an alarm security system has the biggest impact on the attacker – since it not only impacts two existing nodes (window security and door security), but it also inserts four additional nodes into the diagram (and into the attacker's Markov blanket). For instance, in the original system, the adversary's ease in entry via a window was dependent on simple factors such as the window's location and material and whether or not the window is latched. Adding an alarm system severely complicates the adversary's task by increasing both the uncertainty and difficulty. If the adversary wants to ensure the success of the exploit, they now need to consider factors such as whether or not the system includes a motion detector to reduce false alarms, whether or not the homeowners are present and will be alerted by the alarm, the details of how the system is installed, and how quickly the authorities will arrive after the alarm goes off.

Additionally, the updated system influence relational diagram illustrates the inherent diseconomies of scale of over-protecting the doors. There are three different design patterns that influence the security of a door: (1) reinforcing the door frame, (2) replacing the door with a thicker/stronger one, and (3) adding a deadbolt. The first two increase the difficulty for the adversary to break down a door (so they should only

be implemented on the internal entry door from the garage) while the third increases the difficulty for the adversary to pick a lock. Regardless of which, if any, of the design patterns the defense team chooses to implement, they will get less security benefit from each than expected if they choice to implement multiple defensive strategies on the doors. The DAG notation makes these relationships more obvious and gives the design team the information they need to make better choices.

It is important to note that one deliverable presented in Section IV is not reproduced here. The new nodes introduced based on the implemented design patterns are not explicitly assessed here in regards to the previously defined behavioral indicator variables. This task will be described in the full UVA case study (Section V, subsection C), which will go into more detail of the methodology in general.

e. Step 5

Following step four, the team has information regarding the implications potential design patterns have on the attacker and can turn their attention to the implications those same choices have on the defense. Table 2 below shows the implementation cost, lifecycle costs, and collateral system impacts associated with each of the six remaining defensive strategies.

Table 2. Impacts of Potential Design Patterns on Defense, Determined in Step 5

Design Pattern	Implementation Cost	Lifecycle Costs	Collateral System Impacts
Neighborhood Watch Program	$50 for Starter Kit	None	Requires homeowner initiative to start up
Reinforce Door Frame	$90/door	None	None
Replace Door with Thicker/Stronger Material	$125/door	Steel doors need to be repainted	None
Add Deadbolt	$50/door	None	None
Install Alarm Security System	$350	$50/month	Homeowner has to remember to turn alarm on
Add Metal Bars on Windows	$50/window	None	Homeowner views bars as aesthetically unappealing

At this point, the blue team can eliminate certain design patterns if they are beyond budget or have some other unacceptable collateral system impact. In this example, even though the alarm system has the

greatest impact on the attacker, its cost is deemed to be too high by the blue team. Additionally, the metal bars on the window are also eliminated because the collateral system impact of being viewed as an eyesore is considered unacceptable.

f. Step 6

After these two eliminations, there were four remaining potential solutions to be considered by the three teams in the sixth and final step of the relational methodology. Through a collaborative discussion, the group decided upon a final architectural solution of installing deadbolts on the front and back doors, reinforcing the door frame for the internal entry door from the garage, and implementing a neighborhood watch program.

g. General Comments on the Workshop

Overall, the simple example workshop went very well. The team was able to proceed through the methodology in its entirety in the allotted time. Additionally, the participants seemed to be able to focus more on the process itself rather than getting sidetracked by individuals' research topics, questions, and comments. One issue that did arise however was a concern about the subject matter being too in line with traditional network or perimeter security. While this concern is valid, it is important to note that this example was selected because of its approachability and ease of understanding for a mixed group of participants. An example application closer to the UAV project could have been selected: either another vehicle platform that would have a similar mission or an application more similar in terms of technological complexity, like a power generation turbine. However, if a more complex example had been selected, a considerable amount of time would have had to been spent of teaching the participants about the system. The purpose of this second workshop was to select something simple and easy to understand that would allow individuals to focus on the methodology itself.

While the example subject matter of the physical security of a house does appear to be in line with the notion of perimeter protection, that does not mean that the methodology is in any way limited to network and perimeter security approaches. The first step of the simple example workshop began by listing the

system functions that are associated with a house. While providing physical security for both the residents and their belongings was identified as the most critical system function, any of the other functions such as providing privacy, light or cooking capabilities could have been selected. Had this choice been different, the association to perimeter protection would have been less apparent.

C. Case Study Workshop 3: Full UAV Application

After the successful completion of the simple example workshop presentation, the group was ready to apply the methodology to the UAV navigation system attack example again. This case study was divided into a series of meetings occurring over a period of two months, with some meetings involving the entire project team and others only involving a few specific individuals. Since this series of meetings was intended to both provide value to the project and feedback for the framework, this third case study was considerably more detailed than the simple example workshop and therefore provided a more collaborative environment for refining the relational methodology.

a. Step 1

Again, the methodology begins by determining the most critical part of the system from a functional perspective and outlining the variables and their influence relationships within that portion of the system. At the highest level, the success of the UAV mission is dependent on the success of three separate functions: (1) the system navigating to the correct location, (2) the sensors on-board working to collect the correct surveillance data, and (3) the platform remaining safe and operational throughout the mission. Of those three functions, the aircraft navigation was selected as the most critical; this decision was made in part because this was the area the UVA team had been focused on for the majority of the project. Using this to provide the initial scoping for the methodology, a system influence relational diagram was constructed for the Aircraft Navigation function using the DAG notation. This diagram is shown in Figure 8.

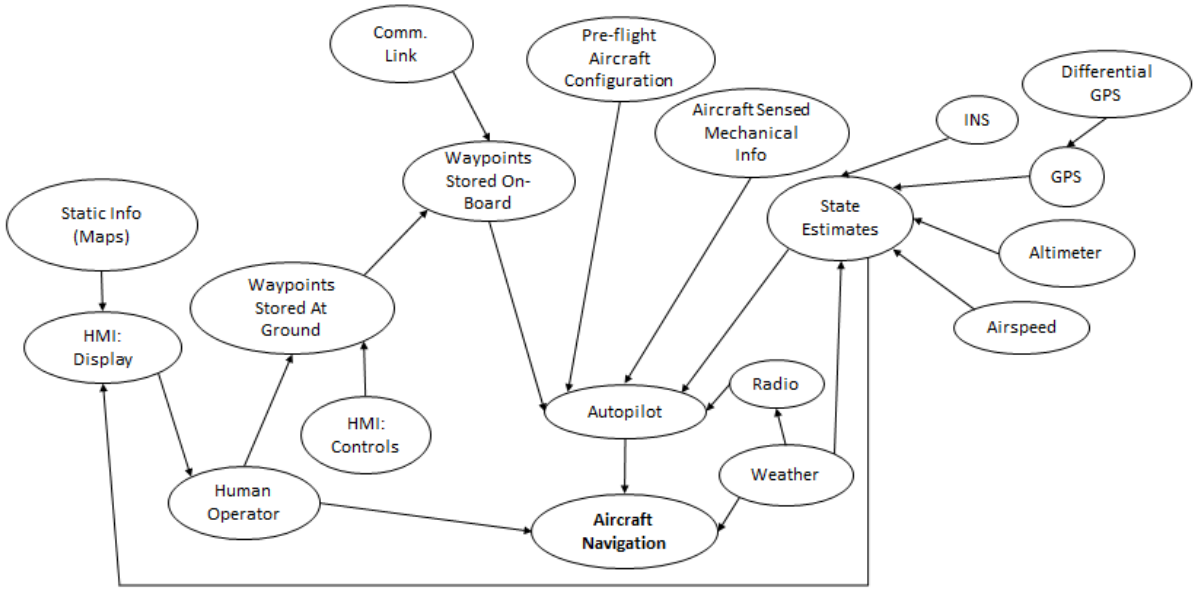

Figure 8. System Influence Relational Diagram for UVA Navigation System "As-Is"

Using the DAG structure as explained in Section III, this graph shows that the outcome of the system function (i.e.: the success or failure of the aircraft navigation function) is dependent on three factors: (1) the actions of the human operator, (2) the functionality of the autopilot software, and (3) the external factor of the weather in the platform's vicinity. Similarly, the diagram shows that the status of the operator display is influenced by static information such as maps that are stored in the software and variable information of the state estimates which are collected on-board the platform. In turn, the information shown on the display influences the actions of the human operator.

b. Step 2

The second step of the relational methodology is focused on the introduction of an intelligent adversary, and this perspective is captured by the construction of an attack tree. Based on discussions with the case study participants at this point, it was determined that a single attack tree would not necessarily represent the whole picture of how an adversary may wish to exploit a particular system function. Specifically, the adversary's desire and motivation for attempting the attack has a large influence on the

manner in which the attack is executed. For instance, for the UAV navigation system attack, two different threat agents could complete the same action (i.e.: change the waypoints on-board the platforms), but the extent to which they do so could create two very different appearing attacks. One attacker may be more concerned with remaining undetected and is willing to severely constrain their attack to accomplish this, while another may want to make more drastic actions and is willing to accept a higher risk of detectability to do so. By dividing the attack structure into multiple trees, the team is able to incorporate the adversary's preferences and motivation and consider the value vs. detectability trade-off that is often present in the cyber attack field.

For the UAV navigation system attack example, the red team constructed three trees, for three different attack types of different potential value to the attacker:

1. A minor trajectory change where the adversary makes a minor change to the waypoints to cause the platform to deviate slightly from the flight path in order to avoid a certain area but then return to the expected path. The assumption here is that the deviation is minor and has a short duration, so the attack is not noticed by the ground station operator.
2. A major trajectory change where the adversary drastically alters the flight trajectory to cause the platform to lose control and crash into the ground. The assumption here is that attack occurs so quickly, that although the ground operator will certainly detect the change, it will be detected too late to prevent the completion of the attack.
3. A concealed major trajectory change where the adversary drastically alters the flight trajectory to cause the platform to reroute and fly to an alternate destination. The assumption here is that the trajectory change on its own will be noticed and prevented by the ground operator, so the adversary must take action to conceal the change in order to successfully complete the attack.

The "Concealed Major Trajectory Change" tree was selected for the analysis moving forward for two reasons. First of all, structurally, all three trees are very similar in regards to how the change is made and the "Concealed Major Trajectory Change" tree includes the nodes of the other two as well as the nodes

representing actions to lower the detectability of the attack. Secondly, the value gained from the "Concealed Major Trajectory Change" attack was most in line with the expected preferences of the adversary profiles the project team was most concerned with.

Figure 9 shows the top level overview of the Concealed Major Trajectory Deviation attack tree constructed in step two. Due to size constraints, the lower portions of the subtrees have been rolled up in Figure 9. These subtrees are depicted in full in Figures 10 -14.

