6-13-2013

Applied Hypergame Theory for Network Defense

Alan S. Gibson

APPLIED HYPERGAME THEORY FOR NETWORK DEFENSE

THESIS

Alan S. Gibson, USAF

AFIT-ENG-13-J-02

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-13-J-02

APPLIED HYPERGAME THEORY FOR NETWORK DEFENSE

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 Computer Engineering

Alan S. Gibson, B.S.C.S.E.

USAF

June 2013

AFIT-ENG-13-J-02

APPLIED HYPERGAME THEORY FOR NETWORK DEFENSE

Alan S. Gibson, B.S.C.S.E.
USAF

Approved:

/signed/	6 Jun 2013
Gary Lamont, PhD (Chairman)	Date
/signed/	6 Jun 2013
Maj Jonathan W. Butts, PhD (Member)	Date
/signed/	6 Jun 2013
Meir Pachter, PhD (Member)	Date

Abstract

Cyber operations are the most important aspect of military conflicts in the 21st century, but unfortunately they are also among the least understood. They encompasses a wide range of situations up to and including the complicated endeavor of computer network defense. The continual battle for network dominance between attackers and defenders is considered to be a complex game. This viewpoint has led to the rise of game theory principles being applied to cyber warfare scenarios. Game theory, with its ability to provide a mathematical analysis of game models, is a device that helps to increase defensive capabilities.

Hypergame theory is an extension of game theory that addresses the kind of games where misperception exists, as is often the case in military engagements. Hypergame theory, like game theory, uses a game model to determine strategy selection, but goes beyond game theory by examining subgames that exist within the full game. Analysis of hypergames offers advantageous reasoning for strategy selection during conflicts through robust situational awareness. The inclusion of misperception and misinformation in the hypergame model shows the usefulness of the theory because it provides strategy selection techniques that are designed to mitigate risks.

This goal of this research is to provide a foundation for the use of hypergame theory for adaptable cyber defense systems able to adjust to the existence of misperceptions and misinformation. A general hypergame model, created around a cyber attack and defense scenario, influences the design of an innovative software tool for hypergame analysis. Extensive testing validates that the custom software achieves hypergame model requirements. The game-based software simulation is achieved by running iterations of game play where adaptability of strategy selection is the key focus. The results discovered in this research effort are an indication that there is utility in applying hypergame theory to the study of network defense techniques. This innovative research indicates that the

application of hypergame theory is able to achieve the adaptive nature required for today's and tomorrow's cyber defense needs.

Acknowledgments

I would like to thank my committee members for their time, Major Jonathan Butts, Professor Meir Pachter, and my advisor Professor Gary Lamont, for his vision and direction.

Alan S. Gibson

Table of Contents

Page

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
NE	Nash Equilibrium
PSNE	pure strategy Nash equilibrium
MSNE	mixed strategy Nash equilibrium
OODA	Observe-Orient-Decide-Act
IDE	Integrated Development Environment
HNF	Hypergame Normal Form
CMS	Column Mixed Strategy
RMS	Row Mixed Strategy
R	rational
GMR	general metarational
SMR	symmetric metarational
FHQ	sequential stable
G	worst case utility
HEU	Hypergame Expected Utility
MO	Modeling Opponent
PS	Pick Subgame
WS	Weighted Subgame
EU	Expected Utility
HML	Hypergame Markup Language
SPA	Security Policy Assistant
XML	Extensible Markup Language
HAT	HNF Analysis Tool
SCADA	Supervisory Control and Data Acquisition

Acronym	Definition
PLC	programmable logic controller
US-CERT	United States Computer Emergency Readiness Team
CSV	comma separated values

APPLIED HYPERGAME THEORY FOR NETWORK DEFENSE

I. Introduction

CYBER operations are the most important aspect of military conflicts in the 21st century, but unfortunately they are also among the least understood. The term "cyber operations" is not even fully defined and can encompass a wide range of situations and targets [32]. Even given this fact, cyber warfare at the very least covers the area of computer network defense. Defense of networks is a complicated endeavor, which is a reason why many ways of accomplishing this task have been explored. Game theory, with its ability to provide a mathematical analysis of conflict, is being explored as an option that increases defense capabilities. "Security at network layer imposes future challenges for us to address security at a larger and more complex scale and game theory provides a preliminary tool that enables a quantitative study of such complex systems" [33]. The fusion of game theoretic techniques and network defense systems is destined to become the backbone of any robust cyber defense technique.

