



**A GAME THEORY MODEL FOR
ALLOCATING SCARCE RESOURCES IN
CRITICAL INFRASTRUCTURE
PROTECTION**

THESIS

Richard E Chappell II, Capt, USAF
AFIT-ENG-MS-17-M-012

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENG-MS-17-M-012

A GAME THEORY MODEL FOR ALLOCATING SCARCE RESOURCES IN
CRITICAL INFRASTRUCTURE PROTECTION

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Cyber Operations

Richard E Chappell II, B.S.

Capt, USAF

March 2017

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENG-MS-17-M-012

A GAME THEORY MODEL FOR ALLOCATING SCARCE RESOURCES IN
CRITICAL INFRASTRUCTURE PROTECTION

THESIS

Richard E Chappell II, B.S.
Capt, USAF

Committee Membership:

Dr. K. Hopkinson
Chair

Dr. D. Hodson
Member

Lt Col J. Pecarina, PhD
Member

Abstract

In this paper, we describe, implement and evaluate a process that tests the fault tolerance of a distributed system, and then applies behavioral game theory to evaluate specific resource allocation problems. Geographically dispersed power grid systems brings unique challenges in deciding where and how to protect assets. Model checking abstracts a system to a logical framework that can be tested for failures. Allocation strategies can then be derived from the comparison of individual elements. Level- k reasoning is then applied to discover iterative refinements to the allocation strategies, until a specific result is established. Case studies are presented that demonstrates how to apply this methodology to evaluate a system.

Acknowledgements

I thank all of my thesis committee, faculty and staff that helped me learn and progress through this program. Thanks to Seth Hamman, for your assistance and push forward. Thanks to all my co-students for working with me and teaching me. A special thanks to my wife and children, to whom I owe my love and appreciation for their sacrifice and support. They are the reason for my success.

Richard E Chappell II

Table of Contents

	Page
Abstract	iv
Acknowledgements	v
List of Figures	viii
List of Tables	x
List of Abbreviations	xi
I. Introduction	1
1.1 Overview	1
1.2 Background	2
1.3 Problem Statement	3
1.4 Research Goals	4
1.5 Contributions	5
1.6 Chapter Review	6
II. Literature Review	7
2.1 Introduction	7
2.2 Critical Infrastructure Systems	7
2.3 Model Checking	9
2.3.1 SPIN	10
2.4 Game Theory	11
2.4.1 Colonel Blotto	12
2.4.2 Level- k Reasoning	14
2.4.3 Resource Allocation	15
2.5 Summary	16
III. System Design and Architecture	17
3.1 Overview	17
3.2 Example WABPS System	17
3.3 Model Checker System	19
3.4 Game Theory Model	20
3.5 Summary	21
IV. Methodology	23
4.1 Overview	23
4.2 Testing Environment	23
4.2.1 Physical Hardware	23

	Page
4.2.2 CPLEX	23
4.2.3 Cygwin	24
4.2.4 SPIN	24
4.2.5 Visual Studio	24
4.3 Assumptions, Delimitations and Limitations	25
4.3.1 Assumptions	25
4.3.2 Delimitations	25
4.3.3 Limitations	26
4.4 Test Scenario	26
4.4.1 Model Checking	26
4.4.2 Game Theory	27
4.4.3 Defensive Strategies	28
4.4.4 Case Studies	30
4.5 Analysis Techniques	31
4.6 Summary	31
V. Results and Analysis	34
5.1 Overview	34
5.2 Vulnerability Test with SPIN	34
5.3 Game Theory Analysis	43
5.4 Case Studies in Defense	48
5.4.1 Case Study A	48
5.4.2 Case Study B	49
5.4.3 Case Study C	49
5.5 Summary	54
VI. Conclusion and Future Work	55
6.1 Overview	55
6.2 Conclusions of Research	55
6.3 Recommendations for Future Work	56
6.4 Final Thoughts	57
Appendices	58
Appendix A. Commands to Implement SPIN	60
Appendix B. Script for Reading Trail Files	61
Appendix C. Example Printout of Faulted States	62
Appendix D. Example of Attack Detection Chart	63
Bibliography	64

List of Figures

Figure		Page
1.	WABPS Schematic Map	18
2.	Excerpt of Hamman's code to model WABPS[22]	20
3.	Integer Linear Program by Seth Hamman[22]	22
4.	Failures with total IED fault	36
5.	Failures with IED sub-components fault	36
6.	Failures with complex IED failure	37
7.	Total failures per component	37
8.	Likelihood of nodes to cause a system failure as distance from the test link grows	42
9.	States explored versus the amount of memory necessary	44
10.	States explored versus the amount of time required	44
11.	Likelihood to detect an attack in case study A with each strategy	48
12.	Likelihood to detect an attack in case Study B with each strategy	49
13.	Likelihood to detect an attack in case study C with each strategy	50
14.	Comparison of defense strategies using higher rates of successful detection	51
15.	Comparison of defense strategies using lower rates of successful detection	52
16.	Likelihood to detect an attack in case study A with the alternate allocation strategy	52
17.	Likelihood to detect an attack in case Study B with the alternate allocation strategy	53

Figure		Page
18.	Likelihood to detect an attack in case study C with the alternate allocation strategy	53
19.	Likelihood of failure to detect an attack, Level-4 strategy, Case Study C	63

List of Tables

Table		Page
1.	Independent Variables	31
2.	Dependent Variables	32
3.	Control Variables	32
4.	Total Possible states and resulting errors	35
5.	Totals for Single Component Failures	38
6.	Totals for Total Component Failures	39
7.	Totals for Total Component Failures	40
8.	Grand Totals of all Failures	41
9.	Efficiency statistics	43
10.	Computed allocation strategies for Total IED failures, nodes selected from 3-16	45
11.	Computed allocation strategies for Total IED failures, nodes selected from 17-30	46
12.	Computed allocation strategies for Single IED Component failures	46
13.	Computed allocation strategies for all failures combined	46
14.	Alternate allocation strategies based on simple proportional allocation	47

List of Abbreviations

Abbreviation	Page
SCADA	Supervisory Control and Data Acquisition 1
PLC	Programmable Logic Controller 1
IT	Information Technology 1
OT	Operational Technology 1
BPS	Backup Protection Systems 3
IED	Intelligent Electronic Devices 3
ILP	Integer Linear Program 3
COTS	Commercial-Off-The-Shelf Systems 7
ACM	Association of Computing Machinery 10
WABPS	Wide Area Backup Protection System..... 10
APT	Advanced Persistent Threat 11
MTTC	Mean time-to-compromise 15
RDA	Regional Decision-Making Agents 17
LDA	Line Decision-Making Agents 17
IBM	International Business Machines 24
IDE	Integrated Development Environment 24
IDS	Intrusion Detection System 25
DDoS	Distributed Denial of Service 29

A GAME THEORY MODEL FOR ALLOCATING SCARCE RESOURCES IN CRITICAL INFRASTRUCTURE PROTECTION

I. Introduction

This first chapter provides an overview of the thesis subject matter and an introduction to the problem. It details the research goals and contributions. It ends with a review of the entire document.

1.1 Overview

Electric power, while a recent innovation to human society, is relatively old compared to modern technology lifespans. From the first electric station in 1882, the power utility grid has grown and spread throughout the US and the world[18]. These systems have been installed over years, often with differing levels of technology as advances occur in phases spread along an upgrade schedule. Supervisory Control and Data Acquisition (SCADA) systems were introduced as electronic devices such as the Programmable Logic Controller (PLC) allowed both automation and remote control over processes. Advances in Information Technology (IT) has led to an evolution in control systems, creating further advances in management capability. Formerly closed, proprietary systems are connecting to networks, allowing greater control, superior knowledge, and more efficient operations. This union of older mechanical systems and IT has been characterized as Operational Technology (OT). Unfortunately, these increased connections bring in the traditional weaknesses inherent to modern networking technology. A lot of OT is utilized in critical infrastructure systems that are spread over wide geographical distances. The power grid, colloquially known as

the *Smart Grid* when these increased technologies are added, is a prime example. The security of the overall system is a high priority[24]; however, resources available to protect the daily operation are necessarily finite, and must be distributed wisely. Developing such a strategy can require detailed knowledge of a systems vulnerabilities, attack patterns of adversaries, and defense strategies. What is proposed in this research is an algorithm for smart grid fault detection using model checking verification and an algorithm to apply game theoretic concepts to security resource allocations. This will be evaluated through a few case studies in which defensive strategies are applied in protection of a model system.

1.2 Background

Modern power grids are large, wide spread affairs, and require diligence in maintaining them. Maintenance is usually spread out over phases of time, and either relies on a schedule or in response to an outage. Natural and accidental causes, such as rodents on a line, are a large source of outages[17]. Of lesser known quantity is the degree to which cyber threats may impact the power distribution system. Deciding when and where to protect physical assets in a distributed system may be straightforward to the engineer who designs the distribution system, but adding in computer and networking technology in tandem to old electromechanical systems is a more difficult process. This is especially true when an engineer has little to no background in a rapidly evolving IT world. Questions that arise may include: where a computer-controlled system may fail, why will it fail, where in the system should be strengthened, and where can there be compromise between redundancy and cost? Increasing system interconnections may seem a type of answer. However, a fully connected mesh network would be extremely cost-prohibitive and physically improbable, while increasing connections unnecessarily can actually decrease the effectiveness[18].

In order to solve the problem of increasing the fault tolerance of a power grid, including against cyber attacks, there is a need to support existing hardware until it can be replaced with more secure devices. The power grid also needs to answer the problem of finite resources; a utility company must balance cyber defense costs with regular maintenance, personnel overhead and legal compliance, to name a few. A possible solution to this are Backup Protection Systems (BPS), which can rely on Intelligent Electronic Devices (IED)s. These IEDs have always functioned as an electronic protection that is able to monitor it's connected system for health and function. Modern versions, however, provide additional functionality and networking to create a distributed BPS. Apart from the system utilized to provide security is deciding on where to focus defensive resources. A large network may have unknown choke points which can threaten the overall functionality. Game theory, then, provides multiple potential frameworks in order to mathematically define potential human behavior. Previous work was done in finding the degree of fault tolerance in a BPS, as well as using game theory principles to formulate a plan to defend finite resources in a distribution network based on mega-wattage of the transformers. The important part of this work is the idea to unify the two models, while instead quantifying within a distributed system the specific nodes that will cause a breakdown in functionality. My hypothesis, then, is that after identifying the weak areas in the BPS, that the ILP developed by Hamman[22] will be able to identify a best case scenario for allocating finite defense resources. The research will also provide an example framework in which demonstrating the effectiveness of the model might be accomplished.

1.3 Problem Statement

Failure to adequately protect critical infrastructure can lead to disastrous consequences for the population that is dependent on its reliability. Knowing where

and how to allocate resources to protect against this possibility is difficult when the system owner is blind to the system’s vulnerabilities and how likely attackers might plan against them. Current research has focused on either vulnerability discovery or defensive strategies, without merging them into one cohesive process. The problem this research aims to solve is the following: *Can distributed system be modeled utilizing model checkers and find the fault tolerance for each node, then use the resulting findings to create a defensive proportional resource allocation strategy that can outperform an equal distribution strategy?* Success in the first half of that problem will be found if the research can successfully find a difference between the worth of each node towards the system’s fault tolerance. Success in the second half will be found if the resulting strategy can be found to have a statistical difference over an equal distribution strategy.

1.4 Research Goals

The goal of this research is to merge the functions of fault tolerance discovery with defensive strategy formation as applied to critical infrastructure systems. This will function by using a model checking framework to analyze specific weaknesses in a system, and then extending game theory to take the specific vulnerability information and automatically analyze a series of potential allocation strategies. Case studies will be provided for how these allocations might perform in an attack scenario.

The goals of this research are three-fold:

1. Develop an algorithm for individual node fault detection in a distributed system through the use of model checking technology.
2. Develop an algorithm to apply behavioral game theory reasoning using information on discovered vulnerabilities to formulate a defensive resource allocation strategy.

3. Demonstrate the end viability of the process by using the proposed allocation strategy in defense against a theoretical attack against the WABPS.

It is hypothesized that model checking can be used to discover specific levels of tolerance at the individual nodes in an abstract distributed system, leading to the discovery of the likelihood that individual nodes will cause a system fault. When this data is submitted through a strategy based on game theory, it should prove to have a positive effect in defending against an attack versus distributing resources equally. Potential difficulties in protecting a large system may arise from not understanding the value each sub-component has on the functionality as a whole. One important benefit of model checking should be that it provides consistent, provable results that demonstrate the fault tolerance of the system. This serves to remove guesswork involved in estimating the inherent risk of an information system.

1.5 Contributions

The results of this research will be used to develop a start to finish approach of analyzing vulnerabilities within a system, then inputting those results into a plan to allocate finite resources towards a quantifiable defensive strategy. The specific contributions of this research will be listed below.

1. An algorithm that can take a distributed system, and verify the fault tolerance for each node within the system
2. An algorithm that can take the fault tolerance for a system, and derive defensive allocation strategies that will outperform an equal distribution
3. A demonstration that these strategies will perform better than the equal distribution through statistical analysis.

1.6 Chapter Review

This chapter serves to introduce the subject area of the thesis. The following chapters will provide the details of the full research process of this thesis. Chapter 2 presents the literary findings that support the research and the structure of the model. It also details current research in the area. Chapter 3 details the example system to be evaluated, the SPIN model and the game theoretical model. Chapter 4 details the methodology used to evaluate how the proposed algorithms perform to accomplish the research goals. Chapter 5 reports the findings from following the experiment design presented in Chapter 3. Chapter 6 provides the conclusions and recommendations for future research.

II. Literature Review

2.1 Introduction

This chapter presents the introductory reading of relevant background information, and a look into the applicable research in the subject matter. It presents the previous research that has framed this paper. It will show how the focus of this document will serve to extend the previous inquiries, and bridge them together in a heretofore unexplored path. Most papers reviewed were written within the last several years, with a few older papers that describe long term concepts in game theory.

2.2 Critical Infrastructure Systems

Historical protection of critical systems has usually focused on physical solutions. These entail strategies like protection zones, electromechanical relays, and backup systems[28][18]. These solutions are usually designed to bring in security against unintentional failures within the system. This can include failing components, accidental damage, acts of nature and so forth. Solutions lately have included adding in newer, more advanced hardware that can communicate with central hubs or even work independently of engineers. One byproduct of this are the concept of a software-controlled backup protections systems[50][51][49]. Adding a large amount of new hardware across a large distributed system is costly, however, and ignores the long lifetime many OT systems have. However, with the rise in use of microprocessor-based products adding Commercial-Off-The-Shelf Systems (COTS) and IT, new strategies have to be developed to protect against intentional and malicious attacks more prevalent in these systems[47][29]. Adding additional links, that don't include the same physical poles can very unwieldy, as a mesh network of power cables going every which way is generally undesirable along with the exorbitant price tag. Solutions must take

into account existing lines without needing to add additional paths.