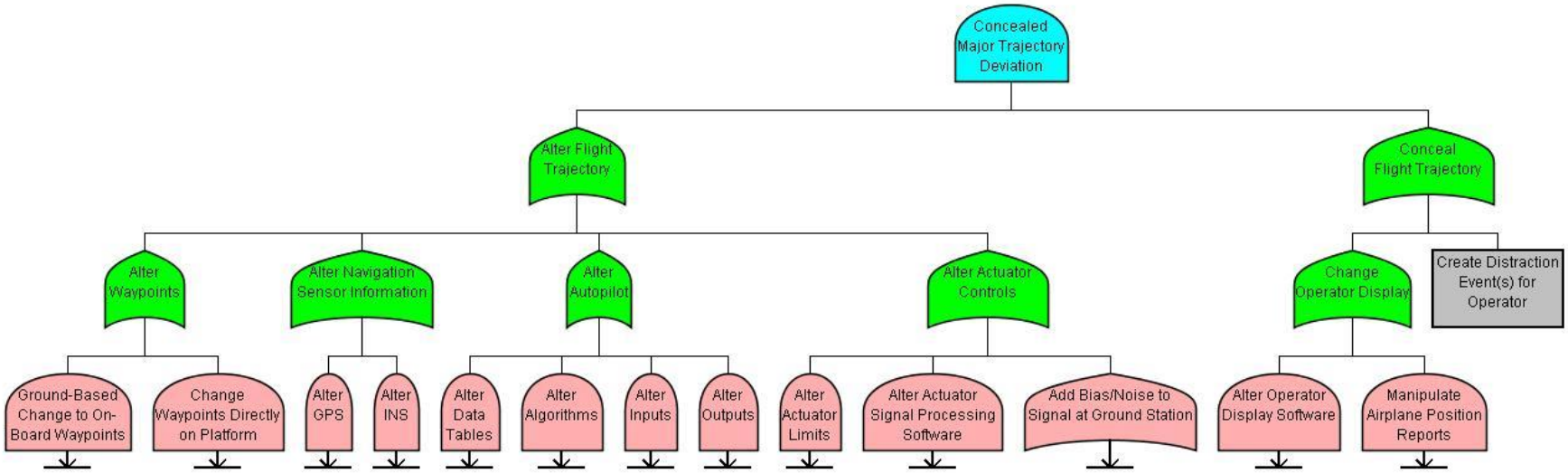

Figure 9. Top-Level Overview of Concealed Major Trajectory Deviation Attack

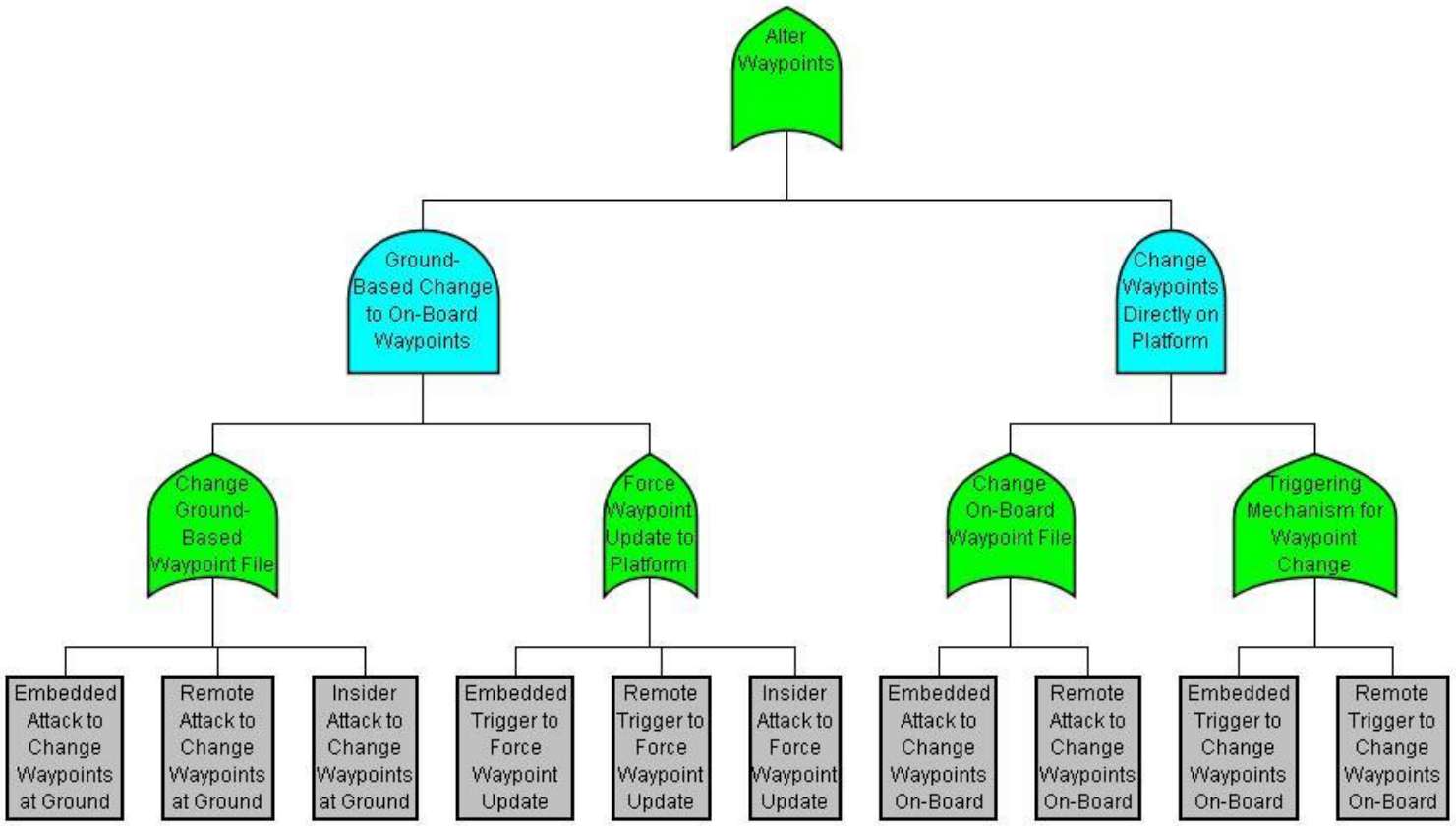

Figure 10. Subtree for "Alter Waypoints"

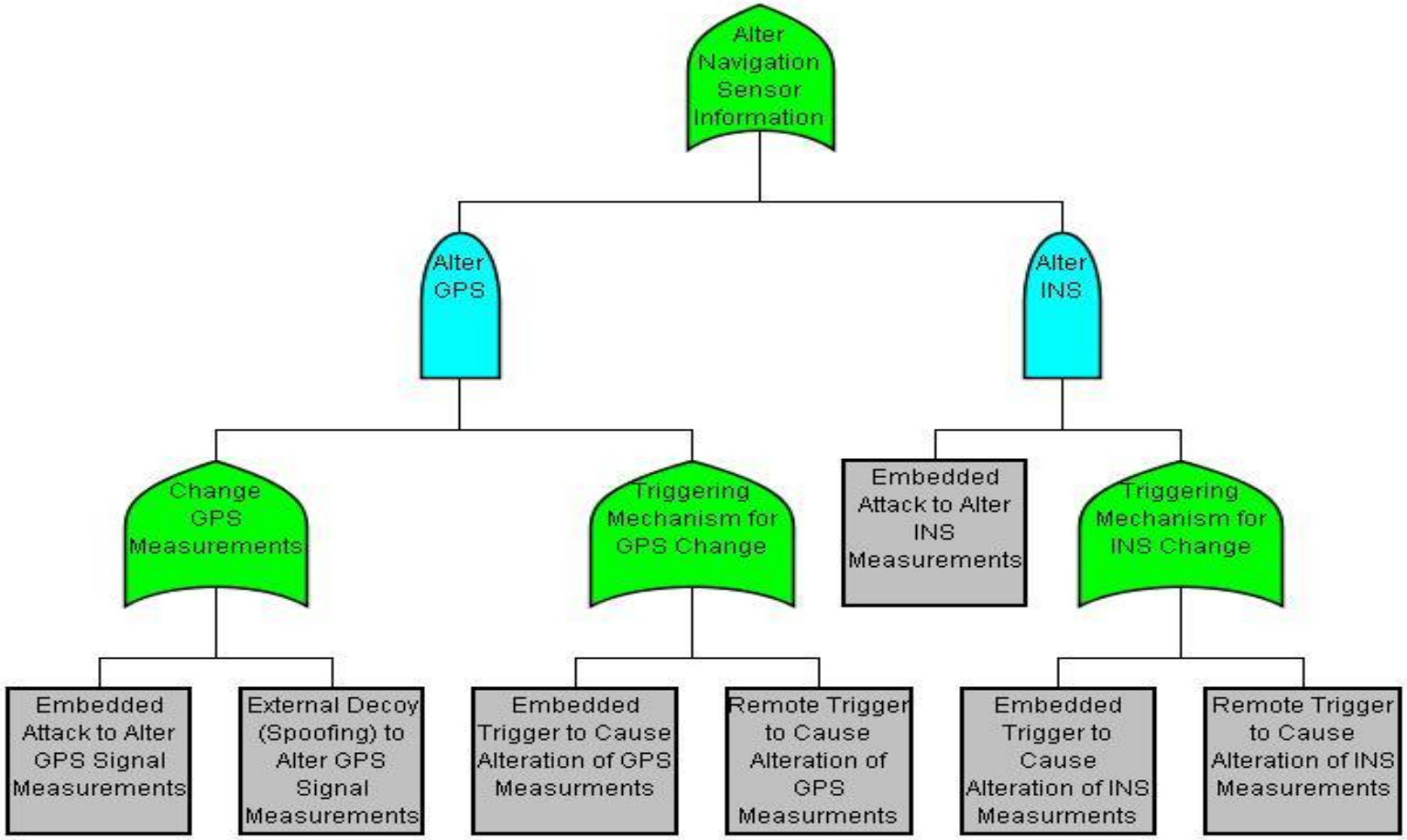

Figure 11. Subtree for "Alter Navigation Sensor Information"

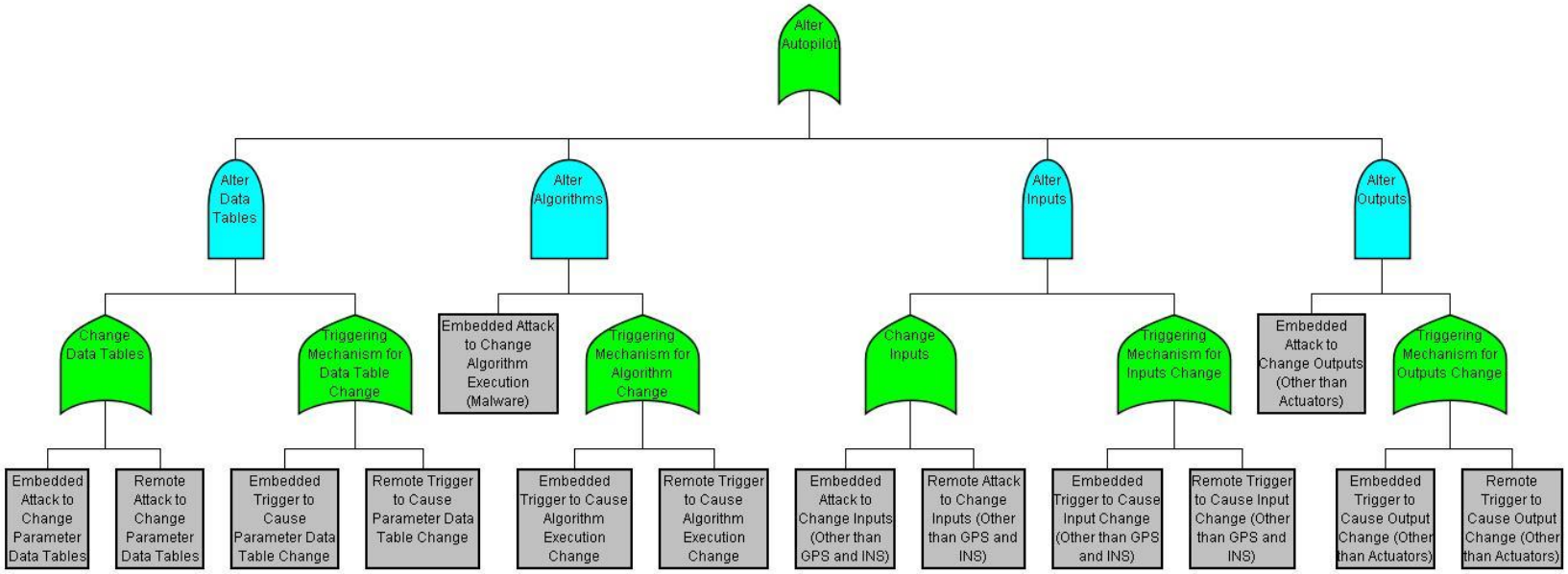

Figure 12. Subtree for "Alter Autopilot"

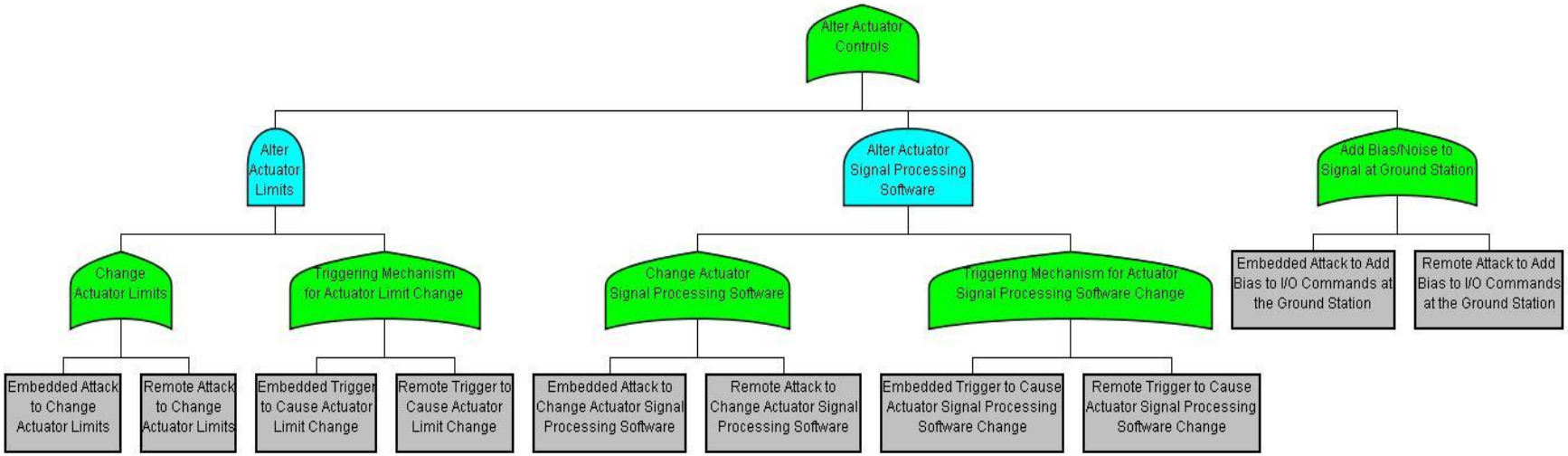

Figure 13. Subtree for "Alter Actuator Controls"

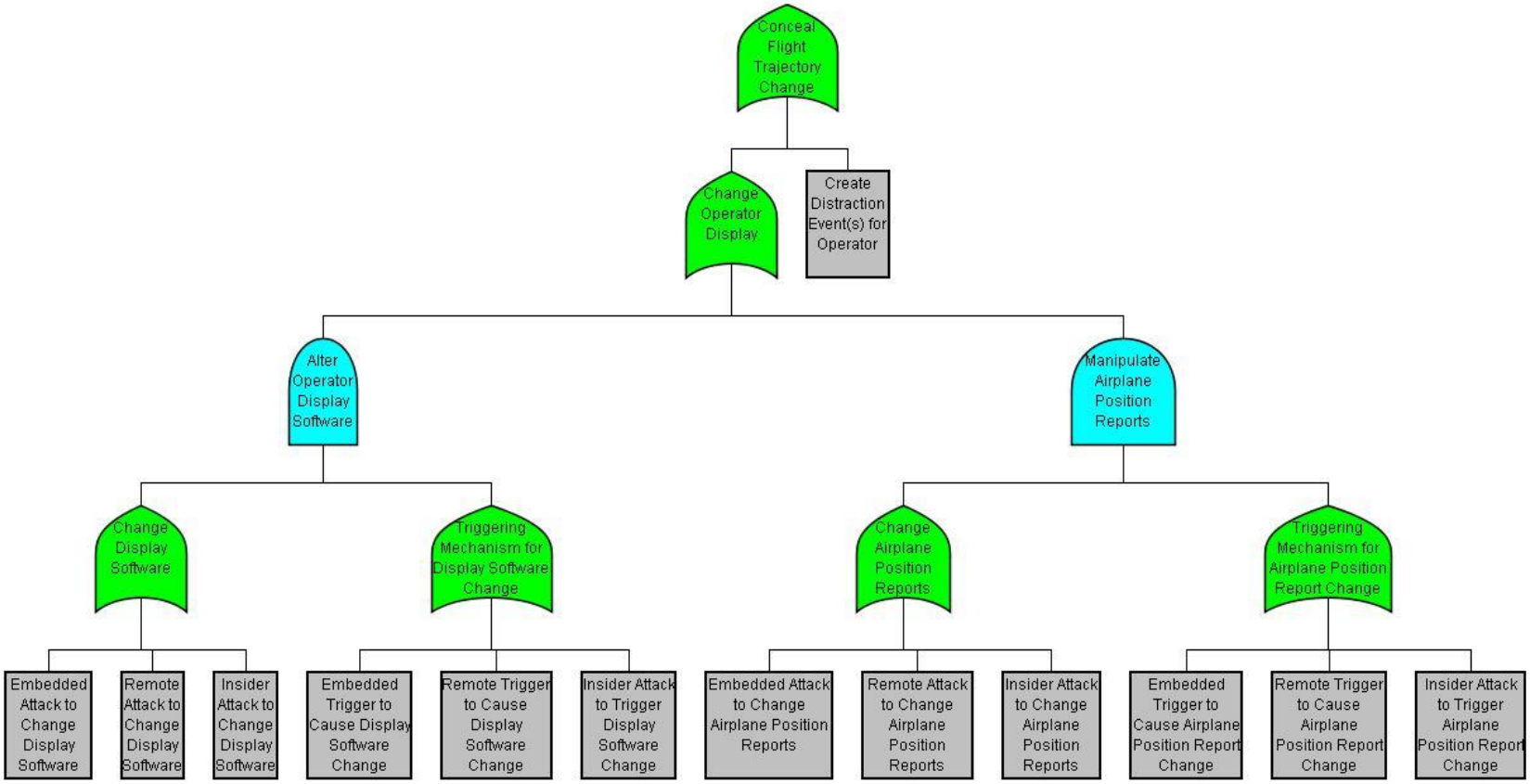

Figure 14. Subtree for "Conceal Flight Trajectory"

As can be seen by the size of the attack trees represented in Figures 9-14 above, the attack on the UAV navigation system exists on a much larger scale that that on the house security from the second case study workshop. There are a total of 55 leaf nodes in the Concealed Major Trajectory Deviation attack tree, which can be executed in various combinations to create a total of 817 possible attack scenarios. Additionally, many of the leaf nodes could be broken down even further in to more specific subtrees detailing their execution. However, the scope of this tree is more than adequate for the purpose of the relational methodology moving forward.