1.1 State of Cyber Defense

The rise of cyber systems as the backbone of society has led to the information age, which is more appropriately named the information reliance age. Indeed, one can consider the simple road map as an example; it is a casualty of the rise of the computerized device. The average U.S. automobile these days does not contain a road map because of the ease of using a GPS device or more commonly an individual's cell phone. Computer networks infiltrate almost every aspect of people's lives, running businesses and governments alike. Almost daily we are given indicators that poor defense of networks creates the possibility of

disastrous results. A recent occurrence includes Pentagon reports of U.S. weapon systems being hacked [20]. This reliance is cause for alarm, but only because these systems, that are so important to people's daily lives, can be attacked, damaged and destroyed. Less reliance on computer networks at this point is not a feasible option; therefore, innovations in security and defense must continue.

On April 27th, 2007, the country of Estonia learned this lesson all too well. In response to the movement of the Bronze Soldier of Tallinn and some Soviet-era war graves, Estonia began to experience a rapidly scaling onslaught on its cyber systems [1]. This was an especially devastating attack because of Estonia's reliance on its information technology infrastructure as they are perhaps the most wired country on the planet. Government websites, banking systems, and even media outlets received roughly 100 times more traffic than they were used to, shutting down availability and having websites defaced [19]. Their dependence upon the cyber infrastructure to provide almost all services to their people could have easily resulted in total disarray. Fortunately for Estonia, their Computer Emergency Response Team was able to shore up defenses and eventually mitigate the attacks. "[Estonia] Defense Minister Jaak Aaviksoo said in Washington that U. S. officials told him Estonia coped better with the attack than the United State would have under similar circumstance" [16]. Although the culture of the U. S. is moving toward full cyber integration, it is not yet at the level present in Estonia so it is unclear if an attack of this scale would be as disruptive. However, the lessons learned are a wakeup call that improved defensive measures are necessary for the defense of computer networks.

The ability of cyber attacks to do more than disrupt networks and steal or destroy data came to light in late 2009. The advent of the Stuxnet worm is evidence as to how a designed attack can affect physical systems as it was able to destroy some of the centrifuges of Iran's nuclear program. Targeting the programmable logic controller (PLC)s, which manage industrial systems, the malware reprograms how hardware operates within the system

which can cause undesirable effects [35]. The sophistication of this level of attack shows how the integration of computer systems that control hardware leads to vulnerabilities that can be exploited from outside those physical systems. These Supervisory Control and Data Acquisition (SCADA) systems were once easily protected by locating in remote settings and great amounts of physical security; however, with the increasing use of Internet connections by local networks even one outside connection can be enough access for an attacker [28]. The threats against computer systems are no longer isolated to just the information present on the systems themselves, but now SCADA systems that run essential services for a nation's citizens are in danger. This underscores the need for reliable defense systems that enable the discovery and quick mitigation of threats.