There are a range of methods developed to combat these new vulnerabilities. Some focus on bringing in traditional IT solutions. Firewalls and intrusion detection systems are one such approach[35]. These, however, introduce timing delays, interdependencies and additional potential weaknesses that can add new problems. Other research take the concepts of traditional IT approaches, and modify them for use with OT. These can include solutions like NexDefense’s Sophia anomaly detection tool[3], modification of honey-pot architecture to replicate network devices[52], and analyzing signal characteristics to identify correct devices[34]. These are novel ideas to advance ways to protect a system. The area that these do not touch is how to decide when and where to apply defensive or fault tolerant approaches, like technology, manpower or system redesigns. The research in this thesis hopes to focus on this aspect of the problem, and examine the areas of weakness within a system so that the owner is able to make an intelligent decision on where these defensive techniques may be best served. This should be able to account for systems that are already in existence as well as future projects.

Finding the correct path to the solution requires understanding and framing the question. Vulnerabilities and weaknesses in the system must be discovered as best as possible, and a solution should improve the system without adding too many additional issues. The backup protection systems as mentioned previously are a good way to use existing lines, however, the experiments that were run in the documents only ran through a limited number of case studies to demonstrate effectiveness. The question remains in how fault tolerant these additional protections actually are. There are a number of types of models that may work to solve this question, but there is a formal verification system that exists in computer science that can be bent to help answer this problem. This technology is known as model checking.

2.3 Model Checking

Evaluating all the potential states that a complex system can enter is a daunting prospect. Aside from performing limited case studies, previous research has explored a few methods of defining a model or verifying aspects of a model[48][5][16][9]. These types of models were rejected as requiring more human effort to complete, or did not fully investigate the functionality. Model checking is a formal verification method that was developed to comprehensively test all possible states. It is designed to allow a system to be implemented in an abstract program, and allows the programmer to examine against predefined specifications, also known as *system specifications*. These specifications should detail what the model must do, and what the model must not do. It can be applied early in a system's development, in order to discover underlying flaws in the architecture. It can be used to look for errors individually, or as a whole. It can also provide information about where the error exists and the situation leading up to it[11]. There are a few weaknesses to take into account. Primarily, it needs to be an accurate representation of the system; anything that is not modeled correctly will provide misleading results. Thus, it is imperative that the programmer have an adequate knowledge of how the system will be built, and the full requirements for complete functionality. Also, historically, state explosion has been a problem, due to the memory requirements for modeling complex systems. Newer hardware technology helps deal with this issue; modern virtualized servers can harness sufficient memory for larger models.

For system specifications, this research will rely on the work done by the BPS designer[49]. The correct operating conditions, fault conditions, system design, etc, are sufficient to build a functional abstract model. There are a variety of model checkers available. These include such offerings as TLA+[32][31], another leading competitor in the field. These applications were more designed to investigate com-

puter science problems, such as how a distributed algorithm may function. It has seen use in several papers to either test telecommunication systems[36], PLCs[54], or other cyber-physical systems[38]. More potential offerings are ACO[10], Alloy, Coq, F*, and Agda. While these other offerings have varying pluses and minuses, the model checker this research utilizes is SPIN.

2.3.1 SPIN.

One of the many model checkers available is the SPIN model checker. It was developed by Bell Labs, and is a open-source software package. It is an abbreviation for Simple Promela Interpreter. It was created by Gerald Holzmann in the 1980's, and has won the System Software Award from the Association of Computing Machinery (ACM)[25]. Models are developed in a custom language developed for it called Promela (Process MetaLanguage), which is C-based. It has seen various uses, from replicating distributed software systems[33] and retail banking systems[44]. It's seen use in modeling flood control barriers[30], telecommunications[27], and spacecraft control software[42][23]. All of these applications show that it can be successfully extended to much more than simple computer science algorithms. Critical infrastructure research has seen it used to verify non-interference in a power grid[46]. More associated with this research, it was used previously to model a Wide Area Backup Protection System (WABPS) by Hamman[22]. All of this shows SPIN as a mature product, based on a programming language widely adopted and well understood. This offers a shorter learning curve, and confidence in its capabilities. It has already demonstrated its potential to find fault tolerance on a system-wide scale, and it should be able to be extended down to the individual node scale as well. Using SPIN also provides the ability to validate further the research performed by Hamman on the same system.

2.4 Game Theory

Game theory is the study of complex human behavior in competitive environments. It allows researchers to simplify these behaviors in models between abstract players. These models serve as tools to allow the user to predict human behavior with repeatable mathematical formulas. Different models are used to demonstrate various aspects of human decision making. These models are designed to answer specific scenarios, so once the problem is understood, a relevant model may be chosen and applied[37]. Game theory as a discipline extends far beyond playing board games, however. It can also be used to model human behavior for security reasons. This can include shipping port security[45], general computer security[4], wireless network security[6] and masking locations of cyberspace shields[40]. The wide-ranging applicability of game theory allows it to be tailored individually to all of these domains, and the realm of cyber offense and defense is the important aspect for this research. Attackers could run a wide range of people and motives. This could run from the everyday script kiddie, who can only use already built tools on easy to reach targets, to the Advanced Persistent Threat (APT)s, that are able to discover new vulnerabilities, create their own tools and hide their tracks more efficiently. The fear is always on the APT as an aggressor, but more common is the lower skilled hacker. Thus, any defensive plan that cannot be tailored at the moment to a specific attacker must be able to handle any scale of attacker. The game theory used must also take an existing system that is difficult to change, so it is best to have a single decision point that works without later fine-tuning.

Game theory includes a variety of game models that can apply for different problems. For the scenario this thesis is exploring, it is non-extensive, meaning that there will only be one decision point. An extensive model would suppose intervals to change strategies in response to an aggressor, however, defensive strategies are typically de-

veloped in advance and set in place for extended periods of time. Thus, this research assumes that a defender sets up their defensive strategy and then employs it until broken or the system undergoes a substantial change. This also allows the concept of defensive strategies to include techniques that are either immobile or difficult to adjust. The model must also not allow perfect knowledge. The attacker may have gained some inside information, but generally the assumption is that the details of the defensive framework are at least partially hidden, so the attacker must guess at the defender’s strategy. It should also keep itself confined to two specific players, the attacker and defender. Usually a private or public utility would be performing its own defense, and not collaborating with always present outside assistance. While the defensive strategy may be employed against any number of attackers in individual engagements, the underlying defensive strategy would be constant. Lastly, the model should have a zero-sum, which would assist in simplifying strategies to a percentage of effort. The specific model this research will use to illustrate the case is a variant of the Colonel Blotto game, and the level- k reasoning method will be used to develop strategies. The previous investigations on the use of these techniques to solve different problems will now be explored.

2.4.1 Colonel Blotto.

Colonel Blotto is one of the common models found within game theory. Colonel Blotto can be codified as a two-person game, based on simultaneous discrete actions, a single finite instance, and a zero-sum ending. Colonel Blotto can be described as a finite number of battlefields and soldiers, where two players allocate their soldiers across the battlefields in a predetermined distribution. Each battlefield is won based on which player had the numerical advantage, and whichever player wins the highest number of battlefields wins[20]. The two players can be represented as the owner of

the power grid, and the opponent as a malicious agent. This extends itself well to a power grid system; the battlefields can be a substation, distribution bus or generator. The simultaneous discrete action is justified in that the goal of this work is to show a strategy that waits and responds to the initial move of the attacker. The zero-sum aspect demonstrates that in this model, the defender and attacker are striving for control of finite nodes, and thus must either win or lose in each position. For the defender, any presence of the malicious actor on a system is unacceptable. For the attacker, any lack of access to affect change means they have failed in their goal. Thus, this meets the previously described goals of a model with two players, zero-sum, a single decision point, and imperfect knowledge.

Colonel Blotto has been referenced in a few research papers for cyber security. Using a modified semi-Markov process has been proposed to model attacks, then focus on the potential drop of services after a successful attack [55]. This approach, however, used a sequential approach that is against the requirements imposed in this thesis. A three-way Colonel Blotto game with two sympathetic defenders is another modification to the original game[21], as well as the use of Colonel Blotto for developing physical defense strategies[39]. The prior of those two does not fully approach the problem this paper presents, as it introduces another player. The latter assumes that the attacker has full knowledge of the defense's strategy, whereas this research is looking for when the attacker at a maximum knows the vulnerabilities of a system. The major paper this research will build on is where Hamman used a variant of Colonel Blotto to define resource allocation for a finite number of heterogeneous substations[22]. None of these papers take into account the underlying weakness of a system, but focus on the relative worth of a node. This approach may help a defender focus on the valuable parts of a network, but ignore the possibility of a weak link that will allow the entire system to fall. Overall, the Colonel Blotto model utilized by

Hamman helps frame the question the best, and allows the experiment to be further validated by extending the types of input that it can handle.

2.4.2 Level- k Reasoning.

Level- k reasoning is a strategy defined in behavioral game theory, where opponents craft a strategy based on what their assumptions on what the other side will do[12]. Level zero is usually seen as the initial instinct, similar to equally distributing soldiers across several battlefields. Level one might be anticipating this strategy, and distributing a concentration of soldiers across specific battlefields in order to win a strategic number of victories. These layers can potentially be repeated ad infinitum. Pop culture examples of this might be found in *The Princess Bride*, where the character Vizzini extrapolates the potential back and forth reasoning of his adversary to decide where the poison is contained[19]. Another example might be found in Sir Arthur Conan Doyle's *The Final Problem*, where the character Sherlock Holmes and his adversary, Professor Moriarty attempt to respond to each other's probable actions in an escalating game of cat and mouse[14]. While these fanciful examples appear to take a lot of active strategic reasoning, if the calculations can be based in quantifiable comparisons, a mathematical model may be devised to create the levels.

There is little research available on using level- k within critical infrastructure or cyber security. Previous research used level- k to prove how players react in gambling money[7]. A key point of this paper is identifying the trend that deeper levels of level- k reasoning are not often reached by human decision-making. For critical infrastructure, work has been done, but a lot of the previous research centers only on initial levels, not exceeding level one in level- k reasoning in defense of a cyber-physical system[8][53]. This research focuses on the work by Hamman[22] in creating multiple levels of strategy. It further extends his work by applying it against gathered data

on fault tolerance instead of the amount of power generated. It then evaluates the expected performance versus an equal distribution strategy. The hope is that this research can continue to demonstrate the ease of setting up additional levels as needed, and explore the effectiveness of those additional levels.

2.4.3 Resource Allocation.

The goal with using the level- k reasoning is to define an optimal resource allocation strategy. Some previous topics on this include examining different allocation of physical security assets across cities[43], but this research does not fully realize the vulnerabilities or reasoning behind weight for each city. The most relevant recent research was set up to propose resource allocations for cyber defense of a SCADA system. The paper implied that a game theoretic approach would be more successful at protecting the system, but it failed to fully examine the weaknesses that contribute to the worth of each node. It also quantified an impact to decreasing or increasing resources for Mean time-to-compromise (MTTC). More in-depth experiments on the allocation strategies would be well served to enforce the ideas in that paper, but actual attack scenarios were not within the scope of this research, so it was not relevant to this paper. The research contained within this thesis should fill in the gaps left by this paper, by identifying why each node has a specific weight and an easy to use method to optimize resource allocation.

The experiments in this paper should provide a process to identify the value of a node and assign unique levels of protection that can take into account these values. The desire is for a system that can outperform an equal distribution, but not require a large amount of planning or forethought to develop. The ILP that was previously developed by Hamman offers a simple to use formula that helps support this solution to the problem.

2.5 Summary

This chapter provides a look at what previous research has covered in respect to the topics present in this thesis. Critical Infrastructure Protection has examined a large variety of potential avenues for defending its services. Model checking is an emerging technology that is beginning to show promise for solving questions about increasingly complex systems and their potential states. Game theory is the established field for mathematically quantifying human behavior. Combining these two disparate areas toward critical infrastructure protection has not been fully explored yet. This paper serves to unite these heretofore unlinked research paths into one unified framework. This will advance the gap between these two research paths. All of this information helps to clear the path of how this paper will succeed in accomplishing the goals delivered in the research objectives.

III. System Design and Architecture

3.1 Overview

This chapter serves to illustrate the framework that will be tested under this research. It will present the system that will be modeled, the software used to model it, and the game theory model used to create the resource allocation strategies. It will present the technologies at use in the research.

3.2 Example WABPS System

The system that will be examined is the WABPS proposed by Tong et al. This system is designed to detect various types of faults and attempt to correct the problem, preventing cascading failures. Various important parts of the system are the IEDs, Regional Decision-Making Agents (RDA), and the RDA sub-components, Line Decision-Making Agents (LDA). Figure 1 shows the example schematic that Tong developed in his research[49].

These decision-making agents are smart grid technology embedded as a software agent, to provide communication capability. It permits better information passing and faster coordination to changes within the system. The RDAs connect with each other in a decentralized peer-to-peer format, allowing for the network to function despite the failure of an individual node. The LDAs verify line health for the connected link, and communicate with other LDAs if there is insufficient information to declare a line fault. The LDAs will place a line into one of four faults depending on the state. Two are states that it can stay in, normal and fault, while the other two are transitional, special and suspect. Normal and fault states are as their names imply, while the suspect and special states are when the system notices a potential for a problem. These states instruct the IEDs to begin communicating with each