A couple of trends became apparent in the construction of the Concealed Major Trajectory Deviation attack tree. First of all, almost every attack strategy included requires two distinct actions: (1) an action to implement the ability to make the desired change and (2) an action to trigger the change on/off as needed. To capture this concept, each attack is represented by an AND node with two subsequent nodes associated to it – representing the change itself and a triggering mechanism to cause the change to take place. For instance, if the adversary is able to insert a compromised chip into the system through some point in the supply chain, they must also have some method to activate the chip at the appropriate time (as determined by a factor such as time or geographic location of the platform.) Without the triggering mechanism, the infected software is either always on (which could lead to a higher likelihood of the attack being detected and prevented) or is never turned on (which makes the action to enable the attack pointless). A second trend identified from the construction of the attack tree was the repetition of several different categories of leaf nodes:

1. Embedded – where infected hardware/software was added to the system at some point in the production process, typically because of a compromised source in the supply chain
2. Remote – where the attack is executed from outside of the system, either through an existing link to the system or a completely external factor
3. Insider – where an individual who has access to critical aspects of the system and detailed-non-public domain knowledge takes action(s) to exploit the system

4. Miscellaneous – attack actions that do not fit any of the three main designations; for instance, an external decoy (spoofing) or causing a distraction event for the operator

c. Step 3

As noted previously, it's infeasible for the blue team to attempt to protect the system against all 55 attack actions identified in step two. Instead, the green team is tasked with identifying a subset of the leaf nodes that are most preferred by a specific attacker. This filtering process involves assessing both the leaf nodes and a potential adversary profile with regards to a set of behavioral indicators, so step three begins by determining the set of necessary behavioral indicator variables to use. Table 3 shows the final set selected for the UAV navigation system attack example.

Table 3. Behavioral Indicator Variables Names, Possible Values, and Meanings for Leaf Node Assessment

Behavioral Indicator Name	Possible Values	Meaning for Leaf Node Assessment
Design Knowledge	• Low • Low-Med • Medium • Med-High • High	What level of design knowledge is required to successfully complete the attack action?
Attack-Specific Technical Ability	• Low • Low-Med • Medium • Med-High • High	What level of attack-specific technical ability is required to successfully complete the attack action?
Resources	• Low • Low-Med • Medium • Med-High • High	What level of resources (i.e.: facilities and equipment) is required to successfully complete the attack action?
Insider Presence (Operational)	• Low • Low-Med • Medium • Med-High • High	To what extent is having an insider present in the operational phase of the system necessary/helpful in completing the attack action? <u>Note on possible values:</u> Low = entirely unneccessary Medium = helpful but not required High = impossible without
Insider Presence (Supply Chain)	• Low • Low-Med • Medium • Med-High • High	To what extent is having an insider present at some point in the supply chain necessary/helpful in completing the attack action? <u>Note on possible values:</u> Low = entirely unneccessary Medium = helpful but not required High = impossible without
Manpower/Time	• Low • Low-Med • Medium • Med-High • High	What level of manpower and time is required to successfully complete the attack action?

One important point to note is that the word "complete" as used here includes both the planning and execution of an attack action.

The first behavioral indicator, Design Knowledge, encompasses the overall accessibility of information about the target system to the adversary. It includes both open source information or

information inferred from open sources, and proprietary or classified information. Most if not all threat actors begin planning an attack by gathering information about the system, beginning with easily accessible open source data. If information is not readily available, it may deter a potential adversary or create considerable uncertainty regarding the attack execution. At the very least, the lack of open source data will drive the attacker's manpower costs and time required up a significant amount as they will have to spend valuable resources gathering information to better understand the structure of the target system.

Attack-Specific Technical Ability goes one step beyond Design Knowledge. The UAV navigation system is complicated and, even with perfect knowledge about the system, not every adversary will have the necessary skills to execute an attack. If an attack action requires a lot of specific capabilities, it will automatically reduce the set of possible adversaries.

Resources was defined here to be the necessary facilities or equipment required to execute the attack. Some attack actions require fairly standard hardware/software equipment while others necessitated that the attacker purchase or have access to a specific tool (in the case of the UAV navigation system example, there were several cases where the attacker needed to own a copy of the Piccolo II autopilot used on-board the platform to be able to experiment with it prior to attempting the attack). Obviously some attackers will have a wider span of equipment available to them than others, so this becomes another indicator as to whether or not an attacker is capable of performing a certain action.

Obviously, considering any one of the 55 attack actions, the presence of an insider will always be helpful (having increased knowledge about or access to the system can only be beneficial). However, for some leaf nodes, it crosses the line between helpful and necessary. Insider presence considered here was partitioned into two categories – the presence of an insider in the operational phase of the system and the presence of an insider at some compromised point in the supply chain. Specifically, the insider attack nodes basically require that the adversary have someone involved with the operation of the target system. Embedded attacks are an area where, although not required, an insider's presence in the supply chain is very advantageous. Attacks in the third category, remote attacks, do not gain as much from having an insider (of either type) since they're executed from completely outside of the system.

The fifth and final behavioral indicator used in the UAV example is manpower/time, which covers a combination of number of people involved and length of time required to execute the attack. These two variables are combined here because there's a relatively simple conversion factor between the two: one adversary might be able to accomplish the attack action with a large number of people working in a very condensed time frame, while another could do the same work with fewer people over an extended period of time. Manpower is different from the previous behavioral indicators in one key way. When aggregating the individual values into one value encompassing an entire attack scenario for the first four variables, the maximum of each value is taken (since doing three actions each with a Low-Med level of Attack-Specific Technical Ability still just requires a Low-Med level of Attack-Specific Technical Ability). On the other hand though, when calculating a cohesive score for the required manpower for an attack scenario, the values must be summed (doing three actions each with a Low-Med level of manpower requires three times that).

To assess these values for the set of leaf nodes, a Behavioral Indicator Variables Assessment Table was created (shown below in Table 4). This table can be reproduced for each of the 55 nodes and the leaf node name can be inserted in the first line for easy identification; members of the green team are then tasked with completing the tables.

Table 4. Behavioral Indicator Variables Assessment Table

Leaf Node Name					
Design Knowledge	Attack-Specific Technical Ability	Resources	Insider Presence (Operational)	Insider Presence (Supply Chain)	Manpower
□ Low □ Low-Med □ Medium □ Med-High □ High	□ Low □ Low-Med □ Medium □ Med-High □ High	□ Low □ Low-Med □ Medium □ Med-High □ High	□ Low □ Low-Med □ Medium □ Med-High □ High	□ Low □ Low-Med □ Medium □ Med-High □ High	□ Low □ Low-Med □ Medium □ Med-High □ High

Several different levels of granularity for the behavioral indicator scales were considered before deciding on a five point Likert-style scale. Using three (Low, Medium, and High) did not allow enough variation while seven (Very Low, Low, Low-Med, Medium, Med-High, High, and Very High) was too

many choices and caused the participants to become overwhelmed and revert towards the simplest three (Low, Medium, and High) in many cases.

While the five categorical assessments for each node are carried into step four and used to prune the attack tree, individual decision makers or project teams may decide that these values are not adequate on their own. For this reason, an optional notes section was also included as part of the assessment procedure to document the information the green team used to justify the five value selections. While the necessary thought process and information that may be used to select a value for each behavioral indicator will be different for each individual participant and will vary from project to project, the following seven topics were included for this application and can be used as a template which can be modified as needed for future applications:

1. Description of Attack – the leaf node name may mean different things to different individuals, depending on their background, so it is not sufficient on its own
2. Availability of Design Information – the presence of available open source and/or classified information about the system influences the overall difficulty of the attack
3. Presence of Existing Exploits – whether or not similar attacks have been successfully executed previously affects the likelihood that another exploit will be successful
4. Insider Attack Control Measures and Need for Insider – comments regarding if an insider's presence is needed or if the target system has measures in place to protect again such an attack
5. Unique Ability/Equipment Required – specification of any particular skillsets, tools, software packages, etc. that are necessary to complete the attack
6. Ease of Implementation – general comments regarding difficulty of the attack action that may not have been captured elsewhere
7. Additional Comments

An example node assessment worksheet for one of the nodes from the "Concealed Major Attack Trajectory" attack tree is shown in Figure 15.

Embedded Attack to Change Waypoints On-Board				
Design Knowledge	Attack-Specific Technical Ability	Resources	Necessary Insider	Manpower
☐ Low	☐ Low	☐ Low	☐ Low	☐ Low
☐ Low-Med	☐ Low-Med	✖ **Low-Med**	✖ **Low-Med**	☐ Low-Med
☐ Medium	☐ Medium	☐ Medium	☐ Medium	☐ Medium
✖ **Med-High**	✖ **Med-High**	☐ Med-High	☐ Med-High	✖ **Med-High**
☐ High	☐ High	☐ High	☐ High	☐ High

1. **Description of Attack:** Wormed Malware – Malware is "piggybacked" to Autopilot firmware package during upload of firmware during development or product upgrade phase. Malware changes waypoints based on "user defined" trigger. Malware can be based on Stack overflow exploit redirecting autopilot execution on a return address from function call, or redirected library call, or trampoline register call.
    a. Origin of Attack – Where does the attack gain ingress to the system? Gains access from the development or upgrade phase.
    b. Vulnerability – What is the nature of the vulnerability(s) to be exploited? Outside the Autopilot: Compromised development station, compromised SDE (Software Development Environment), Compromised Linker/Loader, Compromised uploader communications link. Inside the Autopilot: No bounds checking on data arrays, no safe libraries, no checks on exception handlers, no pointer protection checks.
2. **Availability of Design Information:**
    a. Open source, inferred from existing open source. Yes, most of the information can be gathered from open source.
    b. Proprietary, guarded source, and classified. Some guarded information may be needed to determine how to "piggyback" the malware into the firmware.
3. **Presence of Existing Exploits:**
    a. Are there attack methods, approaches, malware "in the wild or published" that could be used to craft the attack? Yes, most aspects of the attack can be crafted from known exploit methods.
4. **Insider Attack control measures and need for an Insider:**
    a. Yes/No – unknown, company specific.
    b. Control measures used – Unique, Standard, Simple, none
5. **Unique Ability/Equipment Required:**
    a. Specialized needs for carrying out phases of attack – Symbolic Interactive Debugger to analyze the firmware would be helpful.
6. **Ease of Implementation:** Would require a skilled team consisting of the following: Cyber Binary analyst, Embedded systems analyst, Network Exploit Analyst, MS windows expert.
7. **Additional Comments:** Overall difficulty would be medium/high, requiring a skilled set of people usually found in a well-organized cyber team. Manpower Time to investigate, access, design, develop, and implement the attack would be about ½ -1 man-year.

Figure 15. Example of Assessment Results for an Individual Leaf Node

With 55 leaf nodes and 5 behavioral indicators, the leaf node assessment process involved the group making a total of 275 judgments about the difficulty of the attack actions in different regards. For the UAV navigation system example, each individual on the red team completed the full set of the assessments. This worked well for the case study scenario because it provided more feedback on the methodology, but it was recognized to be unrealistic in practice due to an individual's deterioration of attention after a certain point in the process. While having multiple people provide assessments for the same leaf nodes provides more detailed analysis and confidence in the values assigned, it greatly increased the burden required of each participant. For future applications, the possibility of dividing the tree up to have individuals assess portions based on their specific knowledge base was discussed. Each organization will have to consider the number of leaf nodes in the tree and weigh the tradeoff between decreasing attention and more detailed analysis to determine the number of assessments each individual participant is tasked to complete. The assessment process was certainly time consuming and required knowledgeable individuals that were willing to take the necessary time to thoughtfully consider each assessment but that also understood that the process required making some assumptions and "best-guesses" in some cases.

Due to the repetitive nature of the nodes, several trends were discovered throughout the process. For instance, an embedded attack requires that the attacker needs to either have an identical system themselves to experiment with or know the system very well to understand the details of the implementation and ensure that their attack will work under a variety of different operational scenarios. This typically involves a higher need for the presence of an insider, more information about the design of the target system, and/or greater resources than a similar remote attack. Additionally, it was noted that, in many cases, an individual leaf node could be accomplished in multiple ways. As mentioned earlier, this could be addressed by expanding the tree even further to make each leaf node into a new subtree. While this is not recommended here (because of the potential to exponentially increase the scale and complexity of the tree), it is important to recognize that there may be several leaf nodes with varying levels of

associated difficulty. To best deal with this issue, the group chose to consider the easiest, most direct attack in cases where they could think of multiple attack actions for a single leaf node.

In addition to making judgments regarding the resources required for the various leaf nodes, step three also includes an assessment of the resources a particular threat actor is expected to possess. Using the same behavioral indicator variables that are used for assessing the nodes (shown again in Table 5 to reiterate their meaning in assessing an adversary), an adversary profile can be constructed.

Table 5. Behavioral Indicator Variables Names, Possible Values, and Meanings for Adversary Capability Assessment

Behavioral Indicator Name	Possible Values	Meaning for Adversary Assessment
Design Knowledge	• Low • Low-Med • Medium • Med-High • High	To what level do you expect the attacker to have access to design knowledge of the system?
Attack-Specific Technical Ability	• Low • Low-Med • Medium • Med-High • High	What level of attack- specific technical ability do you expect the adversary to possess?
Resources	• Low • Low-Med • Medium • Med-High • High	What level of resources (i.e.: facilities and equipment) do you expect the attacker to have access to?
Insider Presence (Operational)	• Low • Low-Med • Medium • Med-High • High	What is the likelihood that the attacker has an insider present in the operational phase of the system?
Insider Presence (Supply Chain)	• Low • Low-Med • Medium • Med-High • High	What is the likelihood that the attacker has an insider present at some point in the supply chain?
Manpower	• Low • Low-Med • Medium • Med-High • High	What level of manpower (i.e: time and number of people) do you expect the attacker to possess?

These profiles are then used to prune the attack tree – a process that eliminates attack scenarios (and thus, leaf nodes) to create a reduced tree that is specific to an individual threat actor. In future applications, a single adversary profile would be created for the threat actor that the group was most concerned with. For this example however, it was decided that there were four potential classes of adversaries, each of which could have vastly different resources, skills, and motivations, and all four adversary profiles were constructed in order to demonstrate differences among them. The four classes of attackers considered are:

1. Rogue Insider – an individual that has a specific insider connection to the UAV system (i.e.: an operator or avionics engineer) and has decided to take action against the system. He/she has specific knowledge about or access to the system, but their knowledge is more likely to be narrow in scope and they are severely limited in regards to manpower. The individual may not have a particular purpose in attacking the system, but rather a desire to get back at the organization and cause damage however possible.
2. Terrorist Group – a highly-motivated, moderately sized group that is working to compromise the system for a particular reason (i.e.: preventing the platform from collecting surveillance information in a specific geographic area to prevent the detection of some activity).
3. Nation State – a country with considerable resources, manpower, and skills that is acting against the U.S. defense (also, with a particular motivation)
4. Criminal Cyber Group – a moderately sized mercenary group whose goal is to develop some attack capability with the intention to profit off of that ability (either by selling that capability to a terrorist group or nation state or using it for extortion purposes against the U.S. defense). These groups typically select a target system because of their previously existing access or knowledge, so these values are on the higher end of the scale.

Table 6 shows the assessments made for each threat actor profile in regards to the six behavioral indicator variables.

Table 6. Adversary Profiles Showing Assessed Behavioral Indicator Levels

	Rogue Insider	Terrorist Group	Nation-State	Criminal Cyber Group
Design Knowledge	Med-High	Medium	Med-High	High
Attack-Specific Technical Ability	Med-High	Medium	Med-High	High
Resources	High	Medium	Med-High	Medium
Insider Presence (Operational)	High	Med-High	Medium	Medium
Insider Presence (Supply Chain)	Low-Med	Medium	High	Medium
Manpower	Low	Med-High	High	Med-High

Several interesting points can be made after comparing these profiles to the attack tree and creating a pruning tree based on each adversary's expected capabilities. For instance, applying the "Rogue Insider" profile eliminated every single leaf node, which implies that an insider individual isn't capable of executing the Concealed Major Trajectory Deviation attack. This is entirely due to their limited manpower. Even considering the insider's specific knowledge about the system, an individual working on his/her own or even with a small group, simply does not not have the ability to complete the attack – they would either need more people to assist with the workload or a much longer period of time to plan and execute the attack. Additional pruning trees were created by adjusting the Rogue Insider's manpower criteria to both Low-Med and Medium with no change in the results: the insider was still not capable of executing the attack. It should be noted though that the insider may be able to complete the unconcealed version of the attack. Since this eliminates the need for the right-hand portion of the tree (actions focused on concealing the attack and lowering the detectability), there are fewer tasks the attacker needs to complete and their manpower resources are enough for a small subset of the leaf nodes.

Applying the second threat actor profile did not completely eliminate the set of leaf nodes, but it did significantly reduce it. From the pruning process based on the six behavioral indicators, it was found that the terrorist group was not able to change the operator display software or change the airplane position reports; they were only able to conceal the flight trajectory by creating distraction event(s) for the operator. However, attack scenarios can also be assessed for overall detectability after the individual leaf

nodes are pruned. If one scenario is found to be within the adversary's capabilities according to the six behavioral indicators but beyond their tolerance level for the attack being detected and prevented, its corresponding attack actions can be eliminated. "Create Distraction Event(s) for Operator" is a very risky action; i.e.: it is very possible that the operator will not be adequately distracted and the attack will be detected and prevented. Since the terrorist group is very much invested in the outcome of their attack and has more of a specific purpose for the exploit than the rogue individual, it was determined that this node had too much inherent risk and the terrorist group would likely not attempt it. Again, it is important to note that the terrorist group may be able to successfully execute an attack from one of the other two trees. Their motivations may rule out the unconcealed major trajectory change, but the minor trajectory deviation could still be a possibility for them.

The pruning tree produced with the Nation State adversary profile was the largest of the four. Even though they are the least likely to have an insider, their high values for the remaining behavioral indicators more than make up for that hypothetical shortcoming. Figure 16 below shows the Nation State pruning tree, which includes 20 leaf nodes and 31 attack scenarios.

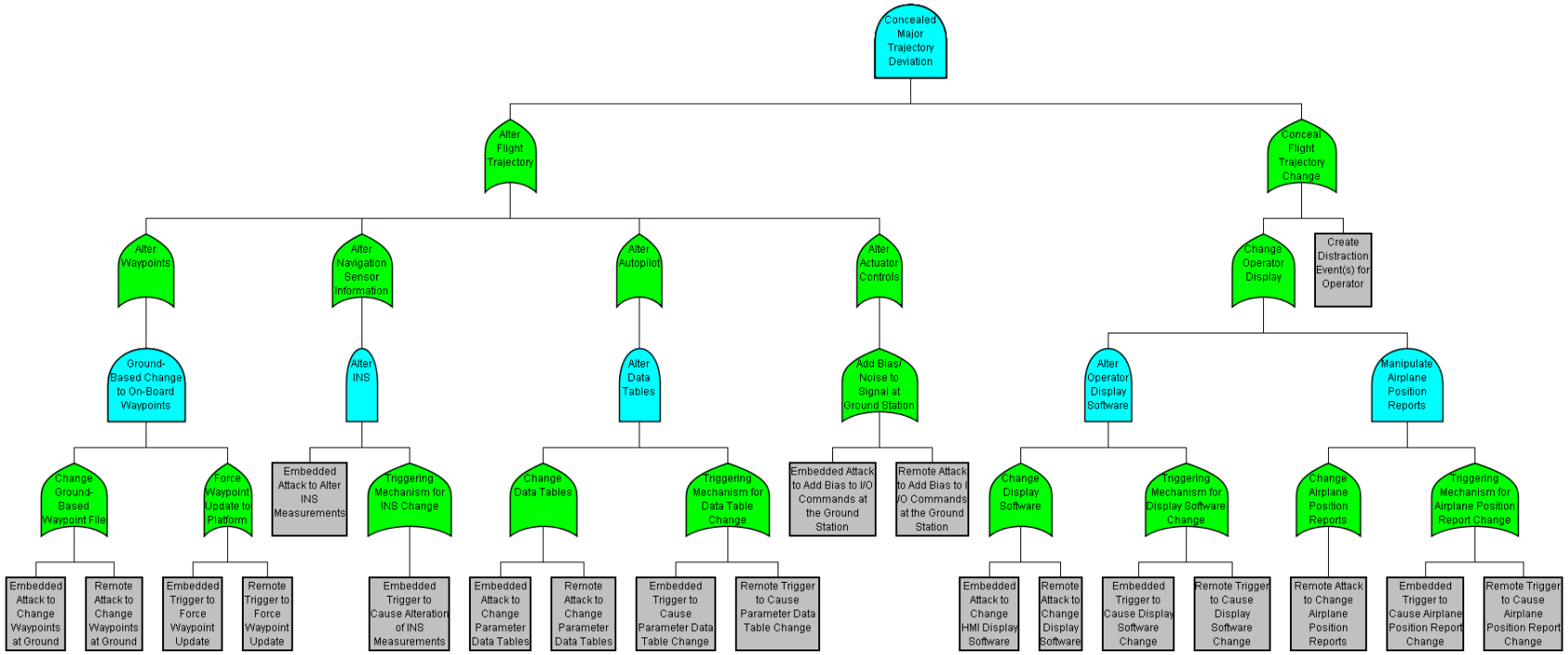

Figure 16. Nation State Pruning Tree

The fourth and final threat actor profile (for the criminal cyber group) also had a large number of leaf nodes remaining after the pruning process. Even though their expected resources and manpower were lower than the Nation State, the fact that attacks are selected based on the group's pre-existing knowledge and ability, meant that those values were higher. Figure 17 below shows the Criminal Cyber Group pruning tree, which includes 16 leaf nodes and 19 attack scenarios.

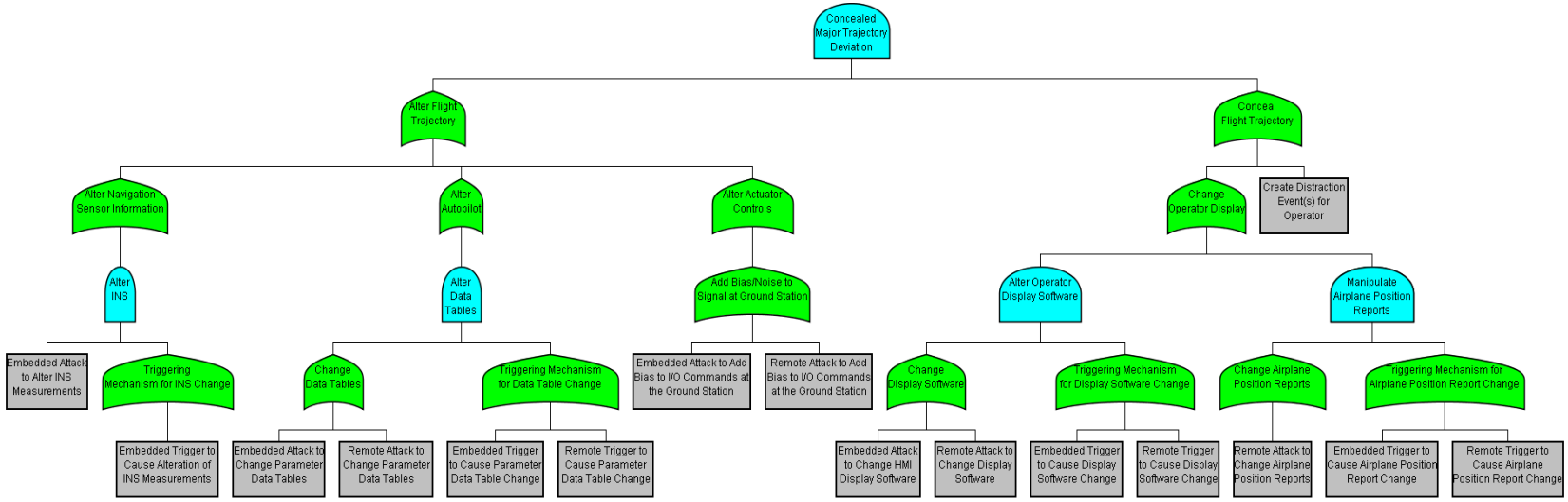

Figure 17. Criminal Cyber Group Pruning Tree

In a real application scenario, a single adversary profile would be constructed and the pruning tree associated with that specific threat actor would be used going forward. In this example case, the results from pruning with the Nation State profile were used. The final list for the UAV navigation system attack example consists of:

1. Embedded Attack to Change Waypoints at Ground
2. Remote Attack to Change Waypoints at Ground
3. Embedded Trigger to Force Waypoint Update
4. Remote Trigger to Force Waypoint Update
5. Embedded Attack to Alter INS Measurements
6. Embedded Trigger to Cause Alteration of INS Measurements
7. Embedded Attack to Change Parameter Data Tables
8. Remote Attack to Change Parameter Data Tables
9. Embedded Trigger to Cause Parameter Data Tables
10. Remote Trigger to Cause Parameter Data Table Change
11. Embedded Attack to Add Bias to I/O Commands at the Ground Station
12. Remote Attack to Add Bias to I/O Commands at the Ground Station
13. Embedded Attack to Change HMI Display Software
14. Remote Attack to Change HMI Display Software
15. Embedded Trigger to Cause Display Software Change
16. Remote Trigger to Cause Display Software Change
17. Remote Attack to Change Airplane Position Reports
18. Embedded Trigger to Cause Airplane Position Report Change
19. Remote Trigger to Cause Airplane Position Report Change

It should be noted that the "Create Distraction Event(s) for the Operator" node was eliminated from the set to be used in step 4 because, similarly to the Terrorist Group, the Nation State was not willing to tolerate the higher likelihood of the attack being detected and prevented. Also, it is interesting to note that

there are no insider attack nodes remaining (only embedded and remote) since the Nation State is not expected to have a high likelihood of an insider present during the operational phase of the system.

d. Step 4

This reduced set of 19 nodes can be compared with the existing portfolio of design patterns to determine which design patterns are the most applicable. A design pattern can be considered applicable for several reasons. It can make a leaf node more difficult, uncertain, or expensive, or make it so the completing the action requires that the adversary have specialized skills or equipment. Additionally, a design pattern may be applicable if it increases the likelihood that the attack can be detected and prevented or decreases the consequence on the defense team given the attack is still successful.