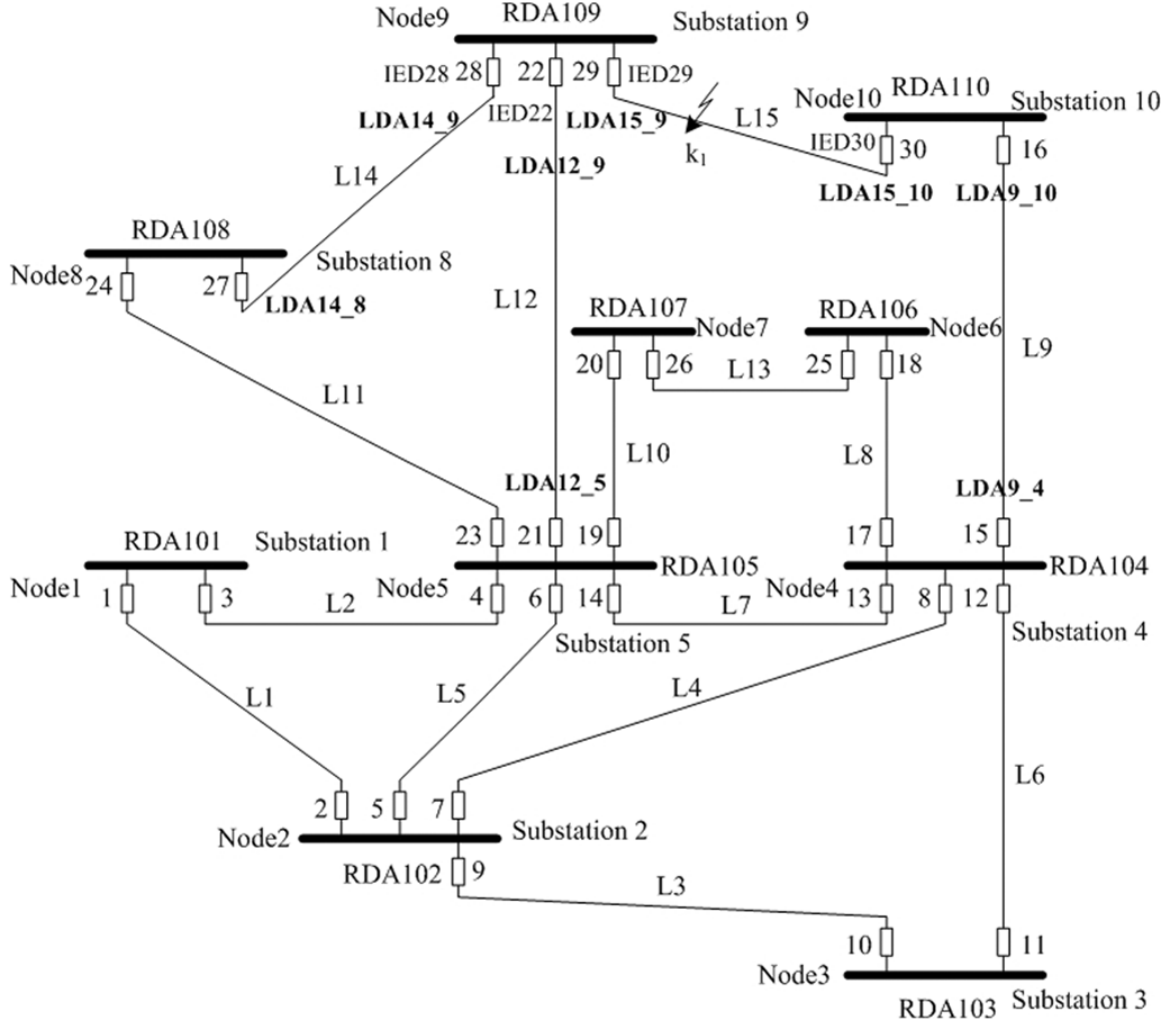


Figure 1. WABPS Schematic Map

other to first verify and then seek to correct the issue. Thus, if an LDA finds a fault condition, it can instigate flipping the circuit breaker to restore the line. Now, the decentralized peer-to-peer format of this WABPS is favorable to greater fault tolerance, as decision-making is not reliant on one hub, and the system can adjust to changes in the architecture. However, this also means that providing an in-depth model of all potential states is a lot more complex. The authors presented as proof of its fault tolerance four test cases in which a properly functioning system would discover the existence of a fault, locate it, and correct the problem. Hamman delved deeper in discovering how the system responded when components started breaking

down.

3.3 Model Checker System

The WABPS was previously recreated in a Promela model to work with the SPIN model checker by Seth Hamman. This previous model was used to verify the fault tolerance of the system as a total. The system was abstracted to fit within the parameters of SPIN, but retaining the intended functionality recorded in Tong et al's work [49]. Moreover, as the original system was a theoretical construct, the model was developed to only prove what was proposed as a part of the original specifications. To provide proof in SPIN, there are two parameters that must be checked: things that can go wrong, and things that must be right. The Hamman model provided a list of potential errors which may be condensed into two groups. One group of five errors, which was instituted as a weak correctness, depicted when an IED failed to report correct readings. The other group of two errors, strong correctness, depicted the occasions in which an IED either failed to communicate or failed altogether. The line that was modeled as the proof was whether the system would correctly detect a fault on Line 15, the original test proposed by the authors. The specifications built into the model by Hamman were two unique tests and a combination of the two unique tests at once. The first test is whether there is a single IED component failure, as depicted in the weak correctness test. The other is a total IED failure, as depicted in the strong correctness. The third test would be a combination of single component and total IED failure[22].

Within SPIN, the way that a model is typically tested, the model checker will process the states possible until it discovers a failure. For this research, specific parameters had to be set to allow SPIN to enter all possible states, not just to discover that one exists. SPIN will leave behind a trail file that holds the details

of the final state that caused an error. This will be overwritten by each new error, unless another specific parameter is entered to create individual trail files for each error discovered[25]. These two extra options in SPIN allow the model to be checked for all states as well as record all details that provide a failure. Figure 2 shows an excerpt of the SPIN model created by Hamman; in order show both the model and the original cases proposed by the author as well as his system fault tolerance tests.

```

192 init
193 {
194     // modify this to measure a particular fault tolerance degree
195     byte numSingleComponentFailuresToTest = 1;
196     byte numTotalIEDFailuresToTest = 1;
197     byte failuresSoFar = 0
198
199     initIEDStatesForL15Fault();
200
201     // these are a list of tests that may be executed
202     //testPapersFailureScenarios();
203     //test(corruptIEDComponent, numSingleComponentFailuresToTest);
204     test(turnOffIED, numTotalIEDFailuresToTest);
205
206     // this is atomic just so the verification mode will run faster
207     // arbitrary interleavings do not change the results, so I am not
208     // having SPIN perform them all
209     atomic {
210         run LDA15(0, 15);
211         run LDA12(1, 12);
212         run LDA14(2, 14);
213         run LDA09(3, 9);
214     }
215
216     // this blocks the init process until all the LDAs have completed
217     do
218         ::timeout -> break
219     od;
220
221     verifyL15Fault(); // weak correctness test
222     //verifyOnlyL15Fault(); // strong correctness test
223

```

Figure 2. Excerpt of Hamman’s code to model WABPS[22]

3.4 Game Theory Model

Creating a level- k strategy requires establishing a baseline that can be supposed as the most likely impulsive plan of action. While one option might be an equal

division of resources, this research will rely on the potential for an adversary who understands the weakness of a system as well. This is suggestive of an intelligent adversary who researches public information on a system and can model itself as well. Thus, the initial strategy will be a proportional allocation based on the vulnerability data gathered from the SPIN model. Modifying the terminology of the ILP that Seth Hamman developed to solve his version of Colonel Blotto, it can instead be established in equation 1 with N as total defensive resources, a power grid with J sites, and U as the utility of sites. The lower case of each letter would then represent the number of resources, each individual site, and the worth of each site respectively:

$$n_j = u_j / U * N, \forall j \in J \quad (1)$$

With this equation, it will take the total number of resources, which this research will state as 100 to create a simple percentage format, and distribute them proportionally across the sites based on the likelihood of being part of a system fault. The full ILP would be as follows in Figure 3.

This equation is essentially the same as developed by Seth Hamman, but instead of battlefields and soldiers, the language of the ILP is specified to mean IEDs and defensive resources, respectively. What this model does, then, is impose a limit of 100 resources to spread out among the totality of IEDs, with a predefined rule to only allocate in whole numbers. It takes the pre-existing strategy, and formulates a response that would succeed against the previously designed plan. This can be repeated for as many level strategies as is defined in a in the ILP shown in Figure 3.

3.5 Summary

This chapter serves to illustrate the framework under test, with both the model to be tested as well as the ILP that will be utilized. The next section will detail the

The ILP is as follows:

Let P be the set of possible attack strategies and K be the set of battlefields

Let a_{pj} be the number of soldiers placed at battlefield j in attack strategy p , for $1 \leq j \leq |K|, 1 \leq p \leq |P|$.

Let z_{jn} be a decision variable where

$$z_{jn} = \begin{cases} 1, & \text{defender places exactly } n \text{ soldiers at battlefield } j \\ 0, & \text{otherwise} \end{cases}$$

for $1 \leq j \leq |K|, 0 \leq n \leq 100$.

Let ε_{pjn} be an indicator variable that is calculated off-line for each $1 \leq p \leq |P|, 1 \leq j \leq |K|, 0 \leq n \leq 100$.

$$\text{For } n \neq 0, \varepsilon_{pjn} = \begin{cases} 1, & n > a_{pj} \\ 0, & n = a_{pj} \text{ , so a tie results in 0 points.} \\ -1, & n < a_{pj} \end{cases}$$

$$\text{For } n = 0, \varepsilon_{pjo} = \begin{cases} 1, & 0 = a_{pj} \\ -1, & 0 < a_{pj} \end{cases} \text{ , so a 0-0 score results in a win for the defender.}$$

Let u_j represent the utility of battlefield j for $1 \leq j \leq |K|$. A model to find an optimal strategy for the defender becomes:

$$\begin{aligned} \text{maximize} \quad & \sum_{p=1}^{|P|} \sum_{j=1}^{|K|} \sum_{n=0}^{100} u_j \varepsilon_{pjn} z_{jn} \\ \text{subject to} \quad & \sum_{j=1}^{|K|} \sum_{n=0}^{100} n \cdot z_{jn} = 100 \\ & \sum_{n=0}^{100} z_{jn} = 1 \quad \forall j \\ & z_{jn} \in \{0, 1\} \quad \forall j, n \end{aligned} \tag{2}$$

Figure 3. Integer Linear Program by Seth Hamman[22]

methodology in general and what will be measured as a result.

IV. Methodology

4.1 Overview

This chapter will discuss how the research will validate the theories put forth in this thesis. The first section will discuss the testing environment. The second section will discuss the assumptions, delimitations and limitations present in this research. The following section will discuss the testing scenarios used to evaluate the algorithms. Next will describe specific statistical analysis. It will then conclude with the summary of the methodology. It is important to note that this research is directed at validating the process of analyzing the system for weaknesses and then developing a resource allocation system. While exploration of how potential defenses would react to an actual attack, this research will not perform an experiment on actual defense techniques, but identify how it might respond to such a trial.

4.2 Testing Environment

The research in this thesis was accomplished primarily on programs that utilized the platforms of Microsoft Visual Studio, CPLEX, Cygwin and SPIN.

4.2.1 Physical Hardware.

This research was accomplished on a HP ZBook 15, running an Intel Core i7-4800MQ @2.7GHz, 32GB DDR3 memory running Windows 7 64-bit Professional as the operating system.

4.2.2 CPLEX.

CPLEX is a software package designed to solve integer programming problems, among other algorithms. This allows for an ILP to be designed and solved through a

program that can be simply maintained and extended. It is overseen by IBM and portable to a number of different languages. It includes an Integrated Development Environment (IDE), but can be extended to many other third party platforms[1].

4.2.3 Cygwin.

Cygwin is a software package that models a Unix environment on a Windows machine. It has the ability to import a large variety of packages. The necessary packages were installed in the 32-bit version to run SPIN[41].

4.2.4 SPIN.

SPIN is a software implementation of model checking, created by Bell Labs in the 1980s. It belongs to the class of tools known as formal methods. It works by testing all the possible states that an abstract system can enter into, and find whether any of these state transitions violate the checks that the developer creates[3]. The usefulness of this tool over standard simulations is that it exhaustively checks any combination. Even with a relatively small and straightforward model like Tong's, modeling all the possibilities of three simultaneous failures can reach tens of thousands. The Promela code that SPIN uses can be compiled into C code and run through any C environment. In the case of this research, it was performed in a Cygwin 32-bit environment.

4.2.5 Visual Studio.

Visual Studio is an IDE and compiler developed by Microsoft. It provides a lot of different benefits well known to the industry, but importantly to this research, works well with including the CPLEX libraries to develop applications in the C++ language[2]. CPLEX is the library utilized to program and run the ILP.

4.3 Assumptions, Delimitations and Limitations

This section will discuss the various assumptions, delimitations and limitations that go along with this research.

4.3.1 Assumptions.

One major assumption within this research is that the primary strategy would be a proportional allocation of resources across the system, versus an equal allocation. Equal allocation would probably be the first step of a system owner uneducated in the differences in the value of each node. This model, however, starts with a proportional allocation, assuming that the owner has completed the first step in analyzing the system for the most critical nodes. This assumption will be addressed by comparing the capability of the Level-0 strategy against an equal distribution strategy. Another assumption this research will state is the capability of defensive techniques - while the research places an arbitrary value as a placeholder, as long as whatever is used is consistent, the results should correlate the same, albeit in different percentages. In the case of a programmed defense system, such as an IDS, anti-virus, and firewalls, performance should be equal. Cyber operators can have differing capabilities, but an organization that hires a group would most likely deliver similar training, techniques and procedures so that all defenders at least have a baseline of performance.

4.3.2 Delimitations.

A delimitation to the extent of this research is to focus on one industrial control system, and more specifically the temporal logic contained within the specifications. The concept of model checking is not yet sufficient to run a fully realized computer program, but more specifically the logic entailed within. Another one is the reliance on general knowledge and assumptions on attack strategies, versus recruiting a selection

of human adversaries. Also, the choice was made to use a basic computer as opposed to a large server, limiting the number of states that could be reached within the model checker. Finally is how the defensive technologies are implemented; this research will focus primarily on overall strategic goals, not the everyday tactical methods of monitoring a network.

4.3.3 Limitations.

Limitations in this research is that the model cannot recreate the end state of hardware and software within the system. As the model checker is designed to test the temporal logic of the abstract model, it relies on the actual builder to recreate the logic faithfully without introducing additional errors. Another limitation is the performance of the model checking engine. While checking all the potential states, up to and including the entire system failing would give the most information; however, the state requirements of this would be unrealistic, and as more of the nodes fail, the system would be unlikely to function regardless of where the breaks were. A last limitation of this research is the failure of the Colonel Blotto game to handle multiple iterations of decision-making. While a defender would need some form of agile response to an attack, the scope of this research is to focus on pre-planning an allocation strategy, so this limitation should not affect the results of this research.

4.4 Test Scenario

The experiment will be conducted in three stages.

4.4.1 Model Checking.

The model checker will go through the entirety of states that the system can enter in, triggering a fault on the predetermined number of IEDs. The model is robust

enough to discover both strong and weak fails, and to choose between introducing a failed node, an individually failed link out of the three possible per node, or any combination of the two. The experiment will examine increasing levels of node failure, link failures, and the combination of all three. The respective nodes that contribute in a system failure will be recorded, as well as an analysis of the degree that the nodes are relevant to a system breakdown. The amount of times each node was complicit in a system failure will be compared against each other, to develop a set of percentages. This should illuminate the different values of each node as related to susceptibility to a system failure. The previous ILP performed by Hamman was based on ranking substations by wattage capability; this research will instead use the new classification.

Additional information will be recorded as the memory usage and time consumed to run all the tests. These will be examined against the states explored in order to illustrate how well the model checker performs in completing a state search of the system.

4.4.2 Game Theory.

After receiving the percentages that each node's fault tolerance, the ILP algorithm will be applied. This will initially provide the output of a proportional allocation, in whole numbers. The ILP will be run to develop multiple levels of allocation strategies. Increasing levels of strategy will be recorded, and the numbers will be used to generate the potential that each level has in defending against prospective attacks. These different levels will be the total of level- k strategies.

As a comparison against the proportional allocation that ILP will start with, an equal distribution of resources to each node will be proposed and compared. Additionally, because the ILP appears to distribute released resources from the previous level equally instead of proportionally, an additional comparison will be made be-

tween the ILP working normally, and redoing the allocation proportionally on each level.

4.4.3 Defensive Strategies.

Finally, this thesis will take the applied reasoning levels, and use them in a theoretical test case which will apply them against a classic attack scenario. This paper will evaluate the effectiveness of a potential resource to identify an on-going attack, and compare it against a basic equal and proportional distribution strategy.

With the previous results, we can provide case studies in how a company could then decide to defend its power grid. There are a few decisions that can be made in how to safeguard the system. These can entail entirely passive defenses, like log monitoring or an intrusion detection system (IDS), which alerts the owner after an attack or in the middle of an attack. These could be preferable when calculations must be made on how to limit an attacker as opposed to loss of services. The defenses could also be active, such as an active defender who looks for erroneous traffic or an intrusion prevention system (IPS) which automatically stops a perceived attack. Other options could include a combination of the two[13].

First, a utility company decides to implement this wide area backup protection system. They build and run a model checker which finds the likelihood that a downed node results in a failed system. After discovering the utility cost of the nodes for the link, the grid owner chooses the third level of allocation. The owner can now present a plan for dedicating 29% of labor to monitoring traffic on Node 29, 26% on Node 30, and etc. This level three strategy would ignore 13 nodes each only occurring only in 1% of failures to increase 19% more resources to the nodes most likely to cause a system failure. Alternatively, the system owner could utilize a portion of the resources to create an extra, redundant link that would be able to continue monitoring the line

when the primary goes down.

A theoretical attacker, then, would undergo reconnaissance on the system, and would probably guess that the most local IEDs next to the node are a likely target. He then performs some manner of attack. This could be blatant, with a DDoS, in which he uses a group of network connections to target the link, flooding it with bad traffic[15]. The surrounding IEDs would then be likely to identify the link as having faulted, and divert power to prevent a further disruption to the grid. The owner having instituted a backup line would have allowed the system to detect the failed node, but still be able to find the line functioning correctly. Alternatively, the attacker could attempt to be stealthy, and put in malware that would alter the data that it is sending out. Finding the malware on the system would rely on technicians having access and monitoring the node. In this case, the owner has a higher chance to find attackers malware when they are dedicating resources to monitoring these critical nodes.

For the purposes of this exercise, assume a passive defense, an intrusion detection system, is placed on each node. An equal probability will be assigned that each detection system will detect an attack, as the software would be homogeneous. For an automated attack, such as a DDoS, this research will assume a 33% of correctly diagnosing an attack. For an advanced threat, like a active attacker seeking entry, assume a 10% chance of success. These percentages reflect the greater ease of detecting common threats that automated malicious software like worms and DDoS platforms, versus the greater difficulty of detecting an intelligent actor. For each unit of defense, assume that it represents an active cyber defense, an operator monitoring activity on the network. This one unit will represent an additional 1% chance of correctly diagnosing an attack. This represents a total number of man-hours that an operator might be viewing logs, current network traffic and processes running on the nodes.

For all exercises, review Figure 1 for the system map.

4.4.4 Case Studies.

Case Study A.

The attacker performs a denial of service(power) against an entire IED. The attacker's goal is disrupting a facility on Line 15, so would plan on faulting at least one of the two adjacent IEDs on the system. The attacker plans on disrupting a component on IED 29. After a hour of no progress, the attacker would move on to disrupting a component on IED 22. After an hour without progress there, the attacker will end taking down a component on IED 30, which brings the system to a system failure for link 15.

Case Study B.

For this scenario, the attacker focuses on bringing down a series of individual components to induce a fault. The attacker starts from IED 21, inducing a fault, pivoting to IED 22 and setting up a fault, then finally pivoting to IED 29 to introduce a bus fault. This is shown by the model checker to break the system's functionality.

Case Study C.

As a last example, the attacker plans on conducting a mixed attack, by focusing on disrupting individual subcomponents and a whole IED. The attacker will cause a bus fault in IED 29, wait an hour, cause a line fault in IED 28, wait another hour, and then bring down IED 30. This is another combination which will cause the system to fail.

4.5 Analysis Techniques

Analysis techniques will involve a comparison of percentage chances of detecting attacks to differentiate between allocation strategies. Box plots will be constructed to illustrate the difference between the allocation strategies. It also includes comparing the estimated plot of the rise of memory usage and time versus the amount of states explored.

All of the independent, dependent and control variables that were isolated for this research are listed below in Tables 1, 2, and 3.

Table 1. Independent Variables

Independent Variable	Normal Operating Level and Range	Proposed Settings	Predicted Effects
# of IEDs that are faulted	0-30	0-5	As the number increases, the number of system faults should increase
# of IED sub-components that are faulted	0-120	0-4	As the number increases, the number of system faults should increase
Rate that an IDS can catch an attack	0-100%	10%, 33%	As the percentage increases, the odds of detecting an attack should increase
Number of IEDs attacked	0-30	0-3	As the number of nodes attacked are increased, the odds of a successful detection should increase
Number of Hours in attack	0-24	0-3	As the number of hours increase, the odds of a successful detection should increase
Levels of Defensive Strategies	0-10s	0-4	As the number of levels are increased, the allocations should gravitate to the more critical nodes
Initial Allocations	0-100	0-100	Either added equally across all nodes, or distributed proportionally

4.6 Summary

The research methodology is laid out in this paper, and a straightforward explanation is essential to show the evaluation of the hypothesis. The items under test are the principles of model checking for individual node fault tolerance and behavioral game theory for applying vulnerability discoveries to allocating resources in a defensive strategy. Exhaustive state checks will be performed for the model checking, and analysis will be carried out on the behavioral game results. This research serves to

Table 2. Dependent Variables

Dependent Variable	Normal Operating Level and Range	Measurement, Accuracy, and Precision	Relationship to Independent Variable
# of States Explored	24-1,000,000s	Whole numbers, based of number of faults imposed	As the number of faulted states are tested, this number should rise
# of times a faulted IED is part of an overall system fault	0-1,000s	Whole numbers, based on how many times a faulted IED helps cause a system fault	As the number of imposed faults increase, there should be more failures recorded
# of times a faulted IED subcomponent is part of an overall system fault	0-1,000s	Whole numbers, based on how many times a faulted IED helps cause a system fault	As the number of imposed faults increase, there should be more failures recorded
# of Time to Complete Test	0-100s	Measured in seconds to the thousandth place value	As the number of imposed faults increase, the time taken should increase
# of Memory Consumed	0-10,000s	Measured in Megabytes, to the thousandth place value	As the number of imposed faults increase, the amount of memory to store states should icrease
Defensive Allocations	0-100	Whole numbers, 100 units to be divided amongst all IEDs	As more levels are added, the numbers should gravitate to the more critical nodes
Likelihood of Successfully Detecting an Attack	0-100	Percentages, recorded to the hundredth place value	As time and more nodes are added, odds of success should increase

Table 3. Control Variables

Control Variables	Desired Experiment Level	How controlled?	Anticipated Effects
# of IEDs, Links	24 Nodes, 12 Links	Program Configuration	Changing the number of nodes and links would affect the system tolerance
Case Studies	3	Design	Maintaining a constant number of case studies for each strategy level keeps the measurements equal

demonstrate the viability of taking a system under proposal, checking for underlying errors, and developing a defensive strategy given finite resources. Finally, it will provide test case evaluation of the created strategies.

V. Results and Analysis

5.1 Overview

This chapter contributes an analysis of the results of applying the framework presented in chapter 3. It serves to demonstrate the feasibility of the full process. The functionality of the process is first demonstrated by applying a model check of the system, interpreting the results into a game theory-derived strategy, and then applying it to a hypothetical attack scenario. The applicability will be shown through gathering vulnerability data, and finding the end capability to defend a system. The results will be both observations of experiments and interpretive analysis.

Please note that this framework is developed off of a conceptual model, and that the end result is based on the accuracy of the model checking, as well as how the system itself is actually developed. This is due to model checking being more suited to higher-level abstract reasoning of how a system should function, and does not include unplanned flaws in actual software or hardware.

5.2 Vulnerability Test with SPIN

We ran tests in accordance with the guidelines established in chapter 3. Using the SPIN model checker, the model ran through each potential state, leaving behind a trail file for each discovered error state. The total number of states possible for each group is seen in table 4. In total, over 4.7 million states were investigated. Using a script, the heads of each trail file were combined into one document, and totals of each IED were added for each group of errors. Due to the consequences of state explosion inherent to model checking and the limitations of the machine used to run the program, the system used was able to test when the system had 0-4 total IED failure, 0-5 individual link errors, and having a combination between one and two of

both IED and a link error. Tables 5 through 8 show the detailed numbers for each IED. The total component failure refers to when an entire IED goes down, single component refers to when individual links go down, and complex is the combination of both.

Table 4. Total Possible states and resulting errors

Type of Failure	Number of Failures	Total Possible States	Number of Errors
Single IED Component	1	96	0
	2	4560	0
	3	142880	268
	4	3321960	4347
Total IED	1	24	0
	2	276	0
	3	2024	6
	4	10626	121
	5	42504	1151
Single, Total	1, 1	2208	0
	1, 2	24288	45
	2, 1	100464	247
	2, 2	1056528	5552

Figures 4, 5, and 6 show the graphs of the totals that were derived from running the experiments across all types of failures. While the total and sub-component IED failure types yielded slightly different percentages, overall the most critical nodes were self-evident.

After running through all the tests and gathering the data show in tables 5 through 7, the decision was made that the trends continued as the number of potential failures increased, so higher levels of failures were not sought. Too many introduced failures would likely cause an overall system fault regardless of where they were, so the limit of tests was accepted.

Figure 7 compiles the total failures of each failure type into one grand total of failures. While only a few IEDs appear to have a great effect in the total IED failure, single and complex failures show a bit of variation in how likely a IED is to cause a