The 19 remaining leaf nodes can be grouped into six categories: (1) ground station based waypoint change, (2) change of INS measurements, (3) change of parameter data tables, (4) addition of bias/noise through the I/O commands at the ground station, (5) spoofing the HMI operator display through a software change, and (6) manipulation of the airplane position report. Table 7 shows several possible design patterns that were determined to be applicable for each of the remaining six attack types.

Table 7. Applicable Design Patterns for Each Attack

Attack Type	Design Pattern	Detailed Description of DP Functionality
1. Ground Station Based Waypoint Change	Parameter Assurance	Typically, there will be a pre-loaded flight plan based on the mission. Parameter Assurance compares the waypoints input at the ground to the table of values in the system to check for large, unexpected deviations.
	Data Consistency Checking	A change of the waypoints at the ground station needs to follow a step order of steps. Data Consistency Checking looks to see where the change originated from to verify that it was initiated by the operator.
2. Change of INS Measurements	Diverse Redundancy	Diverse Redundancy involves the implementation of additional INS devices, from diverse manufacturers/suppliers.
	Physical Configuration Hopping	Physical Configuration Hopping involves hopping between multiple INS components at a pre-determined interval.
	Verifiable Voting	Voting involves comparing the values returned by multiple INS devices to identify and isolate a compromised INS.
3. Change of Parameter Data Tables	Data Consistency Checking	A change of the parameter data tables needs to follow a step order of steps. Data Consistency Checking looks to see where the change originated from to verify that it came from a trusted source.
	Parameter Assurance	Parameter Assurance compares the parameter data table values to a table of preexisting "gold standard" of flight control values in the system to check for large, unexpected deviations.
4. Addition of Bias/Noise Through I/O Commands at Ground	State Estimation	State Estimation uses existing mechanisms in the system to estimate other state variables in the system. These values can be used indirectly to infer what the state in question should be.
5. Spoofing the HMI Operator Display Through Software Change	Data Consistency Checking	A change of the HMI display software needs to follow a step order of steps. Data Consistency Checking looks to see where the change originated from to verify that it came from a trusted source.
	Parameter Assurance	Parameter Assurance involves using a back-up system (possibly considerably more rudimentary than the main operator display) to collect the same information as

		the main display. These values may not be displayed to the operator, but the system compares the main display values to those collected by the back-up display system to check for deviations.
	State Estimation	State Estimation uses existing mechanisms in the system to estimate other state variables in the system. These values can be used indirectly to infer what the operator display should be showing.
6. Change of Airline Position Report	Diverse Redundancy	Diverse Redundancy involves the implementation of additional radio devices, from diverse manufacturers/suppliers (the radio is used as an example here because it is the source that sends the position information from the platform to the ground station, but diverse redundancy could be added to another device earlier in the process to accomplish the same outcome).
	Physical Configuration Hopping	Physical Configuration Hopping involves hopping between multiple radio components at a pre-determined interval (see note regarding diverse redundancy above: the radio is only an example for one device where physical configuration hopping could be implemented).
	State Estimation	State Estimation uses existing mechanisms in the system to estimate other state variables in the system. These values can be used indirectly to infer what the state in question should be. State Estimation would only work in this situation if the change caused by the adversary did not affect all of the state estimates included in the Airline Position Report (the design pattern relies on having some secure estimates to use in order to infer less secure estimates).

At this point, these design patterns can be inserted back into the system relational influence diagram constructed in the first step to understand how the different defensive strategies complicate the actions required by the adversary and how they interact to provide multidimensional coverage of the system. Several examples are discussed here to demonstrate the type of insight that can be gained in this step.

The first defensive strategy to consider is the addition of Parameter Assurance inplemented on the waypoints stored at the ground station. This design pattern works by maintaining access to a pre-loaded flight plan associated with the mission and comparing the waypoints at the ground to these stored values to check for large, unexpected deviations from the expected waypoints. In the initial minimal defense system depicted in the influence relational diagram from step one, if the attacker wanted to execute a ground-based attack of the waypoints, they had to conduct an attack to change the values at the ground and also create a trigger to force the waypoints to update to the platform (both of which could be embedded within the system through supply chain infiltration or done remotely.) After the hypothetical implementation of this design pattern, the attacker still has to do both of those actions, but they now also need to consider two additional elements. Adding Parameter Assurance inserts two new nodes into the system influence relational diagram: the Preloaded Flight Plan Values and the Parameter Deviation Checking Mechanism. In order to successfully execute the attack with Parameter Assurance implemented, the adversary still needs to alter the ground station waypoints but also now needs to do one of the following:

1. Change the preloaded table of expected waypoint values associated with the flight plan to match their manipulated waypoints values so that the functioning comparison mechanism will return that the values are the same
2. Alter the parameter deviation checking mechanism so that it will not report that there is a large deviation between the two sets of waypoints, even though there is

Figure 18 shows the original system influence relational diagram updated with the defensive strategy of Parameter Assurance implemented. In addition to the actions of the human operator and the status of the HMI I/O controls, the ground station waypoints are also now dependent on the outcome of the Parameter Assurance protection. In turn, the value returned by Parameter Assurance is influenced by the preloaded flight plan values and the parameter deviation checking mechanism.

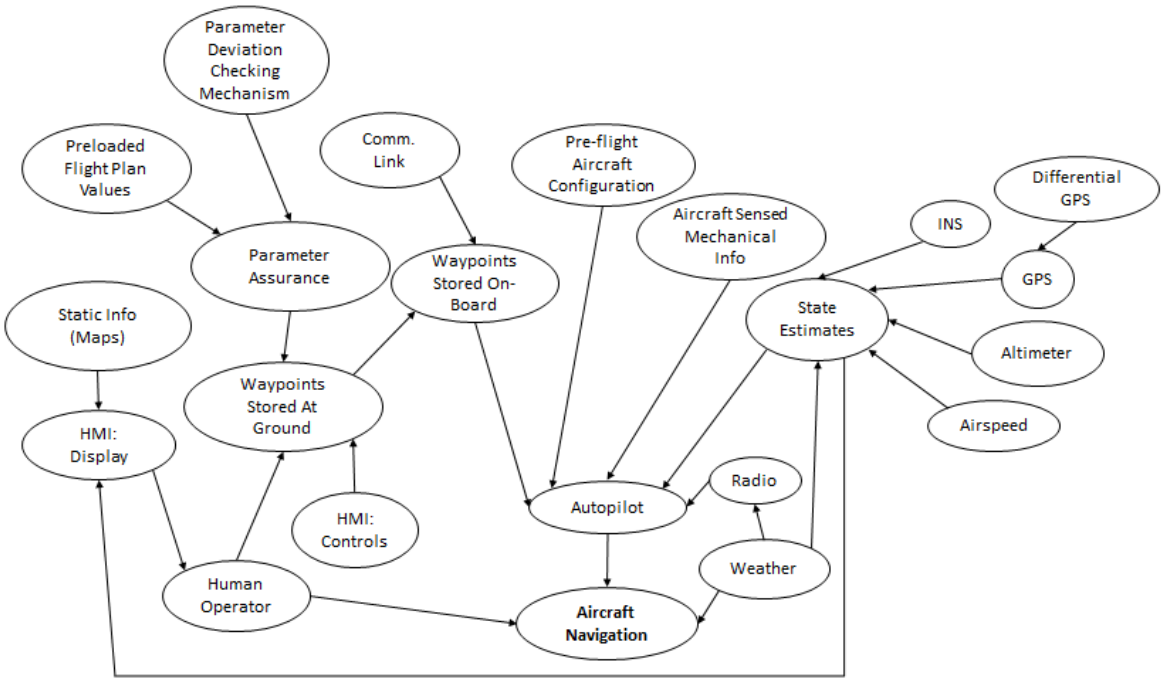

Figure 18. Updated System Influence Relational Diagram with Parameter Assurance Implemented on Waypoints Stored At Ground

It's important to note that implementing Parameter Assurance here (or any other design pattern anywhere in the system) does not eliminate the possibility of a ground-based attack on the waypoint values (or continuing in the general sense, any other attack on the system); the defensive strategies only complicate the adversary's path to a successful exploit. This notion of a design pattern increasing the difficulty for the attacker is what is captured by the Markov blanket concept as applied to the system influence relational diagram. Figure 19 shows the Markov blanket of the Parameter Assurance design pattern as applied here. With "Parameter Assurance" as the node in question, its parents are "Preloaded Flight Plan Values" and "Parameter Deviation Checking Mechanism," its child is "Waypoints Stored at Ground," and "Human Operator" and "HMI: Controls" are the parents of its child (or spouses).

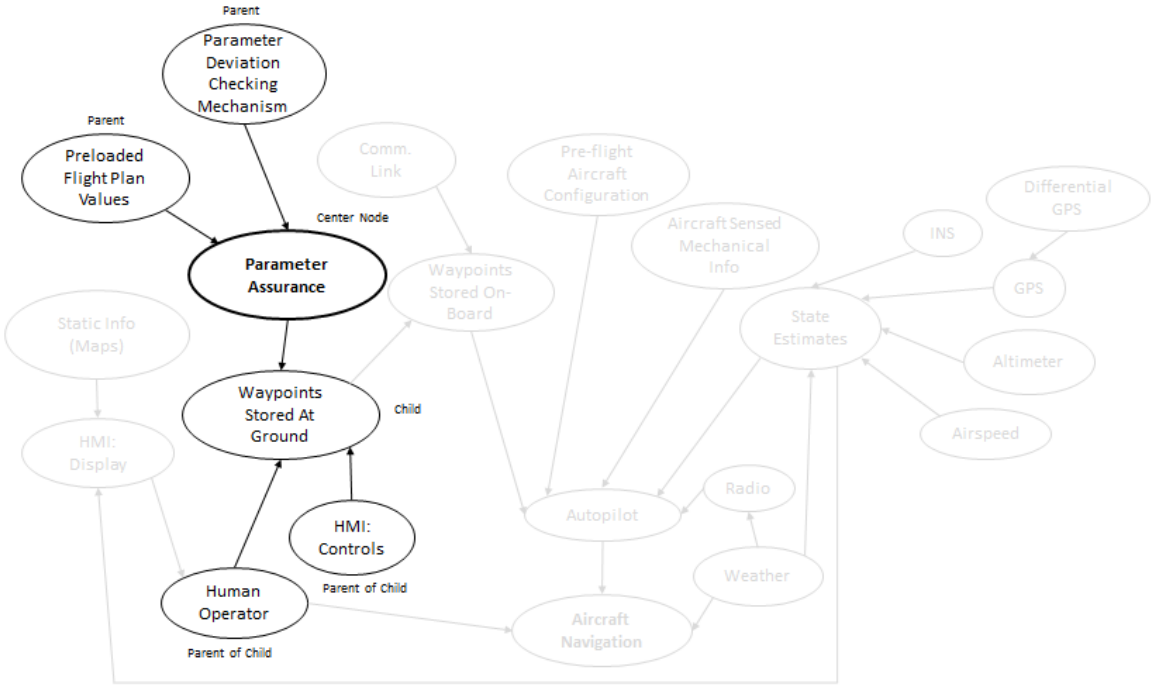

Figure 19. Markov Blanket of Parameter Assurance Implemented on Waypoints Stored At Ground

At this point, the two new nodes that have been added because of the implementation of Parameter Assurance can be assessed for adversary difficulty regarding the same six behavioral indicator variables defined in step three. Table 8 below shows these assessments for "Preloaded Flight Plan Values" and "Parameter Deviation Checking Mechanism."

Table 8. Assessment of New Nodes "Preloaded Flight Plan Values" and "Parameter Deviation Checking Mechanism" Impact to Adversary

	Preloaded Flight Plan Values	Parameter Deviation Checking Mechanism
Design Knowledge	Medium	Med-High
Attack-Specific Technical Ability	Low-Med	Med-High
Resources	Low-Med	Med-High
Insider Presence (Operational)	Low	Low-Med
Insider Presence (Supply Chain)	Medium	Med-High
Manpower/Time	Low-Med	Medium

In this case, changing the Preloaded Flight Plan Values was assessed to require less knowledge, ability, resources, insider presence, and manpower than altering the Parameter Deviation Checking Mechanism. Thus, with the assumption that the attacker will act rationally and choose the simpler of two options, the team can assume that the implementation of Parameter Assurance will force the adversary to manipulate the preloaded flight plan values. This means that the design team's choice to implement Parameter Assurance complicates the adversary's path by the addition of one attack action which requires Medium design knowledge, Low-Med attack-specific technical ability, Low-Med resources, Low operational insider presence, Medium supply chain insider presence, and Low-Med manpower/time.

As a second example, another possible design pattern that can be considered here is the combination of Diverse Redundancy and Physical Configuration Hopping (neither design pattern is considered individually here because diversity provides minimal benefit on its own and hopping cannot be implemented without first having multiple, redundant components). Diverse Redundancy here works by implementing additional INS devices from diverse manufacturers and/or suppliers and Physical Configuration Hopping works by hopping between these components at a predetermined interval. In the initial system as captured in the system influence diagram from step one, if the attacker wanted to alter the INS measurements, they could embed the ability to both change the values and a triggering mechanism to turn the change on/off on-board the platform. After the implementation of the design patterns (say the addition of two more diverse INS devices for a total of three that are hopped between), the adversary must now also do one of the following:

1. The adversary can choose to bypass the hopping mechanism, but this means that they must alter the values of all three INS devices so the manipulated data is consistent regardless of the hopping interval and which device is being used at any particular moment.
2. Otherwise, if the attacker is unable or unwilling to alter three diverse components, the attacker must understand the mechanisms of the hopper and the order/interval that is being used to determine which device is being used. The adversary must still alter the value of one INS (similarly to the original attack, this will be embedded and will involves both the change

itself and the triggering mechanism), and then they will use their knowledge of the hopping mechanism to attack the INS when they know it will be the one in use. The adversary can choose which of the devices to alter: (1) INS 1, (2) INS 2, or (3) INS 3.

Figure 20 shows the original system influence relational diagram updated with the defensive strategies of Diverse Redundancy and Physical Configuration Hopping implemented. Here, the final INS value that is used in the set of state estimates (or the airplane position report) is fully dependent on the Diverse Redundancy and Physical Configuration Hopping. In turn, the value returned by Diverse Redundancy and Physical Configuration Hopping is influenced by the values of the three individual INS devices and the hopping mechanism/interval.