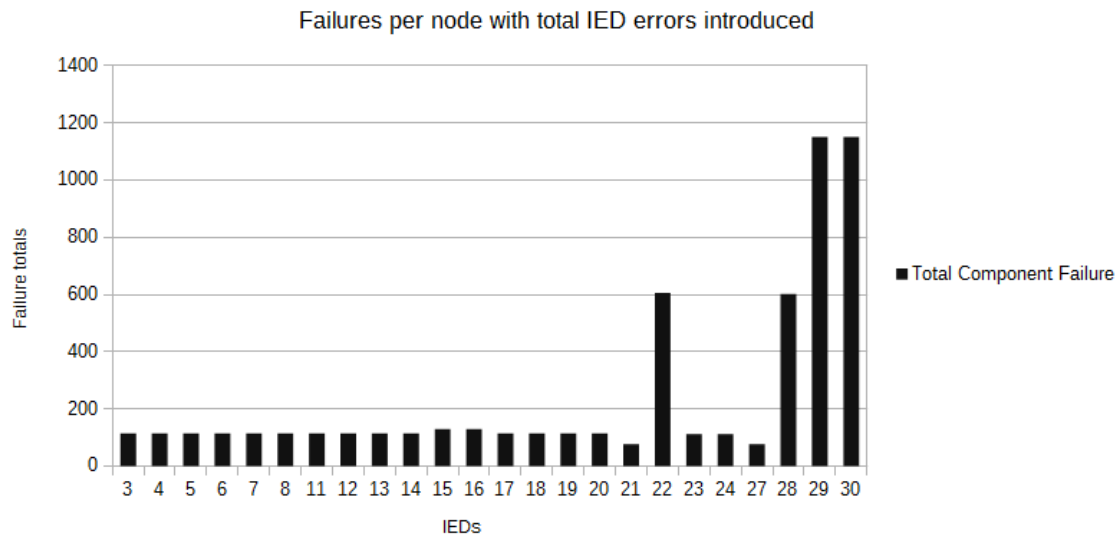


Figure 4. Failures with total IED fault

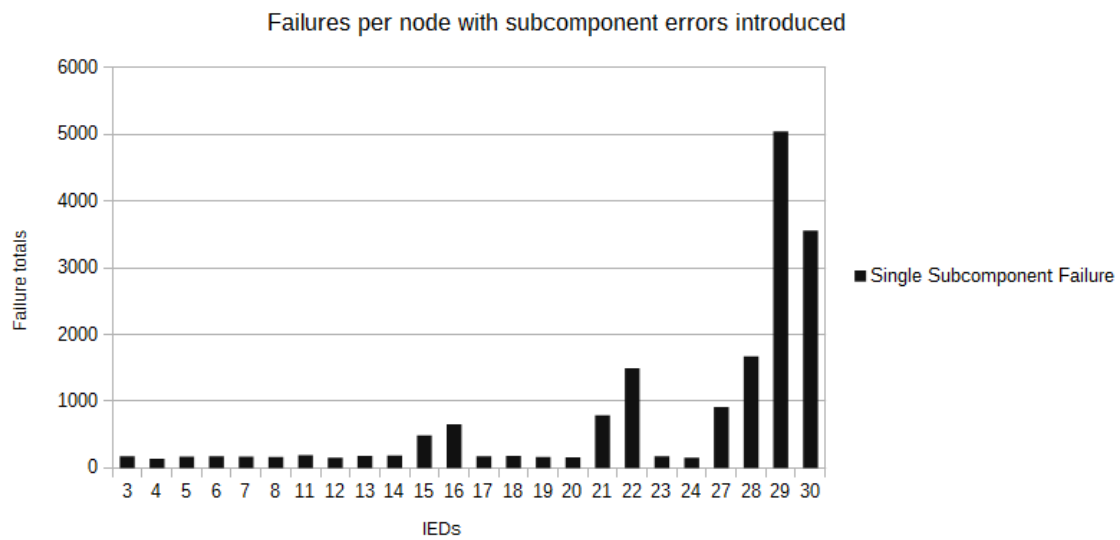


Figure 5. Failures with IED sub-components fault

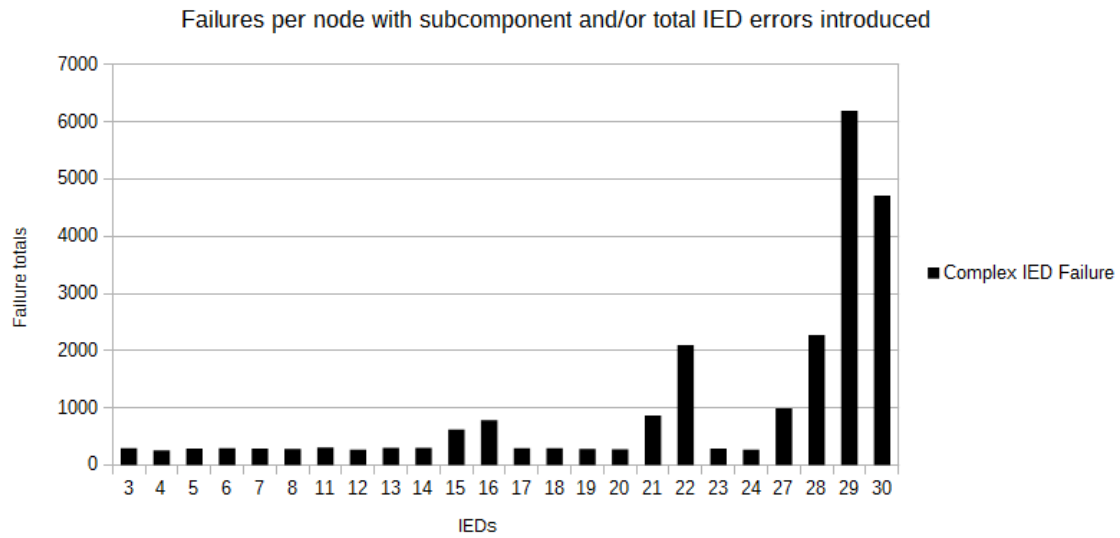


Figure 6. Failures with complex IED failure

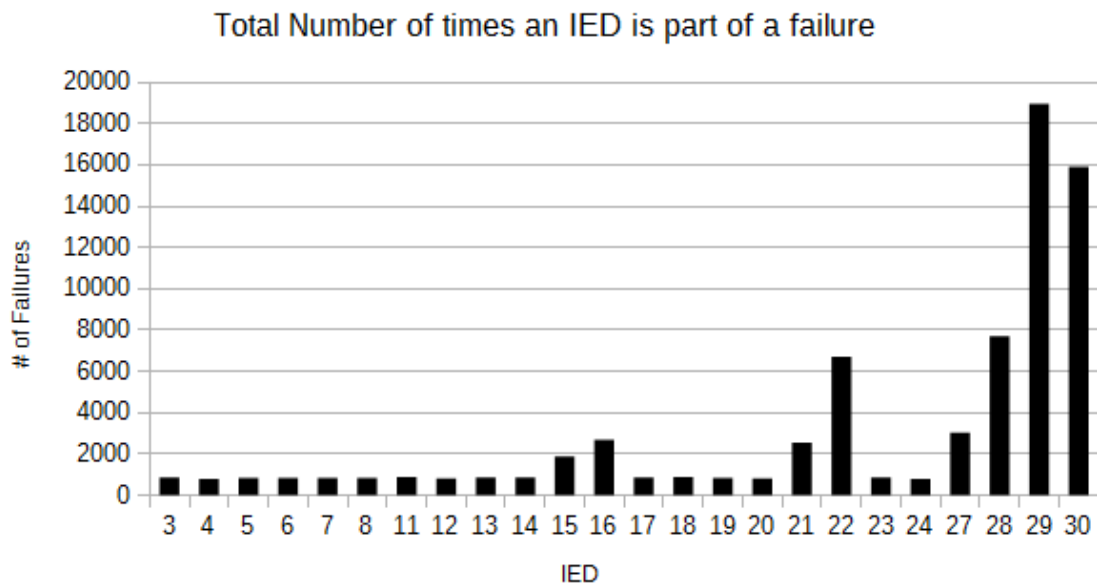


Figure 7. Total failures per component

Table 5. Totals for Single Component Failures

IED	1	2	3	4
3	0	0	0	179
4	0	0	0	141
5	0	0	0	171
6	0	0	0	178
7	0	0	0	173
8	0	0	0	168
11	0	0	0	192
12	0	0	0	155
13	0	0	0	185
14	0	0	0	188
15	0	0	44	490
16	0	0	66	657
17	0	0	0	180
18	0	0	0	182
19	0	0	0	169
20	0	0	0	163
21	0	0	53	791
22	0	0	86	1496
23	0	0	0	176
24	0	0	0	157
27	0	0	53	917
28	0	0	86	1673
29	0	0	205	5046
30	0	0	211	3561

Table 6. Totals for Total Component Failures

IED	1	2	3	4	5
3	0	0	0	6	115
4	0	0	0	6	115
5	0	0	0	6	115
6	0	0	0	6	115
7	0	0	0	6	115
8	0	0	0	6	115
11	0	0	0	6	115
12	0	0	0	6	115
13	0	0	0	6	115
14	0	0	0	6	115
15	0	0	0	7	130
16	0	0	0	7	130
17	0	0	0	6	115
18	0	0	0	6	115
19	0	0	0	6	115
20	0	0	0	6	115
21	0	0	0	4	76
22	0	0	3	62	605
23	0	0	0	6	112
24	0	0	0	6	112
27	0	0	0	4	76
28	0	0	3	62	602
29	0	0	6	121	1151
30	0	0	6	121	1151

Table 7. Totals for Total Component Failures

IED	1 Single, 1 Total	2 Single, 1 Total	1 Single, 2 Total	2 Single, 2 Total
3	0	0	0	264
4	0	0	0	262
5	0	0	0	251
6	0	0	0	234
7	0	0	0	251
8	0	0	0	269
11	0	0	0	268
12	0	0	0	258
13	0	0	0	251
14	0	0	0	235
15	0	26	2	477
16	0	49	3	855
17	0	0	0	265
18	0	0	0	270
19	0	0	0	256
20	0	0	0	236
21	0	38	5	614
22	0	86	18	1999
23	0	0	0	262
24	0	0	0	241
27	0	38	5	838
28	0	86	18	2650
29	0	209	42	5393
30	0	209	42	5309

Table 8. Grand Totals of all Failures

	All Totals	Percentage
3	864	1.82%
4	786	1.65%
5	835	1.76%
6	832	1.75%
7	839	1.77%
8	847	1.78%
11	894	1.88%
12	810	1.70%
13	863	1.82%
14	853	1.79%
15	1875	3.94%
16	2679	5.64%
17	867	1.82%
18	876	1.84%
19	836	1.76%
20	804	1.69%
21	2548	5.36%
22	6711	14.12%
23	850	1.79%
24	791	1.66%
27	3024	6.36%
28	7710	16.22%
29	18953	39.87%
30	15911	33.47%

whole system failure.

It is important to note that each node's assigned number is essentially arbitrary, and what is being tested is the response of a specific node to causing a system failure to identify the fault in Line 15. Figure 8 demonstrates the relationship between proximity to the link and causing a system to fail. Confidence intervals are not included with this graph nor the previous graphs illustrating the count of individual nodes to cause a system fail. This is because these graphs do not illustrate a portion of possible failures, but should constitute the entirety of failures according to the parameters of the experiment.

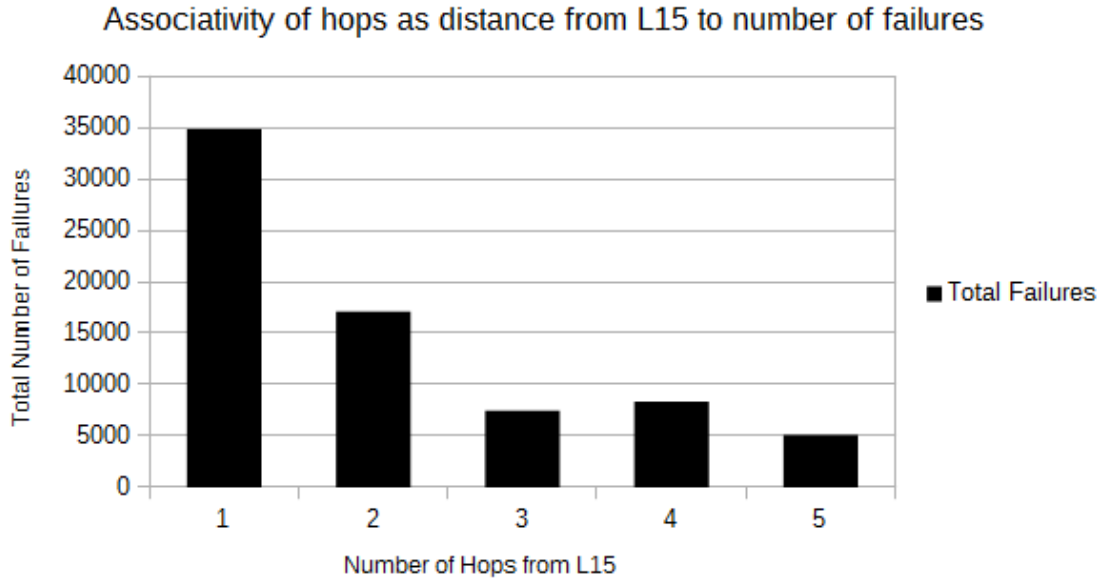


Figure 8. Likelihood of nodes to cause a system failure as distance from the test link grows

As well as verifying effectiveness of the SPIN model to find error states, it is important to examine the efficiency of the model. Processing time and memory usage were recorded for each trial, which are presented in Table 9. As would be expected, as states explored went up, both processing time and memory usage rose as well.

Plotting the memory usage and the time taken versus the states explored through

Table 9. Efficiency statistics

Type of Failure	# of failures	Time Elapsed (secs)	Memory Used (MB)	States	Errors
Single IED Component	1	0.006	2.383	96	0
	2	0.507	230.395	4560	0
	3	29.6	13321.667	142880	268
	4	73.7	31006.765	3321960	4347
Total IED	1	0	0.484	24	0
	2	0.01	6.691	276	0
	3	0.125	63.826	2024	6
	4	0.81	424.207	10626	121
	5	4.08	2101.743	42504	1151
Complex: Single, Total	1,1	0.156	69.962	2208	0
	1,2	2.24	1064.492	24288	45
	2,1	13	5711.749	100464	247
	2,2	48.2	20974.641	1056528	5552

the SPIN model show that both exhibited a similar trend on how it progresses. As seen in Figure 9 and 10, while it does rise over time, the rate slows down over time. This might appear counter-intuitive, and regularly a model checker would exhibit exponential growth in both time and memory usage, however, this is most likely a feature of SPIN’s algorithms; in order to complete the tests, the -DBITSTATE option was implemented. This changes SPIN from the memory-heavy exhaustive search that stores all states. Instead, it uses a bit-state hashing to store records of visited states, which can decrease the memory needed to finish the verification[26]. Without using the -DBITSTATE, however, the machine used to process the algorithm did not have the sufficient memory to complete a full state search of all the experiments.

5.3 Game Theory Analysis

Once the totals for each IED was discovered, the base proportional allotment can be decided. Earlier it was decided that for the purposes of this experiment, the number of resources would be an even hundred to make finding percentages easy.

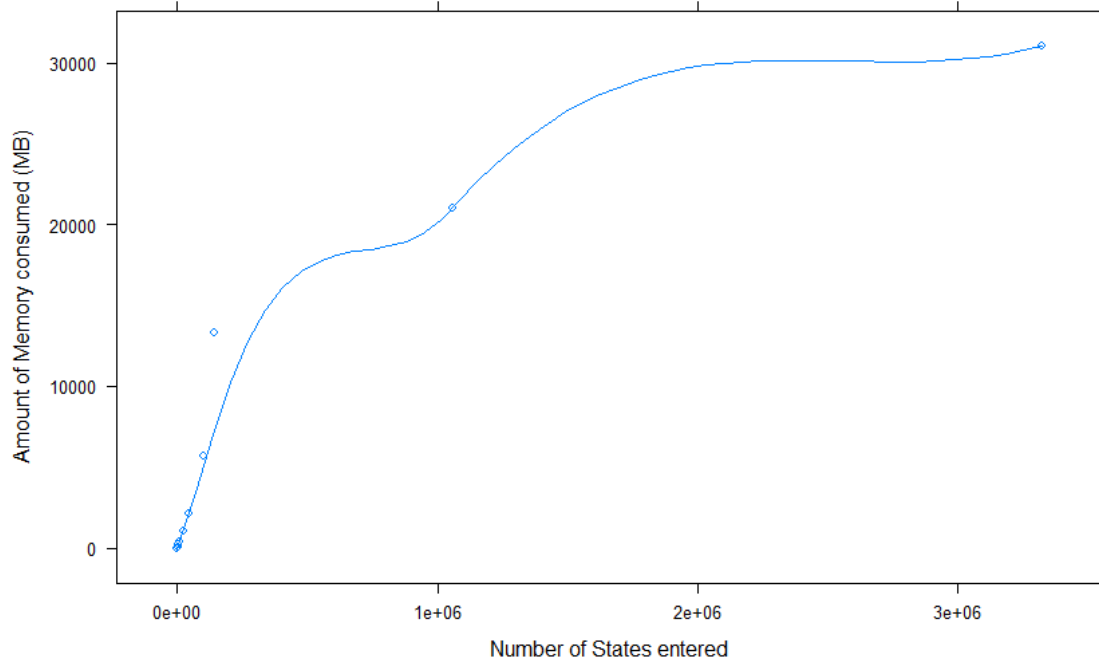


Figure 9. States explored versus the amount of memory necessary

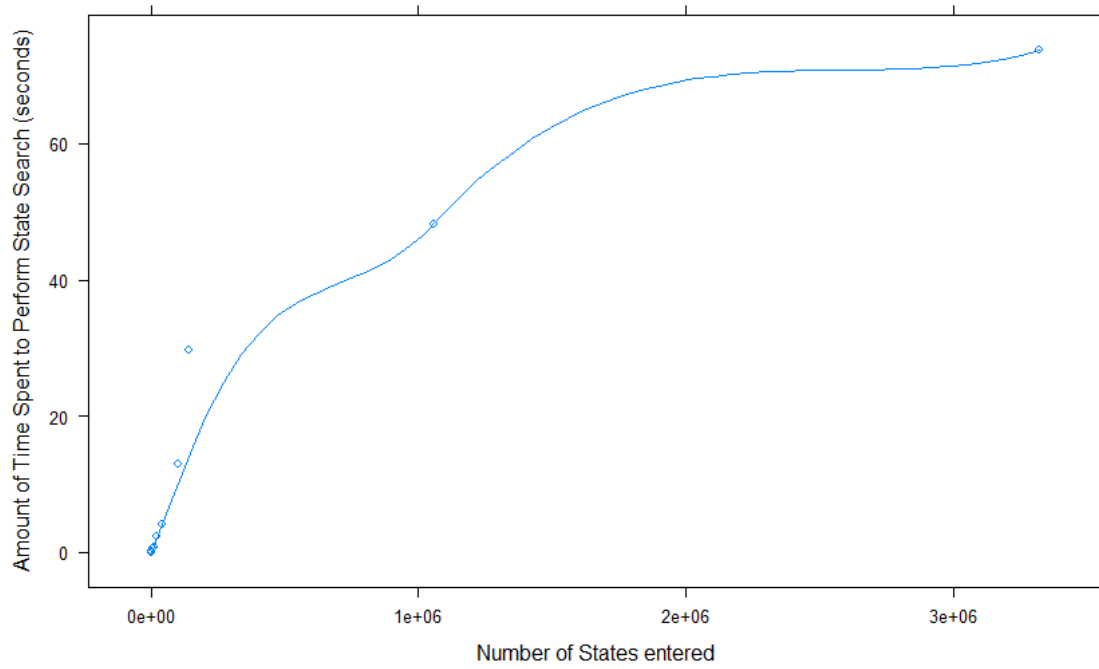


Figure 10. States explored versus the amount of time required

Running all the totals for the various types of errors got slightly different results, but a common theme as to which IEDs were most critical to the function of the system as a whole. The most critical IEDs for maintaining the link 15, not surprisingly, were IED 29 and 30, but unexpectedly, the IEDs sharing the substation with IED 29, IEDs 22 and 28, were much more critical than the IED shared at substation 10, IED 16 and its counterpart at the opposite end of the link. With the totals gained from model checking, the allotment of resources was dealt to each IED in whole numbers in terms of the likelihood to cause a system failure. In Table 13, we see Level 0, the proportional allocation scheme, a division of resources that closely resembles the percentage each node had in reference to failing fault tolerance. The subsequent levels found in Table 13 reflect answering levels of strategy in response to the previous. The visible trend is ignoring the higher valued nodes increasingly at the risk of low value nodes, but the diminishing value of continuing down this path is not securing a vast portion of the network, which could cause a system fault if enough were targeted all at once. Utilizing the failures from just the sub-component and just the total IED separately, the allocation strategies shown in Tables 10, 11, and 12 can be seen to closely parallel the overall strategy shown in Table 13. This reflects the extension of the similar distribution of faults discovered in the previous step.

Table 10. Computed allocation strategies for Total IED failures, nodes selected from 3-16

Total IEDs, 3-16												
IED Node:	3	4	5	6	7	8	11	12	13	14	15	16
Level 0	2	2	2	2	2	2	2	2	2	2	2	2
Level 1	3	0	0	3	3	3	0	3	3	3	3	0
Level 2	4	0	0	4	0	0	0	0	4	4	4	0
Level 3	5	0	0	5	0	0	0	0	5	0	5	0
Level 4	0	0	0	6	0	0	0	0	0	0	6	0

Table 11. Computed allocation strategies for Total IED failures, nodes selected from 17-30

Total IED, 17-30													
IED Node:	17	18	19	20	21	22	23	24	27	28	29	30	
Level 0	2	2	2	2	1	11	2	2	1	11	20	20	
Level 1	3	0	3	0	3	12	0	0	0	12	21	21	
Level 2	4	1	4	0	0	13	0	0	0	13	22	22	
Level 3	5	0	5	0	0	14	0	0	0	14	23	23	
Level 4	0	0	6	0	0	16	0	0	0	16	24	25	

Table 12. Computed allocation strategies for Single IED Component failures

Single IED Components																				
IED Node:	3	4-8	11	12	13	14	15	16	17	18	19	20	21	22	23	24	27	28	29	30
Level 0	1	1	1	1	1	1	3	4	1	1	1	1	4	9	1	1	5	9	29	21
Level 1	0	0	2	0	2	2	4	5	0	2	0	0	5	10	0	0	6	10	30	22
Level 2	0	0	0	0	0	0	5	6	0	0	0	0	6	11	0	0	7	11	31	23
Level 3	0	0	0	0	0	0	6	7	0	0	0	0	7	12	0	0	0	12	32	24
Level 4	0	0	0	0	0	0	0	8	0	0	0	0	8	13	0	0	0	13	33	25

Table 13. Computed allocation strategies for all failures combined

IED Node:	3	4-8	11	12	13	14	15	16	17	18	19	20	21	22	23	24	27	28	29	30
Level 0	1	1	1	1	1	1	3	4	1	1	1	1	4	9	1	1	4	11	26	23
Level 1	2	0	2	0	0	0	4	5	2	2	0	0	5	10	0	0	5	12	27	24
Level 2	0	0	0	0	0	0	5	6	0	0	0	0	6	11	0	0	6	13	28	25
Level 3	0	0	0	0	0	0	5	7	0	0	0	0	0	12	0	0	7	14	29	26
Level 4	0	0	0	0	0	0	6	0	0	0	0	0	0	13	0	0	8	16	30	27

Another interesting discovery in the implementation of the ILP was that though it started with proportional allocation that was fed in, the resources that were freed between each level was allocated evenly across each node, instead of being distributed proportionally to the most relevant nodes. This is in consequence to the math in the ILP. Each level is designed to specifically outperform the level preceding it, so a succeeding level just needs to beat the resources distributed in the higher nodes by one while abandoning nodes that still allow it to win the overall majority of battlefields in question. As a comparison, I built a single table of allocation strategies outside of the ILP, that would distribute the resources proportionally as the levels increased. The result shown in Table 14 is that the most critical nodes received more resources at a higher rate, but the overall strategies abandoned the less important nodes at a higher rate as well. Comparing the performance of the two allocation strategies is not considered within the scope of this paper, but the different strategy is left to illustrate an alternative method that may be explored in the future.

Table 14. Alternate allocation strategies based on simple proportional allocation
Straight Proportional Allocation

IED Node:	3	4-8	11	12	13	14	15	16	17	18	19	20	21	22	23	24	27	28	29	30
Level 0	1	1	1	1	1	1	3	4	1	1	1	1	4	9	1	1	4	11	26	23
Level 1	1	0	1	0	0	0	3	4	1	1	0	0	4	10	0	0	4	13	31	27
Level 2	0	0	0	0	0	0	0	4	0	0	0	0	4	11	0	0	4	15	35	30
Level 3	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	16	39	33

Deciding which level of strategy to implement is a question that needs to be answered. Previous research found that the third level of strategy was most effective against an average of skilled attacks[22], which in this case would mean placing defensive resources at a third of the available nodes. This would still maintain a coverage of the nodes most likely to be a damaging loss, while still ensuring sufficient resources are in place for the most critical assets. In the next section, the research will go over a theoretical model of how each level would respond to an attack.

5.4 Case Studies in Defense

After collecting the results of each allocation strategy, the numbers are placed against the theoretical model proposed in the test scenario. Each scenario will be reviewed in turn.

5.4.1 Case Study A.

With case study A, the probability of detecting an attack as seen in Figure 11 was very low initially for an equal distribution. In the case study, the attacker went after an important node with an easier to detect attack, and was only 63% as likely as the level 0 proportional defense strategy. The remaining hours into the attack showed a decreasing rate of return as the defensive strategies changed levels.

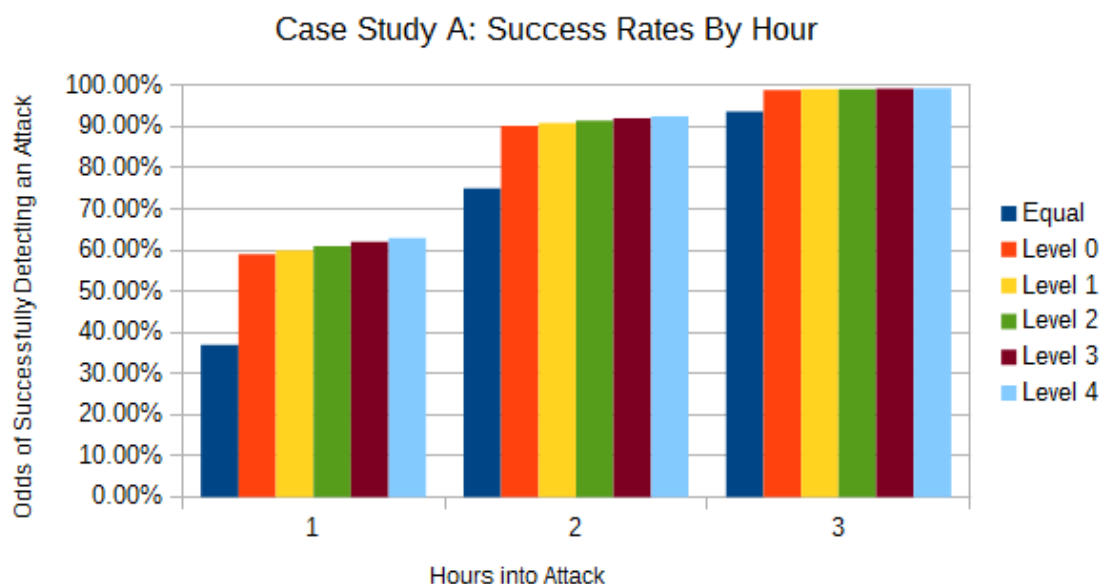


Figure 11. Likelihood to detect an attack in case study A with each strategy

5.4.2 Case Study B.

Case study B presents an entirely different story. In this scenario, an attacker starts at a lower priority node using stealthier techniques, and moves towards their target, heading for the higher priority goal. In here, as can be viewed in Figure 12, the proportional strategies do not detect the initial entry, but the first two downed sub-components would not induce a fault yet. Also, there is a drop in likelihood for the highest levels, implying that potentially too many surrounding nodes were ignored in the allocation strategy. Over time, the higher levels will still beat out the equal allocation when it gets to an actual fault condition. At the end, the equal allocation strategy performs almost 82% as well as the next lowest allocation strategy

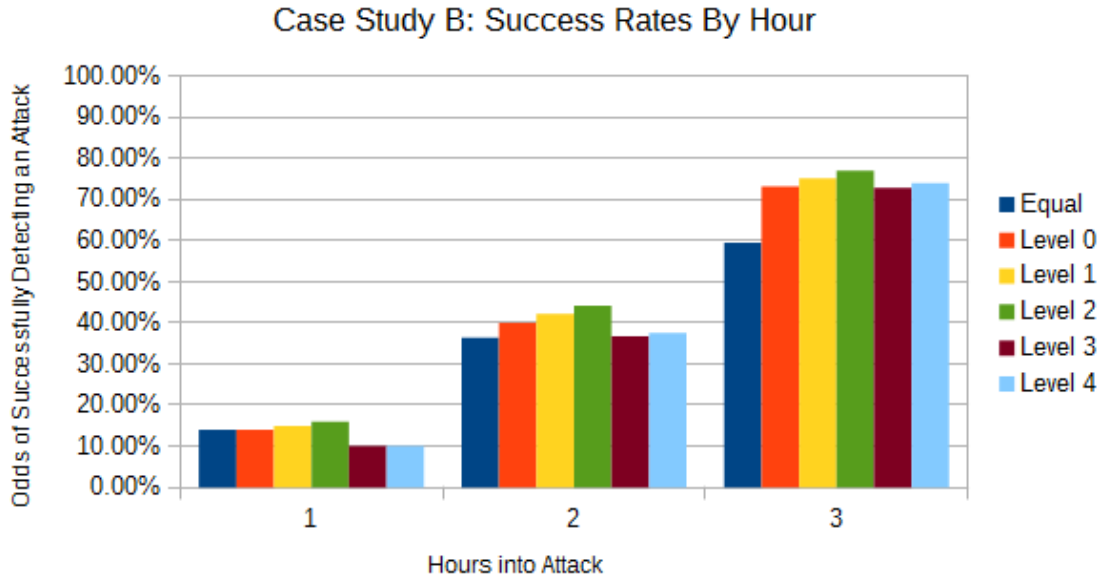


Figure 12. Likelihood to detect an attack in case Study B with each strategy

5.4.3 Case Study C.

Case Study C shows the biggest potential of the developed allocation strategy. In this scenario, the attacker used stealthy techniques that centered around the target.

As the attacker focused on nodes that had higher resources when proportionally allocated, equal allocation only ended up performing 39% as well initially as the next lowest level, and then 64% as well in the end.

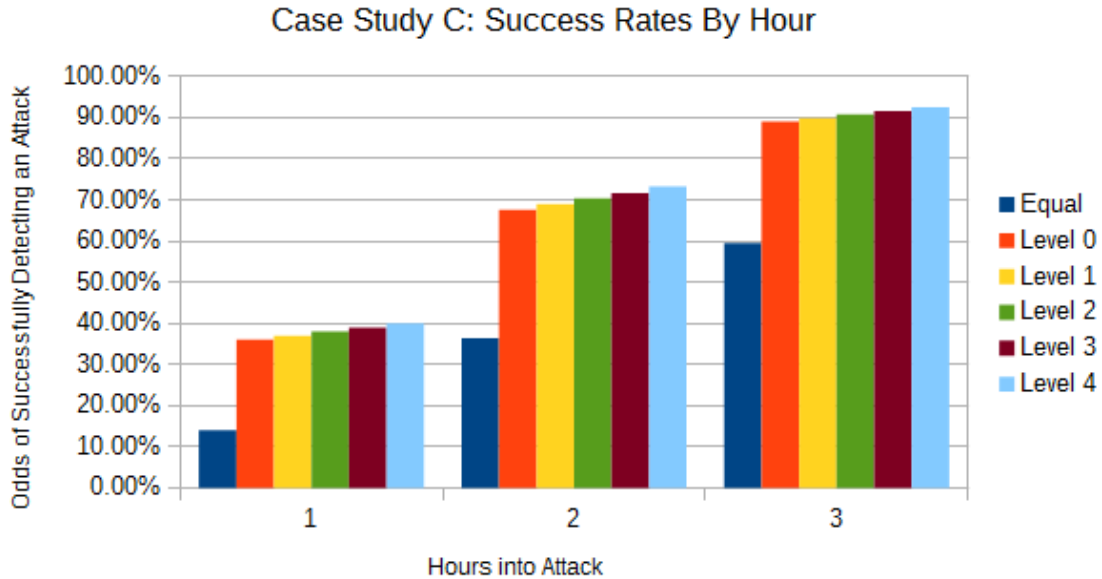


Figure 13. Likelihood to detect an attack in case study C with each strategy

After looking at the probabilities, there is an increasing chance of detecting an attack with each step in levels of strategy when an attacker focuses on high value targets, but there is a decreasing rate of return as the levels increase. When the attacker does not attack the high value targets, the differences in the strategies' capabilities are in some cases worse; however, these situations are much less likely to cause the system to stop functioning correctly.

To examine the statistical difference between the equal and ILP-generated graphs, the attack strategies were all run equally through the higher and lower detection systems, and the totals were all put together to generate a box plot of how the strategies worked in comparison. Figures 14 and 15 show the results of plotting the end predicted chance to discover an attack. A statistical difference is seen between

the equal allocations and the level- k strategies. However, no statistical difference was discovered between the different levels. This could be explained by the low number of tests performed; but more likely due to the slowly incrementing numbers of defensive resources as the levels increased.