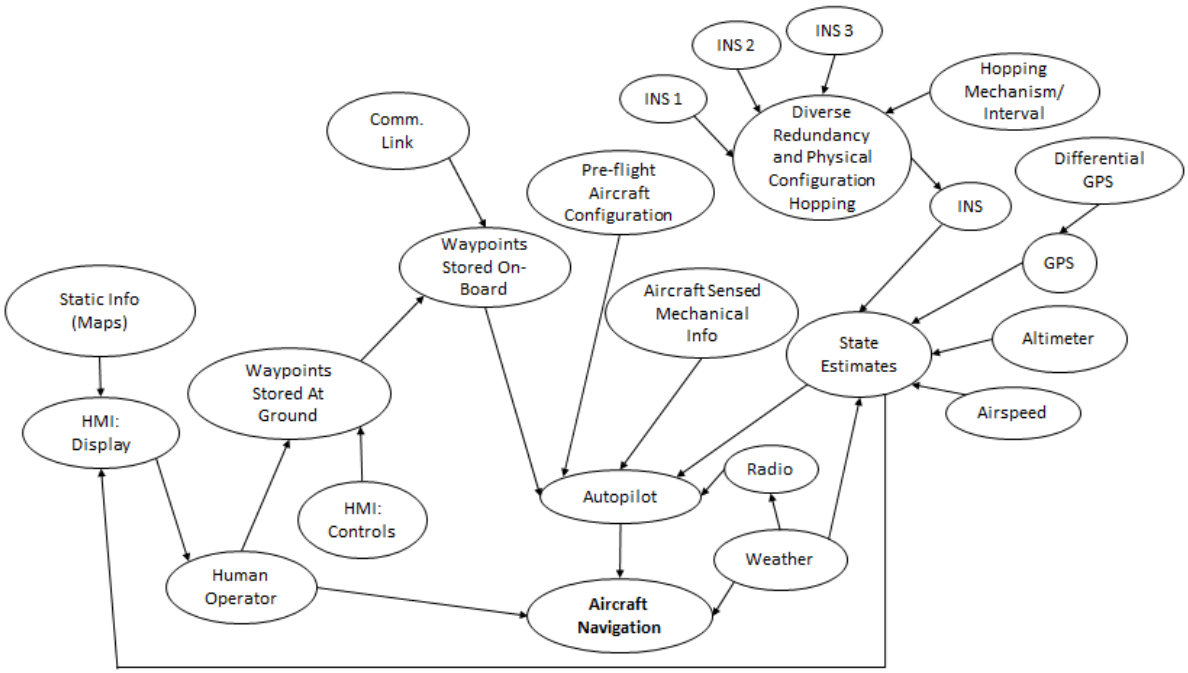

Figure 20. Updated System Influence Relational Diagram with Diverse Redundancy and Physical Configuration Hopping Implemented on the INS

Figure 21 shows the Markov blanket of the Diverse Redundancy and Physical Configuration Hopping design patterns as applied here. With "Diverse Redundancy and Physical Configuration Hopping" as the node in question, its parents are "INS, 1" "INS 2," "INS 3," and "Hopping Mechanism/Interval," and its

child is the final INS value (which is then feed into the set of state estimates). Since the final INS values are only dependent on the information returned from the Diverse Redundancy and Physical Configuration Hopping design patterns, there are no additional parents of children (or spouses) here.

Figure 21. Markov Blanket of Diverse Redundancy and Physical Configuration Hopping Implemented on the INS

Again, the new nodes that were added based on the implementation of Diverse Redundancy and Physical Configuration Hopping can be assessed for adversary difficulty here. Table 9 shows these assessments for "INS 1," "INS 2," "INS 3," and "Hopping Mechanism/Interval." In this case, without knowing specifics about the individual INS makes and models considered, corrupting any one of the three is assumed to require the same resources as corrupting the initial INS device which was assessed in step three.

Table 9. Assessment of New Nodes "INS 1, 2, and 3" and "Hopping Mechanism/Interval" Impact to Adversary

	INS 1, INS 2, INS 3	Hopping Mechanism/Interval
Design Knowledge	Med-High	High
Attack-Specific Technical Ability	Med-High	Low-Med
Resources	Medium	Med-High
Insider Presence (Operational)	Low-Med	Low-Med
Insider Presence (Supply Chain)	Med-High	Low-Med
Manpower/Time	Medium	Med-High

In this case, neither manipulating the individual INS devices nor exploiting the hopping mechanism dominates over the other to become an obvious choice for the adversary to select. Using the mechanics of the hopper to their advantage by only attacking one INS but ensuring that it is the one in use based on the hopping interval and order requires the adversary have a high level of design knowledge about the system. On the other hand, attacking the three INS devices requires a higher likelihood of having an insider present at some point in the corrupted supply chain since the adversary now has to execute three distinct embedded attacks. One final difference between the two is the level of manpower/time required for each new attack action. Understanding the hopping mechanism/interval has an associated manpower/time demand of Med-High because of the amount of time required to adequately research the hopper to the point where the adversary felt confident that they understood its logistics in a variety of operational scenarios. If this action is chosen, the adversary still has to corrupt one INS (which has an associated Medium manpower/time demand). If the adversary chooses to ignore the hopper and instead manipulate the three diverse INS devices, they now need to have the manpower/time capabilities to complete three tasks, each with an associated manpower/time demand of Medium. Since neither attack action option is dominating, the decision of which action would be chosen is very much dependent on the adversary under consideration. Since the Nation State adversary profile was selected for the completion of this case study, exploiting the hopping mechanism/interval is ruled out (because its required high design knowledge is above the Nation State's expected Med-High capabilities). If the Criminal Cyber Group profile had been selected instead, altering the values of all three diverse INS devices would have been ruled out because the manpower/time demand of Medium X 3 would have been beyond their capabilities.

Thus, for this example, the design team's choice to implement Diverse Redundancy and Physical Configuration Hopping complicates the adversary's path by the addition of two new attack actions which both require Med-High design knowledge, Med-High attack-specific technical ability, Medium resources, Low-Med operational insider presence, Med-High supply chain insider presence, and Medium manpower/time.

e. Step 5

Step five is focused on evaluating the potential design patterns from step four in regards to their impact on the defensive team. This impact can be categorized into three criteria: (1) implementation cost, (2) lifecycle cost, and (3) collateral system impacts. Similarly to the behavioral indicator variables assessed in step three, a five point Likert scale (with possible values of Low, Low-Med, Medium, Med-High, and High), along with an optional notes section for justifying comments, was used here for ease of evaluation. Table 8 shows the levels assessed for each design pattern regarding implementation cost, lifecycle cost, and collateral system impacts.

Table 10. Implementation Cost, Lifecycle Costs, and Collateral System Impacts Associated with Each Applicable Design Pattern

Attack Type	Design Pattern	Implementation Cost	Lifecycle Costs	Collateral System Impacts
Ground Station Based Waypoint Change	Data Consistency Checking	Medium	Low-Med	Low
	Parameter Assurance	Low-Med	Medium	Medium
Change of INS Measurements	Diverse Redundancy and Physical Configuration Hopping	Medium	Medium	Medium
	Diverse Redundancy and Verifiable Voting	Low-Med	Medium	Low
Change of Parameter Data Tables	Data Consistency Checking	Low-Med	Med-High	Low
	Parameter Assurance	Med-High	Medium	Med-High
Addition of Bias/Noise Through I/O Commands at Ground	State Estimation	Medium	Low-Med	Low
Change of the HMI Display Software	Data Consistency Checking	Medium	Low-Med	Medium
	Parameter Assurance	Med-High	Med-High	Med-High
	State Estimation	Medium	Low-Med	Low-Med
Change of Airline Position Report	Diverse Redundancy and Physical Configuration Hopping	Med-High	Medium	Med-High
	State Estimation	Medium	Low-Med	Medium

With just 12 strategies to evaluate in regards to three different criteria, the 36 assessments required here was much more reasonable than the set of leaf nodes assessments conducted in step three. One comment that was voiced about the process is that it was easier to assess the Implementation Costs and Collateral System Impacts than it was the Lifecycle Costs. This was due to the fact that the Implementation Cost and Collateral System Impact values are fully dependent on the design pattern itself, while the Lifecycle Costs are dependent on a combination of the design pattern and the system it is being applied to. Confidently assessing the Lifecycle Costs would require the participants to have more knowledge about the system itself. A second point to note is that the system environment greatly influences the Collateral Systems Impact values. Environments with more constraints (such as the UAV platform) will typically have higher associated collateral system impacts than those with fewer constraints (like the ground station).

At this point, the set of possible defensive solutions can be reduced one more time. Based on the defensive budget and preferences regarding collateral system impacts, any design patterns that are beyond budget or have unacceptable collateral system impacts can be eliminated and do not need to be considered in the sixth and final step. For instance, if "Med-High" costs over the lifecycle of the system were deemed to be over budget and "Med-High" collateral system impacts were deemed to be unacceptable, four alternatives can be eliminated. The remaining set of possible defensive strategies is:

1. Data Consistency Checking implemented on the ground station waypoint file to prevent a ground-based waypoint change
2. Parameter Assurance implemented on the ground station waypoint file to prevent a ground-based waypoint change
3. Diverse Redundancy and Physical Configuration Hopping implemented on the INS to prevent the alteration of the INS measurements
4. Diverse Redundancy and Verifiable Voting implemented on the INS to prevent the alteration of the INS measurements

5. State Estimation implemented to prevent the addition of bias/noise through I/O commands at the ground station
6. Data Consistency Checking implemented on the HMI operator display at the ground station to prevent a change of the HMI display software
7. State Estimation implemented on the HMI operator display at the ground station to prevent a change of the HMI display software
8. State Estimation implemented to prevent the manipulation of the airplane position report

While any strategies eliminated at this point do not need to be fully discussed in step six, it is important to note that they have not been completely disregarded. Design patterns that are considered to have unacceptable impacts on the defense will be documented so they can be revisited at a later point if the budget or the team's views on certain collateral system impacts changes.

f. Step 6

Going into the final step, there are eight possible defensive strategies to consider, all of which increase the difficulty for a specific adversary to complete one of their most preferred attack actions while remaining at an acceptable impact to the defense. While eight is much more reasonable than the entire original set of possibilities, it is still more than what the design team can afford to implement. Step six involves all of the project participants coming together for a collaborative discussion focused on weighing the security trade-offs that exist among these remaining alternatives in order to select a subset to be implemented as part of the final solution. There are four factors that should be considered when piecing together the final cohesive security architecture: (1) budget, (2) coverage, (3) multi-dimensionality, and (4) asymmetry.

Budget is used as the initial prescreening criteria here. At this point, the project team as a whole decides that they would like to implement about a quarter of the remaining design patterns and are comfortable with the implementation costs associated with two or three Low-Med or Medium valued solutions. The idea of system coverage is used as a second prescreening filter. After completing all of the analyses throughout the framework, team members will probably have certain alternatives of interest in

the remaining set because of the areas of the system they protect or the types of attacks they protect against. For this example, it was determined that the team was more concerned with the flight trajectory change itself than the concealment of that change. Narrowing the coverage scope in this regard made sense because protecting against the change itself protects the system against all three valued attack trees (rather than just the Concealed Major Trajectory Deviation tree), and if the action to alter the flight trajectory is difficult enough that the attacker can't complete it, the actions to conceal it become unnecessary. Specifically, the team wanted to focus on the possibility that the adversary would alter the navigation via a ground-based waypoint change or alteration of the INS values on board the platform and thus wanted to further examine the options for protecting against these threats. Three possible solutions were constructed to be compared in a more detailed analysis:

1. Implementation of both parameter assurance and data consistency checking on the ground station waypoint file to prevent a ground-based waypoint change
2. Implementation of both diverse redundancy, physical configuration hopping, and verifiable voting on the INS to prevent the alteration of the INS measurements
3. Implementation of one defensive strategy on the ground waypoint station and one on the INS device (specifics determined by the analysis of architectures 1 and 2)

These three architectural solutions were analyzed in regards to each's impact on the adversary relative to its impact on the defense. Since there were six behavioral indicator variables used to assess the attack actions and construct the adversary profiles, the impact of a defensive strategy can be assessed across six different dimensions. Different defensive actions will increase the difficulty for an attacker in different areas, and an optimal solution can be constructed by combining strategies that complement each other in regards to the multidimensionality.

Figure 22 shows the updated system influence relational diagram with the implementation of two design patterns on the waypoint file at the ground station (the first sample solution architecture

constructed). The impacts associated with these two strategies (both to the adversary and to the defense) are listed in Table 11.

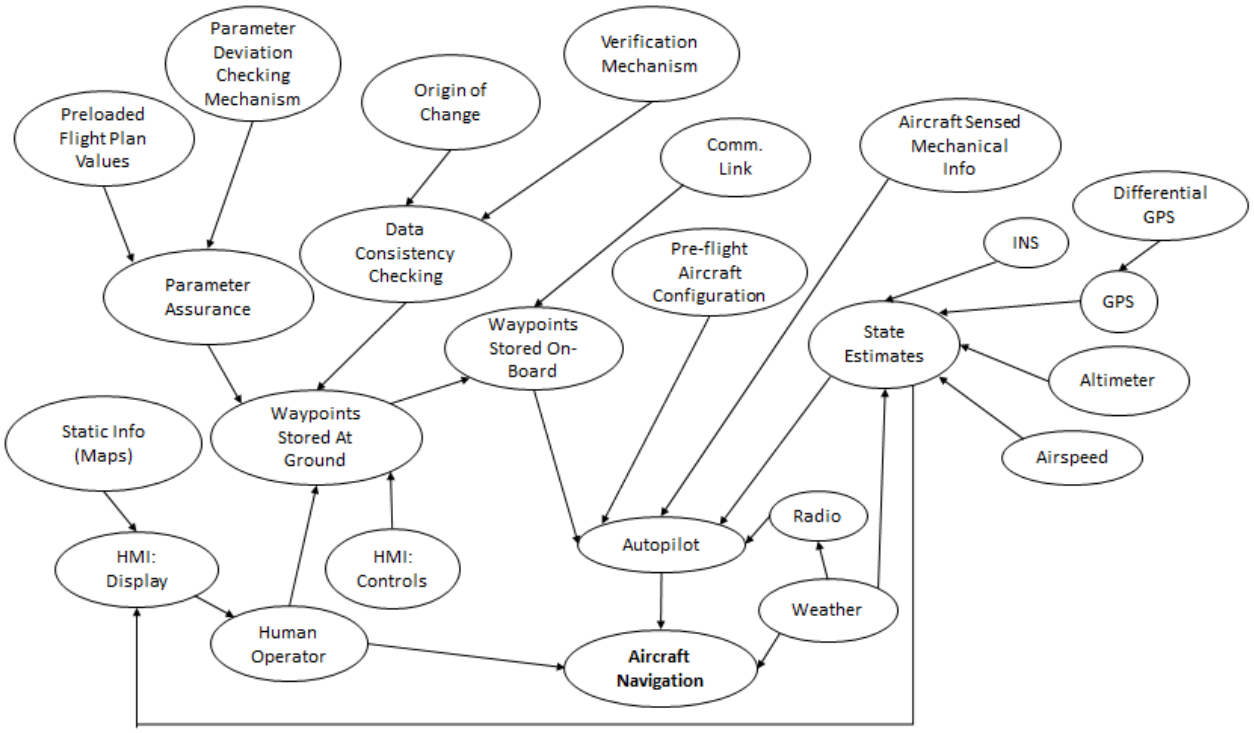

Figure 22. Updated System Influence Relational Diagram with Parameter Assurance and Data Consistency Checking Implemented on the Ground Station Waypoint File

Table 11. Impact to Adversary Relative to Impact to Defense for Parameter Assurance and Data Consistency Checking

Parameter Assurance		**Data Consistency Checking**	
Impact to Defense	**Impact to Adversary**	**Impact to Defense**	**Impact to Adversary**
Implementation Cost: Low-Med	Design Knowledge: Medium	Implementation Cost: Medium	Design Knowledge: Medium
Lifecycle Cost: Medium	Attack-Specific Technical Ability: Low-Med	Lifecycle Cost: Low-Med	Attack-Specific Technical Ability: Med-High
Collateral System Impact: Medium	Resources: Low-Med	Collateral System Impact: Low	Resources: Med
	Insider Presence (Operational): Low		Insider Presence (Operational): Medium
	Insider Presence (Supply Chain): Medium		Insider Presence (Supply Chain): Med-High
	Manpower/Time: Low-Med		Manpower/Time: Medium

Between the two protection strategies implemented on the ground station waypoint file, data consistency checking has a greater impact on the adversary. Both design patterns require the adversary to have a Medium level of Design Knowledge, but data consistency checking has a greater impact in regards to the other five behavioral indicators. Also, while parameter assurance had a lower upfront implementation cost, data consistency checking had a lower impact to the defense through the lifecycle of the system with lower lifecycle costs and collateral system impacts. Since data consistency checking has a greater impact on the adversary across the board than parameter assurance, there is no value of implementing both design patterns (if the team implements data consistency checking, there is no additional subsequent value gained from also implementing parameter assurance).At this point, the teams can discuss the two alternatives to determine which option best reverses the asymmetry: mainly, discussing the question of whether the lower implementation cost of parameter assurance is worth the higher impacts to the defense down the line and lesser impacts on the adversary across all six behavioral indicators.

Moving to the second sample architecture considered, Figure 23 shows the updated system influence relational diagram with the implementation of two defensive strategies on the INS components. The impacts associated with these two strategies (both to the adversary and to the defense) are listed in Table 12.

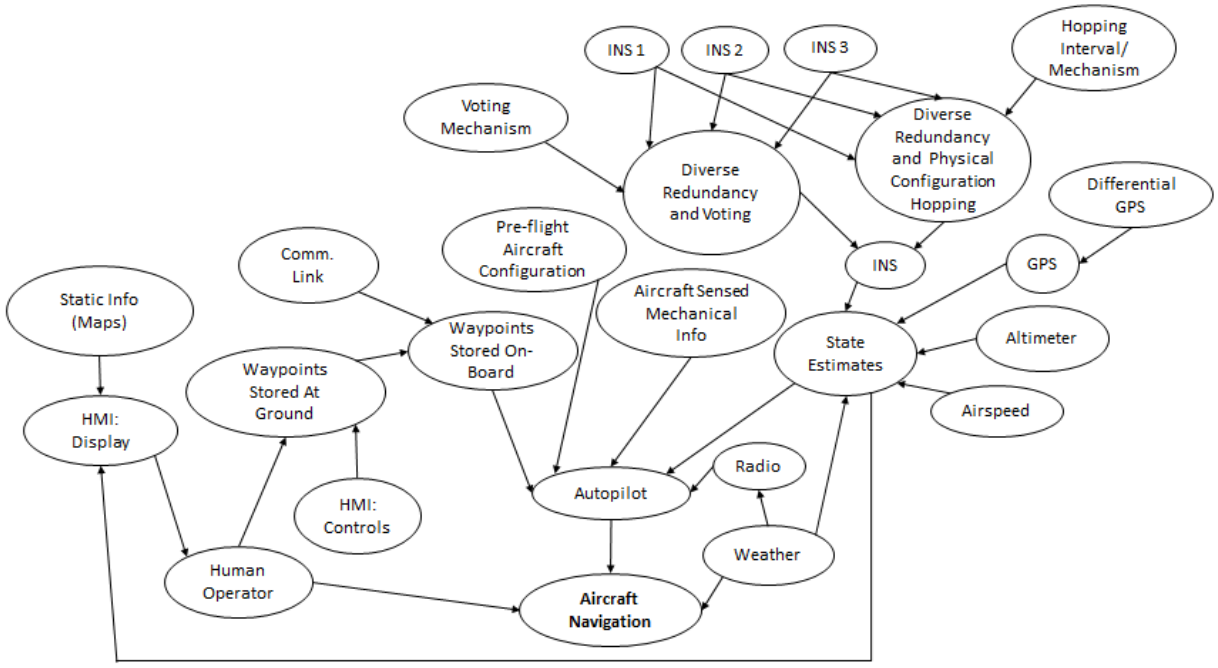

Figure 23. Updated System Influence Relational Diagram with Diverse Redundancy, Physical Configuration Hopping, and Verifiable Voting Implemented on the INS

Table 12. Impact to Adversary Relative to Impact to Defense for Diverse Redundancy, Physical Configuration Hopping, and Verifiable Voting

Diverse Redundancy and Physical Configuration Hopping		**Diverse Redundancy and Verifiable Voting**	
Impact to Defense	**Impact to Adversary**	**Impact to Defense**	**Impact to Adversary**
Implementation Cost: Medium	Design Knowledge: Med-High	Implementation Cost: Low-Medium	Design Knowledge: Med-High
Lifecycle Cost: Medium	Attack-Specific Technical Ability: Med-High	Lifecycle Cost: Medium	Attack-Specific Technical Ability: Med-High
Collateral System Impact: Medium	Resources: Medium	Collateral System Impact: Low	Resources: Medium
	Insider Presence (Operational): Low-Med		Insider Presence (Operational): Low-Med
	Insider Presence (Supply Chain): Med-High		Insider Presence (Supply Chain): Med-High
	Manpower/Time: Medium X2		Manpower/Time: Medium

In this case there is very little value of implementing both physical configuration hopping and verifiable voting. Since the adversary is more likely to alter the multiple diverse INS devices than to exploit their knowledge of the hopping mechanism or manipulate the voting mechanism, the additional action(s) they will be forced to take to bypass the protection will be basically the same. In the case with three diverse INS components, the adversary has to alter the measurements of all three to guarantee that the manipulated data will be used regardless of which INS is active with the hopping interval. Additionally, to avoid the voting mechanism detecting and identifying the altered INS measurements, the adversary needs to corrupt at least two of the INS devices – which they would have already done if they were working around the physical configuration hopping protection as well. In this case, implementing diverse redundancy and physical configuration hopping increases the difficulty to the attacker at a greater extent (specifically, requiring twice the manpower/time demand) than diverse redundancy and verifiable voting. However, verifiable voting has a lower impact to the defense (a lower implementation cost and set of collateral system impacts) than physical configuration hopping. The team can discuss whether the doubling of the manpower/time required by the adversary (to alter an additional INS device) is worth the increase in cost to the defense.

To continue the UAV navigation system example, the Medium implementation cost associated with data consistency checking implemented on the waypoints was determined to be acceptable and well worth the increase from Low-Med (for parameter assurance) in exchange for the lower defensive impacts down the line and high impacts to the adversary across the board. Additionally, it was decided that the doubled manpower/time demand could be a factor in preventing the Nation State's completion of the attack and it was worth the slightly high implementation cost to the defense (Medium as compared to Low-Medium). These two strategies – data consistency checking on the ground waypoints and diverse redundancy and physical configuration hopping– were combined to create a third possible architectural solution. Figure 24 shows the updated system influence relational diagram for this solution, and Table 13 shows the impacts associated with these two strategies (both to the adversary and to the defense).

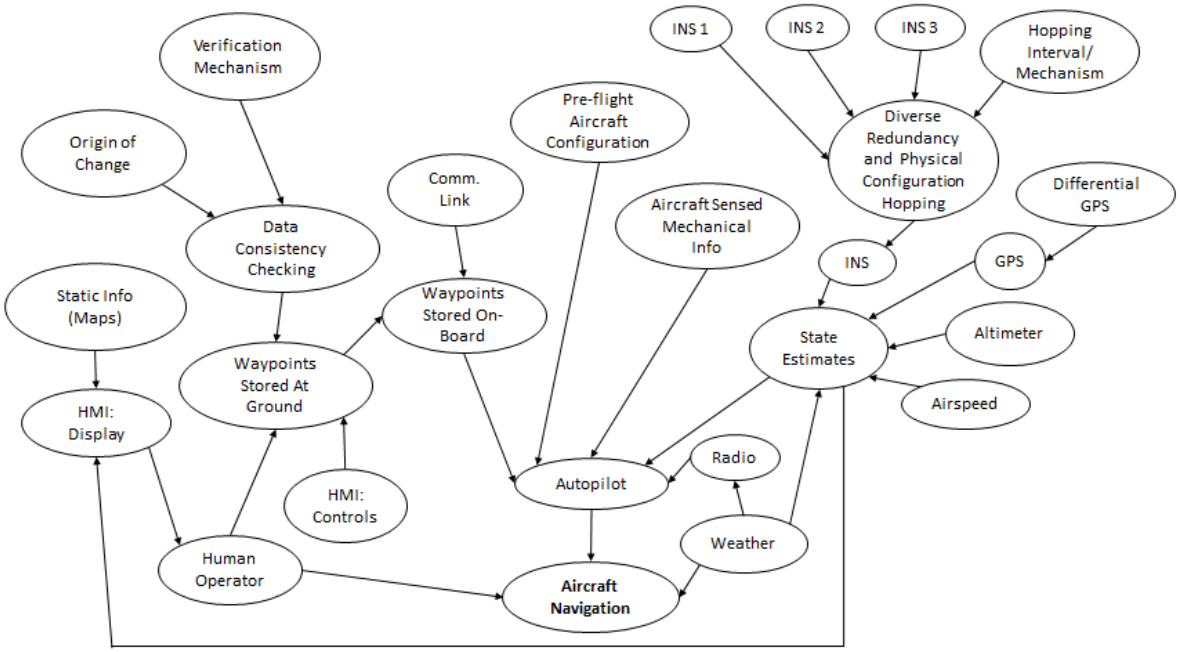

Figure 24. Updated System Influence Relational Diagram with Data Consistency Checking Implemented on the Ground Station Waypoint File and Diverse Redundancy and Physical Configuration Hopping Implemented on the INS

Table 13. Impact to Adversary Relative to Impact to Defense for Data Consistency Checking and Diverse Redundancy and Physical Configuration Hopping

Data Consistency Checking on the Ground Station Waypoint File		**Diverse Redundancy and Physical Configuration Hopping on the INS Devices**	
Impact to Defense	**Impact to Adversary**	**Impact to Defense**	**Impact to Adversary**
Implementation Cost: Medium	Design Knowledge: Medium	Implementation Cost: Medium	Design Knowledge: Med-High
Lifecycle Cost: Low-Med	Attack-Specific Technical Ability: Med-High	Lifecycle Cost: Medium	Attack-Specific Technical Ability: Med-High
Collateral System Impact: Low	Resources: Medium	Collateral System Impact: Low	Resources: Medium
	Insider Presence (Operational): Medium		Insider Presence (Operational): Low-Med
	Insider Presence (Supply Chain): Med-High		Insider Presence (Supply Chain): Med-High
	Manpower/Time: Medium		Manpower/Time: Medium X2

Even though the two strategies here do not complement each other in terms of multidimensionality protection, there is still value in implementing them in combination because they protect against two different types of attacks on the system. An adversary will never alter the waypoints at the ground station and alter the INS measurements both. Since they only need to do one to successfully alter the flight trajectory, the idea of two solutions filling in the other's gaps regarding the required levels of each behavioral indicators in less important here. If the team feels good about this pair of solutions and the asymmetric changes they force (med-high impacts on the adversary in a couple areas for low, low-med, and medium impacts on the defense), they can then return to the initial filters of budget and coverage to determine their next steps.

At this point, the team may decide that they would like to consider what additional coverage, specific to the "Concealed Major Trajectory Change" attack, they could gain by increasing their budget to implement an additional defensive strategy. Shifting focus to the attack actions that work to conceal a trajectory change, the team decides to consider the possibility of implementing data consistency checking on the HMI operator display to prevent a change to the display software. Data consistency checking is selected over state estimation for consideration of both budget and coverage reasons. Since the team is already at the high end of what they originally determined they wanted to spend, data consistency checking is a good option because the upfront costs to understand and develop the design pattern have already been spent on its installation on the ground station waypoint file, and the costs for a subsequent installation should be considerably lower. Data consistency checking on the HMI operator display software also makes sense from a coverage stand point since it can detect changes made to spoof the operator display, which the adversary may attempt regardless of which action they choose to take to alter the flight trajectory.

Figure 25 shows the updated system influence relational diagram with the addition of data consistency checking implemented on the operator display software.