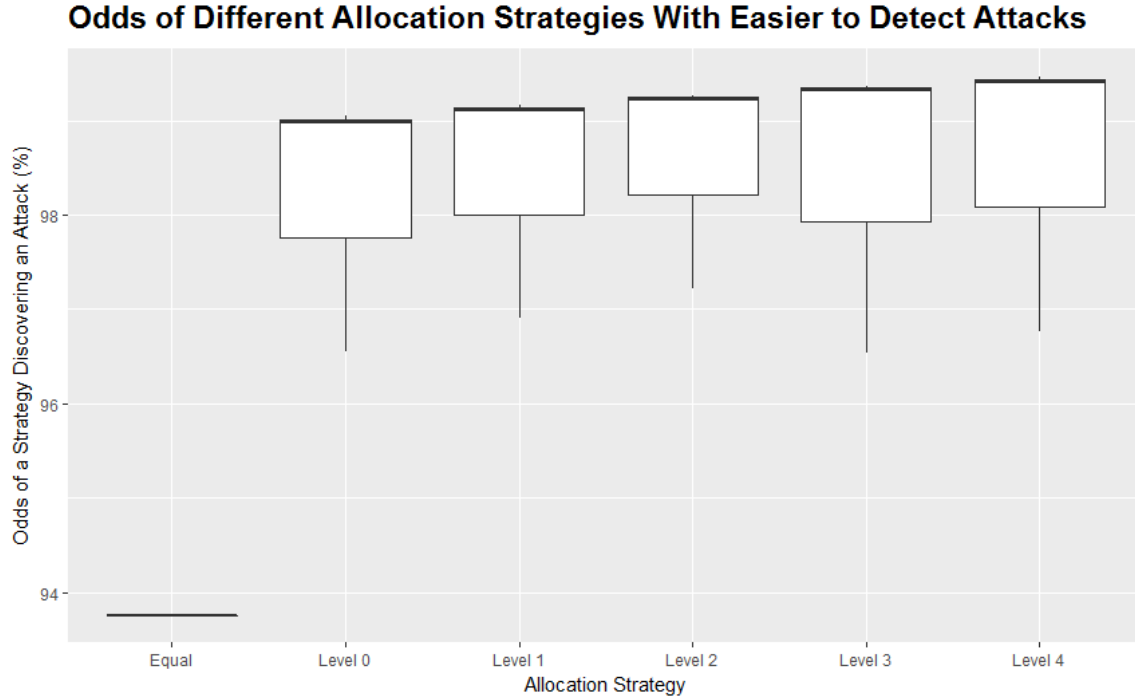


Figure 14. Comparison of defense strategies using higher rates of successful detection

To return to the alternate allocation strategy that was devised to explore continuing the proportional distribution, the strategies were put through the same case studies as the ILP derived allocations. Seen below in Figures 16, 17, and 18. With these case studies, it is easy to visualize that the drop in coverage of less critical nodes so quickly can reduce the odds of successfully detecting an attack overall when it's focused on these outer edge cases. Using the limited case studies, it would appear likely that the strategies devised by the ILP perform better than the alternate strategy that continues the initial proportional allocations. Overall, it generally appears to perform better than the equal distribution strategy.

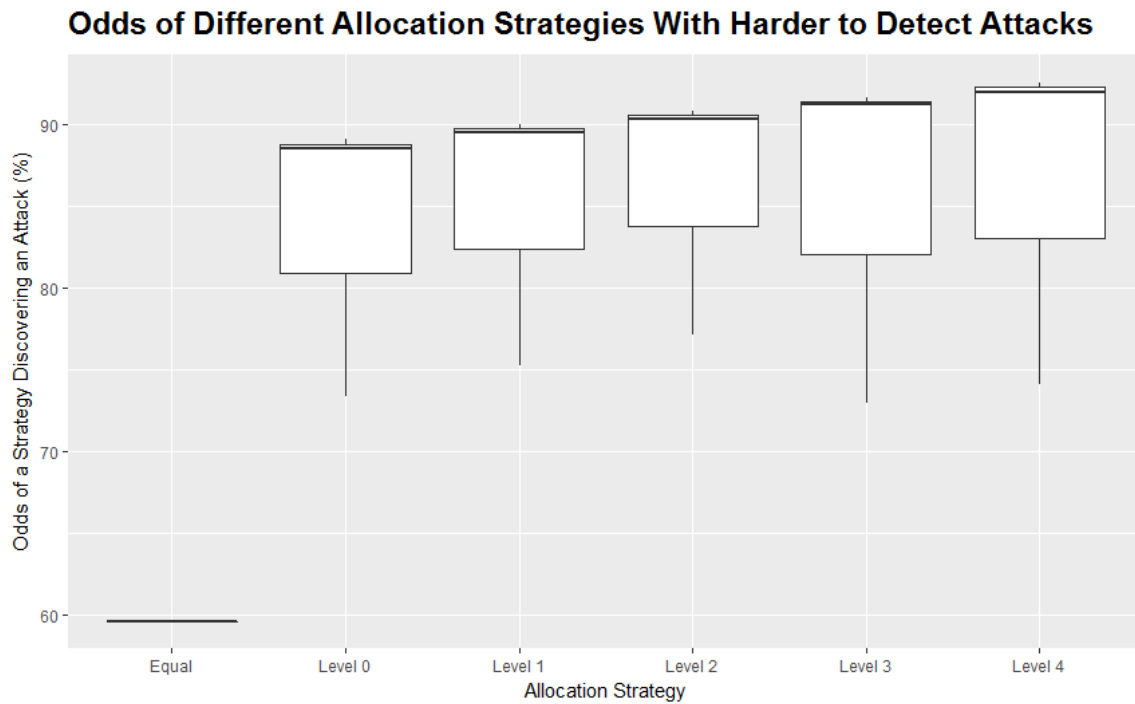


Figure 15. Comparison of defense strategies using lower rates of successful detection

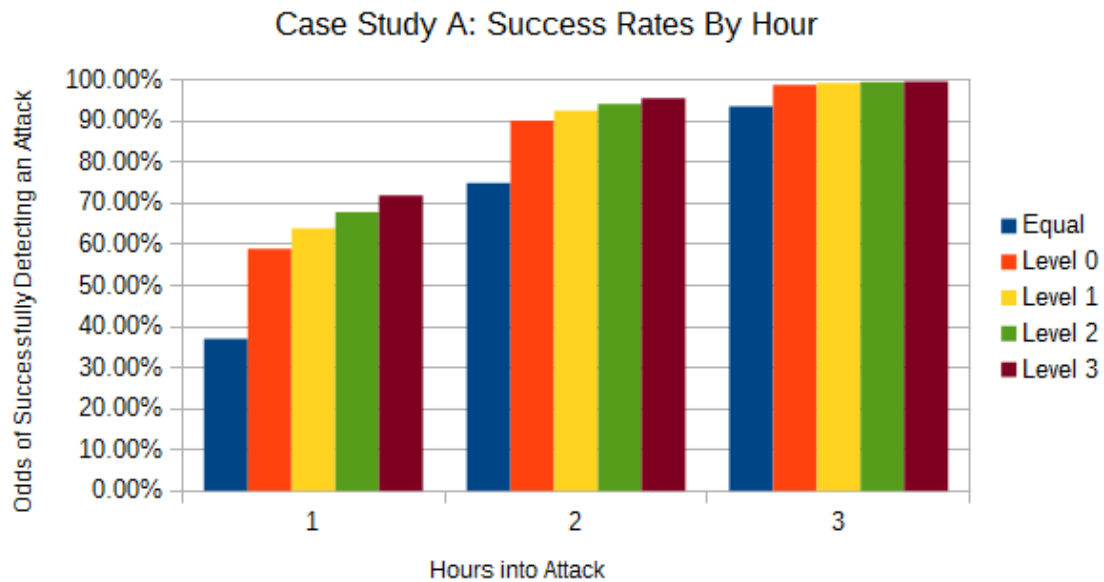


Figure 16. Likelihood to detect an attack in case study A with the alternate allocation strategy

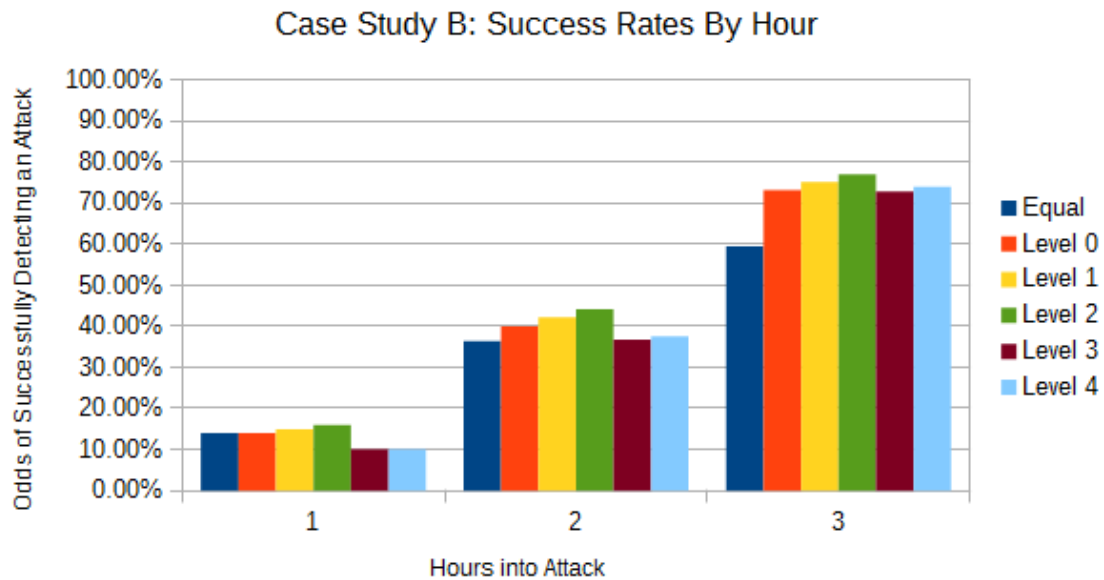


Figure 17. Likelihood to detect an attack in case Study B with the alternate allocation strategy

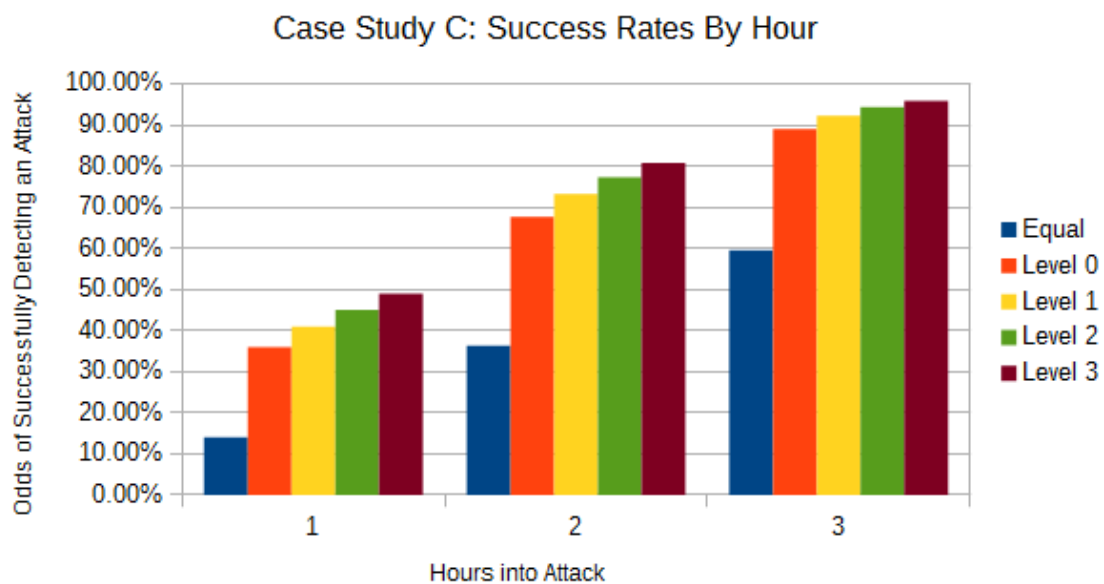


Figure 18. Likelihood to detect an attack in case study C with the alternate allocation strategy

5.5 Summary

This chapter provided the results of applying the model checking on a WABPS system to discover fault intolerance in individual nodes, applying game theory to make defensive strategies, and demonstrating the end results with a case study. The feasibility of the process as a whole is demonstrated through the end results of how it performed against example attack scenarios.

The final results speak to using a computer science technique, model checking, as a viable method to finding weak links in a distributed system, and that game theory serves to use these findings to evaluate various levels of defensive unit allocation strategies. The process also highlights the benefits of a defensive strategy based on observable characteristics of a system. All of these findings are useful for owners of critical infrastructure systems to find risk and propose a quantitatively-demonstrative method to prove it.

VI. Conclusion and Future Work

6.1 Overview

This section will provide the conclusions gained from the results of this research. It will provide analysis on how each research goal was fulfilled, and present ideas for future work.

6.2 Conclusions of Research

These are the research goals proposed in chapter 1:

1. Develop an algorithm for individual node fault detection in a distributed system through the use of model checking technology.
 - The model made of the WABPS, when run through a model checker, provides not just the fault tolerance of the system as a whole, but can be extended to provide statistics on how each specific subcomponent is likely to impact the overall system functionality. This allows the system owner to either make corrections to strengthen weaker spots, put in countermeasures, or assume the risk from leaving it unguarded.
2. Develop an algorithm to apply behavioral game theory reasoning using information on discovered vulnerabilities to formulate a defensive resource allocation strategy.
 - When the data for all the nodes is collected, the owner is presented a number of levels that define allocation strategies in a method that focuses on the points most likely to damage the system. The owner is presented with various levels to choose from.

3. Demonstrate the end viability of the process by using the proposed allocation strategy in defense against a theoretical attack against the WABPS.
- After looking at the probability that each strategy will detect an attack in progress, it is visible that rejecting an equal distribution when nodes have unequal worth for the proportional distribution will perform much better at protecting the most critical nodes. As the levels of defensive strategy increase, the odds of detecting attacks on the critical nodes become more favorable for the defender, but at the cost of losing ancillary nodes. If an attack focused on the nodes less likely to cause a system fault, the differences drop.

6.3 Recommendations for Future Work

There are a few limitations with this research that deserve further exploration. For one, this process could be expanded to different systems, including production industrial control systems in place. Also, the system could be functionally modeled in a test platform and have the attacks presented to verify the statistical predictions against practical findings. Here are some specific ways in which this research can be extended:

- Introduce the system model into the EPOCHS network simulator. The third research goal presented the hypothetical modeling of an attack against the system, and how this framework would succeed in detecting it. This virtual model would present a cost effective way to model a physical build, and present a practical demonstration of various types of a cyber attack against the system in question.
- Provide additional models to pass through the framework. There are large numbers of unique industrial control networks in existence and in development.

This framework would be further strengthened by testing it against a variety of systems to demonstrate its general applicability and feasibility.

- Adjust the SPIN model to verify that the association of hops to the node hold true throughout the model.
- Adjust the criteria used in developing the worth of each component. This model assumes the worth of each node based on its susceptibility to causing a system failure. In truth, there could be a variety of inputs that derive the inherent value of a node, such as the amount of power generation, dependence of local infrastructure, cost of replacement, and strategic objectives. Further research could be done to demonstrate the manipulation of node value for dividing resources based upon a larger array of parameters.
- Test the results of the mathematically generated findings against a group of human attackers, via a study. This study might involve polling an audience on what methods or strategies they would use to initiate an attack, or permit them access to a virtual model to perform attacks in a controlled setting. This would serve as an additional verification of live human behavior as opposed to general estimation.

6.4 Final Thoughts

Critical infrastructure engineers typically are electrical engineers first, used to dealing with systems that potentially last decades, with little experience in the fast evolving IT world. In order for these engineers to be able to effectively find vulnerabilities and understand how to divvy up finite monetary resources in defending a large, distributed network, they need a good framework that is simple to learn and apply, which can be used to model any system. This framework offers a potential

solution that critical infrastructure engineers can apply to proprietary systems and be able to present clear reasoning for how to defend their network to system owners, insurance providers and governmental regulations.

Critical infrastructure is a vital part of modern society's way of life, and safeguarding these systems should be a high priority. However, there is a lot of uncertainty concerning the growing confluence of IT with traditionally lower-tech industrial control systems. The advantages of understanding the importance of specific components and their impact to the overall functionality allows system designers and maintainers a greater confidence in protecting their networks despite the rapidly changing IT world.

Appendices

A GAME THEORY MODEL FOR ALLOCATING SCARCE RESOURCES IN CRITICAL INFRASTRUCTURE PROTECTION

A. Commands to Implement SPIN

These are the commands necessary to compile and run the model through SPIN. The additional commands in the second line help streamline the compiled code to use less memory. The commands in the third line instruct SPIN to examine all error states and print out a trail file for each faulted state.

```
$ spin -a wabps_model.pml
```

```
$ gcc -O2 -DBITSTATE -DNOREDUCE -o pan pan.c
```

```
$ ./pan -c0 -e
```

B. Script for Reading Trail Files

Shell script to read through and find details of each error stored in the SPIN trail files and combine them into one central document

```
#!/bin/bash
for i in {1..# of files to parse}
do
spin -k wabps_model.pml${i}.trail -c wabps_model.pml
    | head >> Trails.txt
done
```

C. Example Printout of Faulted States

```
proc 0 = :init:
using statement merging
    Corrupted Component IED29, dir = LINE_FAULT
    Corrupted Component IED28, dir = LINE_FAULT
    Turned Off IED29
    Turned Off IED30
proc 1 = LDA15
proc 2 = LDA12
proc 3 = LDA14
proc 4 = LDA09
proc 0 = :init:
using statement merging
    Corrupted Component IED29, dir = LINE_FAULT
    Corrupted Component IED28, dir = NONE
    Turned Off IED29
    Turned Off IED30
proc 1 = LDA15
proc 2 = LDA12
proc 3 = LDA14
proc 4 = LDA09
```

D. Example of Attack Detection Chart

This chart shows the likelihood that an attack will not be detected per hour on an IED.

Scenario 2	Chance of not discovering attack per hour					
IED	1	2	3	4	5	6
3	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
4	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
5	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
6	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
7	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
8	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
11	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
12	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
13	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
14	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
15	84.00%	70.56%	59.27%	49.79%	41.82%	35.13%
16	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
17	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
18	90.00%	81.00%	72.90%	65.61%	59.05%	35.13%
19	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
20	90.00%	81.00%	72.90%	65.61%	37.07%	53.14%
21	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
22	77.00%	59.29%	45.65%	35.15%	32.77%	20.84%
23	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
24	90.00%	81.00%	72.90%	65.61%	59.05%	53.14%
27	82.00%	67.24%	55.14%	45.21%	37.07%	30.40%
28	74.00%	54.76%	40.52%	29.99%	22.19%	16.42%
29	60.00%	36.00%	21.60%	12.96%	7.78%	4.67%
30	63.00%	39.69%	25.00%	15.75%	9.92%	6.25%

Figure 19. Likelihood of failure to detect an attack, Level-4 strategy, Case Study C

Bibliography

1. Cplex optimizer. World Wide Web Page. Available at <https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/>.
2. Microsoft visual studio. World Wide Web Page. Available at <https://www.visualstudio.com/>.
3. Nexdefense. World Wide Web Page. Available at <http://www.nexdefense.com/>.
4. Palvi Aggarwal, Zahid Maqbool, Antra Grover, V. S Chandrasekhar Pammi, Saumya Singh, and Varun Dutt. Cyber security: A game-theoretic analysis of defender and attacker strategies in defacing-website games. *2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment, CyberSA 2015*, 2015.
5. Aleksandr A. Kerzhner, Kymie Tan, and Elyse Fosse. Analyzing Cyber Security Threats on Cyber - Physical Systems using Model - Based Systems Engineering. In *AIAA SPACE 2015 Conference and Exposition*, pages 1–11, 2015.
6. Luiz DaSilva Alle MacKenzie. *Game Theory for Wireless Engineers*. Morgan & Claypool, 2006.
7. Ayala Arad and Ariel Rubinstein. The 11 20 Money Request Game :. *American Economic Review*, 102(7):3561–3573, 2012.
8. Scott Backhaus, Russell Bent, James Bono, Ritchie Lee, Brendan Tracey, David Wolpert, Dongping Xie, and Yildiray Yildiz. Cyber-physical security: A game theory model of humans interacting over control systems. *IEEE Transactions on Smart Grid*, 4(4):2320–2327, 2013.
9. J C Campelo, F Rodríguez, P J Gil, and J J Serrano. Design and Validation of a Distributed Industrial Control System’s Nodes. *Reliable Distributed Systems*, pages 300–301, 1999.
10. Francisco Chicano and Enrique Alba. Finding liveness errors with ACO. *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, pages 2997–3004, 2008.
11. Joost-Pieter Katoen Christel Baier. *Principles of Model Checking*. MIT Press, 2008.
12. David Rios Insua David L. Banks, Jesus Rios. *Adversarial Risk Analysis*. CRC Press, Boca Raton, FL, 2016.
13. S. E. Donaldson, S. G. Siegel, C. K. Williams, and A. Aslam. *Enterprise Cybersecurity: How to Build a Successful Cyberdefense Program Against Advanced Threats*. Apress, 2015.

14. Arthur Conan Doyle. *The Memoirs of Sherlock Holmes*. George Newnes, London, England, 1894.
15. Tom Liston Ed Skoudis. *Counter Hack Reloaded*. Prentice Hall Press, Upper Saddle River, NJ, 2005.
16. E. Estévez and M. Marcos. Model-based validation of industrial control systems. *IEEE Transactions on Industrial Informatics*, 8(2):302–310, 2012.
17. Sean Gallagher. Whos winning the cyber war? the squirrels, of course, Jan 2017.
18. J Duncan Glover, Mulukutla S Sarma, and Thomas J Overbye. *Power Systems Analysis and Design*. Cengage Learning, Stamford, CT, 2012.
19. William Goldman. *The Princess Bride*. Harcourt Brace Jovanovich, 1973.
20. O. Gross and R. Wagner. A continuous colonel blotto game. *RAND Corporation*, 1950.
21. Abhishek Gupta, Galina Schwartz, Cédric Langbort, S. Shankar Sastry, and Tamer Başar. A three-stage Colonel Blotto game with applications to cyberphysical security. *Proceedings of the American Control Conference*, pages 3820–3825, 2014.
22. Seth Hamman. *Improve the Cybersecurity of Cyber-Physical Systems Through Behavioral Game Theory and Model Checking in Practice and in Education*. PhD thesis, Air Force Institute of Technology, 2016.
23. Klaus Havelund, Mike Lowry, and John Penix. Formal analysis of a space craft controller using spin. In *In Proceedings of the 4th SPIN workshop*, 1998.
24. Rattikorn Hewett, Sudeeptha Rudrapattana, and Phongphun Kijsanayothin. Smart Grid security: Deriving informed decisions from cyber attack game analysis. *2014 IEEE International Conference on Smart Grid Communications, Smart-GridComm 2014*, pages 946–951, 2015.
25. Gerald J. Holzmann. *The Spin Model Checker*. Addison-Wesley Professional, 2004.
26. Gerard J. Holzmann. An analysis of bitstate hashing, 1995.
27. Gerard J. Holzmann and Margaret H. Smith. Automating software feature verification. *BELL LABS TECHNICAL JOURNAL*, 5:72–87, 2000.
28. Edward J. Holmes Juan M. Gers. *Protection of Electricity Distribution Networks, 3rd Edition*. IET, Herts, United Kingdom, 2011.

29. Stamatis Karnouskos. Stuxnet worm impact on industrial cyber-physical system security. *IECON Proceedings (Industrial Electronics Conference)*, pages 4490–4494, 2011.
30. Pim Kars. The application of promela and spin in the bos project, 1996.
31. Leslie Lamport. TLA in Pictures. *IEEE Transactions on Software Engineering*, 21(9):768–775, 1995.
32. Leslie Lamport. Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers [Book Review]. *Computer*, 35(9):81–81, 2002.
33. Barry Long, Juergen Dingel, and T.C. Nicholas Graham. Experience applying the SPIN model checker to an industrial telecommunications system. *Proceedings of the 13th international conference on Software engineering - ICSE '08*, page 693, 2008.
34. M. Lukacs, P. Collins, and M. Temple. Device identification using active noise interrogation and rf-dna ”fingerprinting” for non-destructive amplifier acceptance testing. In *2016 IEEE 17th Annual Wireless and Microwave Technology Conference (WAMICON)*, pages 1–6, April 2016.
35. Robert Mitchell and Ing-Ray Chen. Effect of Intrusion Detection and Response on Reliability of Cyber Physical Systems. *IEEE Transactions on Reliability*, 62(1):199–210, 2013.
36. Chris Newcombe, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker, and Michael Deardeuff. Engineers use TLA+ to prevent serious but subtle bugs from reaching production. How Amazon Web Services Uses Formal Methods. *Communications of the Acm*, 58(4), 2015.
37. Martin J. Osborne. *An Introduction to Game Theory*. Oxford University Press, New York, New York, 2004.
38. Tamal Paul, Jonathan W. Kimball, Maciej Zawodniok, Thomas P. Roth, Bruce McMillin, and Sriram Chellappan. Unified invariants for cyber-physical switched system stability. *IEEE Transactions on Smart Grid*, 5(1):112–120, 2014.
39. Robert Powell. Sequential, nonzero-sum ”Blotto”: Allocating defensive resources prior to attack. *Games and Economic Behavior*, 67(2):611–615, 2009.
40. Mason Rice, Daniel Guernsey, and Sujeet Sheno. *Using Deception to Shield Cyberspace Sensors*, pages 3–18. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
41. Dave Richardson. Spin pc installation instructions.

42. Francis Schneider, Steve Hf. Easterbrook, John R. Callahan, and Gerard J. Holzmnn. Validating requirements for fault tolerant systems using model checking, 1998.
43. Xiaojun Shan and Jun Zhuang. Hybrid defensive resource allocations in the face of partially strategic attackers in a sequential defender-Attacker game. *European Journal of Operational Research*, 228(1):262–272, 2013.
44. Huiling Shi. A Case Study of Model Checking Retail Banking System with SPIN. *Journal of Computers*, 7(10):2503–2510, 2012.
45. Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. {PROTECT}: A Deployed Game Theoretic System to Protect the Ports of the {U}nited {S}tates. *Aamas*, pages 13–20, 2012.
46. Y. Sun, B. McMillin, X. Liu, and D. Cape. Verifying noninterference in a cyber-physical system the advanced electric power grid. In *Seventh International Conference on Quality Software (QSIC 2007)*, pages 363–369, Oct 2007.
47. Lior Tabansky. Critical Infrastructure Protection against Cyber Threats. *Military and Strategic Affairs*, 3(2):61–78, 2011.
48. Han Tang and Bruce M. McMillin. Security property violation in CPS through timing. *Proceedings - International Conference on Distributed Computing Systems*, pages 519–524, 2008.
49. Xiaoyang Tong, Xiaoru Wang, Rui Wang, Fei Huang, Xueyuan Dong, Kenneth M. Hopkinson, and Gongyi Song. The study of a regional decentralized peer-to-peer negotiation-based wide-area backup protection multi-agent system. *IEEE Transactions on Smart Grid*, 4(2):1197–1206, 2013.
50. Dong Qing Wang, Shi Hong Miao, Xiang Ning Lin, Pei Liu, Ying Xin Wu, and Dan Yang. Design of a novel wide-area backup protection system. *Proceedings of the IEEE Power Engineering Society Transmission and Distribution Conference*, 2005:1–6, 2005.
51. Eric Ke Wang, Yunming Ye, Xiaofei Xu, S. M. Yiu, L. C. K. Hui, and K. P. Chow. Security Issues and Challenges for Cyber Physical System. *2010 IEEE/ACM Int’l Conference on Green Computing and Communications & Int’l Conference on Cyber, Physical and Social Computing*, pages 733–738, 2010.
52. Michael M Winn. *CONSTRUCTING COST-EFFECTIVE AND TARGETABLE ICS HONEYPOTS SUITED FOR PRODUCTION NETWORKS*. PhD thesis, Air Force Institute of Technology, 2015.

53. M. Yampolskiy, P. Horvath, X. D. Koutsoukos, Y. Xue, and J. Sztipanovits. Systematic analysis of cyber-attacks on cps-evaluating applicability of dfd-based approach. In *2012 5th International Symposium on Resilient Control Systems*, pages 55–62, Aug 2012.
54. Hehua Zhang, Stephan Merz, and Ming Gu. Specifying and verifying PLC systems with TLA+ : A case study. *Computers and Mathematics with Applications*, 60(3):695–705, 2010.
55. Yichi Zhang, Lingfeng Wang, Yingmeng Xiang, and Chee-wooi Ten. Inclusion of SCADA Cyber Vulnerability in Power System Reliability Assessment Considering Optimal Resource Allocation. *IEEE Transactions on Power Systems*, 31(6):1–16, 2016.

REPORT DOCUMENTATION PAGE					Form Approved OMB No. 0704-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.						
1. REPORT DATE (DD-MM-YYYY)		2. REPORT TYPE		3. DATES COVERED (From — To)		
23-03-2017		Master's Thesis		Aug 2015 — Mar 2017		
4. TITLE AND SUBTITLE A Game Theory Model for Allocating Scarce Resources in Critical Infrastructure Protection				5a. CONTRACT NUMBER		
				5b. GRANT NUMBER		
				5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S) Chappell, Richard E II, Capt, USAF				5d. PROJECT NUMBER 16G178		
				5e. TASK NUMBER		
				5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT-ENG-MS-17-M-012		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory, Information Directorate Attn: Michael Gudaitis 525 Brooks RD Rome, NY 13441 (315) 330-4478 (587-4478/ michael.gudaitis@us.af.mil)				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RITE		
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)		
12. DISTRIBUTION / AVAILABILITY STATEMENT DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.						
13. SUPPLEMENTARY NOTES This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.						
14. ABSTRACT In this paper, we describe, implement and evaluate a process that tests the fault tolerance of a distributed system, and then applies behavioral game theory to evaluate specific resource allocation problems. Geographically dispersed power grid systems brings unique challenges in deciding where and how to protect assets. Model checking abstracts a system to a logical framework that can be tested for failures. Allocation strategies can then be derived from the comparison of individual elements. Level- k reasoning is then applied to discover iterative refinements to the allocation strategies, until a specific result is established. Case studies are presented that demonstrates how to apply this methodology to evaluate a system.						
15. SUBJECT TERMS Critical Infrastructure, Model Checking, Game Theory, Level- k , Resource Allocation						
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON	
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Kenneth Hopkinson, AFIT/ENG	
U	U	U	U	81	19b. TELEPHONE NUMBER (include area code) (937) 255-3636, x4555; kenneth.hopkinson@afit.edu	