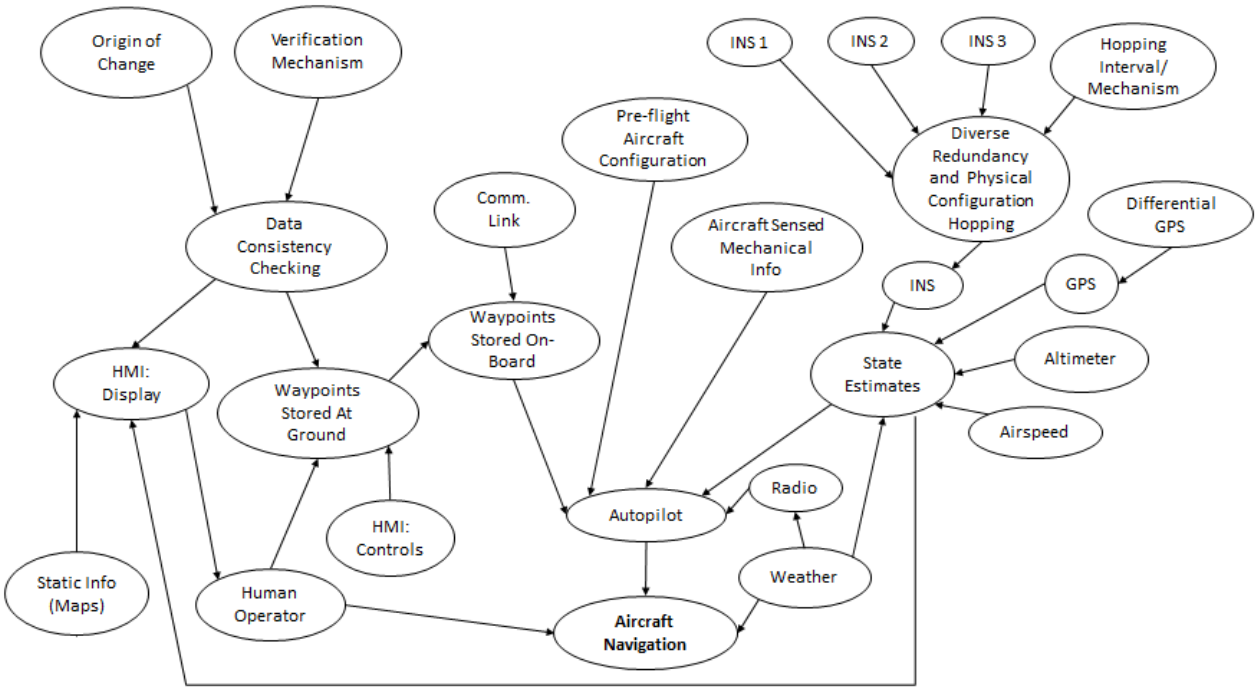

Figure 25. Updated System Influence Relational Diagram with Data Consistency Checking Implemented on the Ground Station Waypoint File and the HMI Operator Display Software, and Diverse Redundancy and Physical Configuration Hopping Implemented on the INS

In addition to graphically depicting the extent to which the defensive solution increases the difficulty and uncertainty for the attacker, the system influence relational diagram also serves to show how the design patterns fit into the context of the overall system. One of the limitations of other similar methodologies in existence is the fact that they consider design patterns individually as opposed to in combination, which ignores possible (dis)economies of scale. The DAG notation allows for multiple design patterns to be shown on the same diagram, which allows the defense team to intuitively see how they may interact to influence the adversary's actions. For instance, Figure 25 shows Data Consistency Checking implemented at two points in the system to protect against two different attack types: a ground station based waypoint change and a change of the HMI operator display software.

While this may have some beneficial implications for the defense (i.e.: the cost to add data consistency checking to a subsequent point in the system will be considerably less than the initial costs involved to develop the design pattern and implement it in the first instance), it may also have some

beneficial implications in the attacker's eyes that the defense team needs to be aware of. In Figure 25, the nodes "Origin of Change" and "Verification Mechanism" are common between the two separate Markov blankets. This means that an attacker would only be required to do a single additional action: in this case, alter the verification mechanism so that it would return that the changes did come from a trusted source, even though they did not in actuality. Doing this one action bypasses the protection implemented by the defense and makes both of the attacker's changes go through without being prevented. In the case of the Concealed Major Trajectory Deviation attack, the successful completion of these two changes (one to alter the waypoints for the UAV platform and the second to conceal the change to the operator) is enough to lead to a successful exploit.

One final point that needs to be discussed at this point is the issue of trust assuredness. System-Aware Cyber Security is based upon the assumption that the developed design patterns are more secure than the system they are intended to protect. If this holds, the second implementation of data consistency checking greatly reverses the asymmetry: for a minor additional implementation cost, the defense is able to provide protection on the second required action of the "Concealed Major Trajectory Change" attack and have an impact on the attacker that is much greater than the impact they incur. However, if the protection itself cannot be fully trusted, then implementing data consistency checking essentially compromises the protection by creating a single point of failure that if the adversary is able to corrupt the verification mechanism, they are able to bypass the protection with the completion of a single action. For this reason, it is critical that the teams discuss the level of trust assurance in each design pattern being implemented to guarantee that the protection strategy is considered to be significantly more secure than the system itself.

g. General Comments on the Workshop

The first point to note about the full scale UAV navigation system attack case study was the level of time involved. The workshop in its entirety was fairly time intensive and took longer than originally anticipated. Part of this was certainly due to the fact that this example was this first time that the framework had been applied to a full-scale, real-world application. Throughout the process, participants

were focused on providing information that would both advance the UAV project and allow the framework to be revised and refined as necessary.

For future applications, the time commitment for applying the relational methodology would be expected to be considerably less than that experienced for this case study. However, it should not be expected to be a quick process by any means. Future iterations of the method will still involve the same number of steps and level of analyses; there will simply be less discussion on the framework itself included in the process. Additionally, if the method is applied to future projects, the analyses required by the method would not have to be completed in such a short period of time. For the UAV navigation system application, the methodology was being applied and evaluated for the purpose of being included in this research thesis, which required that the entire process be completed over a two month time period. In future applications, a project timeline for the development of a system with System-Aware Cyber Security protection services implemented would be considerably longer and could even span multiple years. This would allow the framework to be utilized on a less condensed time frame and one that advanced more at the speed of the project itself.

Additionally, another option to reduce the time commitment for any particular participant is to revisit the team structure. While the framework was developed to use three distinct teams, the majority of the UAV case study drew upon the knowledge of the entire project team as opposed to subdividing the group into distinct blue, red, and green teams. This was done to ensure a large pool of knowledge and increase the amount of input and feedback received about the methodology. To decrease the time required from individuals across the board, smaller groups were utilized later in the methodology (steps 3, 4, and 5 were conducted with individuals or small groups with more specialized knowledge). While this still did not align with the originally described three team structure (there was still some overlap between teams – e.g.: an individual could serve on both the red and green teams but be excused from tasks requiring the blue team), it was closer and reduced the burden on participants that didn't have the appropriate knowledge for a particular step.

Even with the time commitment required, the relational methodology proved to be very useful for the UAV navigation system attack application. The System-Aware Cyber Security architecture solution process is a very complex decision process – with a large number of stakeholders and design alternative and the tremendous potential for uncertainty. Without some form of a decision support tool, the decision makers have no guidance and would typically select solutions that they have the most experience with, regardless of whether they are the best alternatives. The purpose of the framework was to provide a structure to force the team to consider all of the possibilities and ensure that nothing was overlooked. In a situation where there are so many possible choices, like the System-Aware Cyber Security architecture selection process, it's critical that the decision maker feels confident in their final decision and this confidence can only come from feeling that they have sufficiently explored the design space. By collecting information on the structure of the system to be protected, potential threat actor profiles, and the existing design pattern portfolio, and using that to guide the development of the potential design space, the relational methodology relies on using information elicited from the team to generate a set of possibly valuable alternatives and then constructively filter it down. Even in the condensed structure required for the UAV navigation system case study workshop, it was recognized that the structure presented by the framework enabled the participants to move through the decision space in a way that made them feel comfortable with the fact that they had adequately explored the decision space by the time they reached the sixth and final step.

Part of the motivation for the development of the relational methodology was a desire to avoid the quantitative nature of scoring which results in the decision maker more or less blindly receiving the final recommendation. Instead, the relational methodology aimed to keep the decision process as open and visual as possible. This was also successfully demonstrated through the case study; the structure of the framework allowed the decision makers to see the implications of their decisions and how the design pattern portfolio evolves throughout each step of the filtering process.

VI. Conclusions

Due to several critical issues that made the System-Aware Cyber Security architecture selection process different from a typical MCDA problem, there was a recognized need for a decision support tool set that could help stakeholders navigate the complex decision space and design an appropriate solution. Unlike the original weighted scoring method and other similar methodologies such as ASF, MORDA, and NRAT, the end goal of the relational methodology described here is not necessarily to reach a final recommendation. Rather, the framework aims to provide a structure to collect insight and information that otherwise could be overlooked and filter the set of possible solutions down to only those that increase the difficulty for the attacker while remaining at an acceptable impact to the defense.

The overall goal here is to reverse the asymmetry present in the problem by identifying areas where the defensive team can make minimal changes to the system that will cause a maximum increase in difficulty, uncertainty, or expense for the attacker. By utilizing existing graphical modeling techniques of as well as concepts from game theory to drive the order of the steps, the methodology aims to recognize that the system's outcome is dependent on the actions of both the adversary and the defense. This has allowed the methodology to leverage existing notation to create a structure that individuals with different knowledge and experiences can understand and contribute to as the architecture selection decision process evolves. Additionally, the methodology draws upon one very important, but previously ignored point: while it's obvious that the attacker's actions can influence the defense's system, the defense's choices also have a large impact on the adversary's success. By first identifying the possible actions an adversary could take to exploit the target system and filtering those down to a subset deemed most preferable based on the adversary's capabilities and preferences, the defense team can create a clearer image of which of their existing design patterns may be most beneficial. Then, considering the impact this subset of design patterns would have on both the attacker (the extent to which its implementation increases the difficulty associated with still executing the attack) and the defense (the associated monetary costs and additional collateral system impacts) allows the team to collectively filter this set down even further and discuss the security trade-offs of different solutions.

References

[1] Valarie Benton and Theodor J. Stewart. *Multiple Criteria Decision Analysis: An Integrated Approach.* Boston: Kluwer Academic, 2002.

[2] Kneale T. Marshall and Robert M. Oliver. *Decision Making and Forecasting: With Emphasis on Model Building and Policy Analysis*. New York: McGraw-Hill, 1995.

[3] Ralph L. Kenney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* New York: Wiley, 1976.

[4] Kenneth Chelst and Yavuz Canbolat. "Framing Decisions with Influence Diagrams," in *Value Added Decision Making for Managers*. London: Chapman & Hall, 2010, pp. 1-31.

[5] Rick Jones and Barry Horowitz. "System-Aware Cyber Security Architecture." *Systems Engineering*, vol. 15, no. 2, pp. 224-240, 2012.

[6] Rick Jones. "System-Aware Cyber Security." Ph.D. Dissertation, University of Virginia. 2012.

[7] Cristian-Aurelian Popsecu and Christina P. Simion. "A Method for Defining Critical Infrastructures." *8th World Energy System Conference.* vol. 42, no.1, pp. 32-24, 2012.

[8] Alissa L. McDowell. "Selecting a Pharmacy Layout Design Using a Weighted Scoring System." *American Journal of Health-Science Pharmacy.* vol. 69, no 9, pp. 796-804, 2012.

[9] Terrance R. Ingoldsby. "Attack Tree-based Threat Risk Analysis." *Amenaza Technologies Limited*. 2010.

[10] John L. Casti. "The Ways of Modelmaking: Natural Systems and Formal Mathematical Representations." *Alternate Realities: Mathematical Models of Nature and Man.* New York: Wiley, 1989, pp. 1-44.

[11] Ross D. Shachter. "Model Building with Belief Networks and Influence Diagrams." *Advances in Decision Analysis: From Foundations to Applications.* Ward Edwards, Ralph F. Miles, and D. Von Winterfeldt, Ed. Cambridge: Cambridge UP, 2007, pp. 177-201.

[12] David Corney. "Designing Food with Bayesian Belief Networks," in *Association for Configuration and Data Management*, Apr. 2000.

[13] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Francisco: Morgan Kaufmann, 1988.

[14] Yacov Y. Haimes. *Risk Modeling, Assessment, and Management.* New York: Wiley, 1998.

[15] Bruce Schneier. "Attack Trees." *Dr. Dobb's Journal*. [On-line]. vol. 24, no. 12, Dec. 1999. Available: http://www.schneier.com/paper-attacktrees-ddj-ft.html [Feb. 2013].

[16] Shelby Evans, James Wallner, David Heinbuch, John Piorkowski, and Elizabeh Kyle. "Risk-Based Systems Security Engineering: Stopping Attacks with Intention." *IEEE Security and Privacy*, pp. 59-62. Nov./Dec. 2004.

[17] David Rios Insua, Jesus Rios, and David Banks. "Adversarial Risk Analysis." *Journal of the American Statistical Association,* vol. 104, no. 486, pp. 841-854. Jun. 2009.

[18] Robert Powell. "Allocating Defensive Resources with Private Information about Vulnerability." *American Political Science Review,* vol. 101, no. 4, pp. 799-809, 2007.

[19] Edieal J. Pinker. "An Analysis of Short-Term Responses to Threats of Terrorism." *Management Science,* vol. 53, no. 6, pp. 865-880. Jun. 2007.

[20] Vicki Bier, Santiago Oliveros, and Larry Samuelson. "Choosing What to Protect: Strategic Defensive Allocation against an Unknown Attacker." *Journal of Public Economic Theory,* vol. 9, no. 4, pp. 563-87, 2007.

[21] Rick Jones, Barbara Luckett, Peter Beling, Barry Horowitz. "Architectural Scoring Framework for the Creation and Evaluation of System-Aware Cyber Security Solutions," unpublished.

[22] Don Buckshaw, Innovative Decisions, Inc. "Mission-Oriented Risk and Decision Analysis (MORDA)." MORS IA Special Session, Laurel MD. 7 March 2007. http://www.mors.org/UserFiles/file/meetings/07ti/buckshaw.pdf

[23] Bud Whiteman. "Network Risk Assessment Tool (NRAT)." *IAnewsletter*, vol. 11, no. 1, pp. 4-8, 2008.