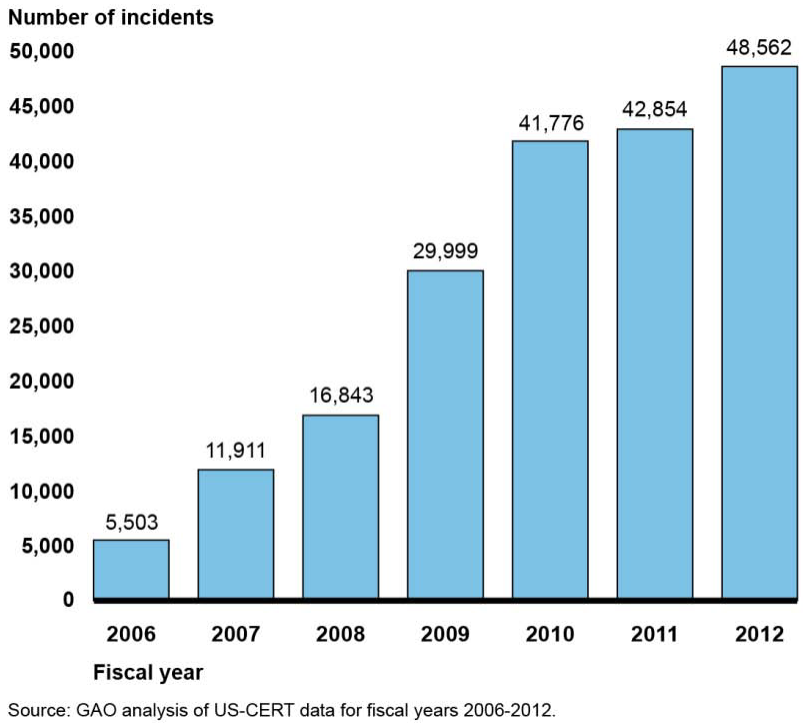


Figure 1.1: Number of incidents reported to US-CERT 2006-2012 [56]

The future seems clear enough, attacks on computer systems will only increase. The number of incidents reported to United States Computer Emergency Readiness Team (US-CERT) increases every year (Figure 1.1). A large portion of the reason for this is the ever expanding motivation for attacks. Motives range from the simple desire for profiteering (criminal) to furthering political goals through protest and reduction of government operations (hacktivism) and all the way up to nation sponsored attacks against other nations (warfare) [14]. Hacking has come a long way from the simple lone youth down in the basement causing mischief. Organized swaths of intelligent computer savvy attackers now exist and use their abilities to further the goals of their criminal organization or nation-state. Chris Inglis, National Security Agency deputy director, is quoted as saying, "If your security depends upon a static advantage and the static nature of compliance-based standards, your heart's going to be broken on a fairly regular basis" [17]. Not only has the game changed, but the landscape continues to evolve and the game being played becomes more complex. In fact, the continual battle for network dominance between attackers and defenders can been seen as precisely that, a game [46]. This view has led in recent years to the rise of game theory being applied to cyber operations and defense.

1.2 History of Game Theory

1.2.1 Beginnings of Game Theory.

Although roots of the ideas behind game theory exist as far back as the first centuries of A.D., the first truly known minimax mixed strategy solution (a solution where percentage choices of strategies by players lead to a predictable outcome) to a card game exists in a letter by Francis Waldegrave in 1713 [53]. He provides a solution to a two-person version of *le Her*, but does not extend the concept beyond this game. Some small developments continued to be made, but it was in 1920s that Émile Borel showed that solutions could be generalized for two-player games. John von Newmann expanded this concept by proving the minimax theorem and established game theory as a field of study in the 1940s [37].

Over the decade much work would be done in research of game theory concepts. The most influential breakthrough would happen at the hands of a troubled genius, John F. Nash, Jr.

1.2.2 Nash Equilibrium.

Nash provided game theorists with an insight that to this day permeates all branches of game theory research. He proved the existence of a strategic equilibrium for non-cooperative games that ended up being name after him, the Nash Equilibrium (NE). Simply stated, the NE is the set of strategy selections such that no player can increase their game payoff by changing strategy [36]. Thus it is not profitable to select a different strategy, because the reward for doing so is less than the reward for another strategy when taking into account the other player's action. The most impressive and useful concept to arise from this is that allowing mixed strategies (assigning a percentage choice to available player strategies) to be included means that at a minimum one NE exists in any finite game. This means that given full knowledge of the game's players, strategies and payoffs, any player can maximize their available utility.

1.2.3 Rational Players.

The players who use this knowledge to obtain the best outcome possible are described as being rational. A rational player has no reason to deviate from the NE because doing so would reduce the payoff that he or she hopes to obtain. This concept of a rational decision becomes very important in certain games. In the situation where I know the rules and strategies of the game, and I know you know the rules and strategies of the game, and you know that I know that you know, etc., then following the strategy sets outlined by the NE is the expected way to play the game. If a player acts irrationally, then his or her opponent will easily gain more utility and 'win' the game. However, what happens when all this information is not known to both sides can be very unpredictable. Actions by one player can be misinterpreted to be irrational because each side has differing views of the

same game. Thus, a player may seem to be irrational to the opposition, but may in fact be operating in a rational manner according to a different view of the game.

1.2.4 Hypergame Theory.

Hypergame theory is an extension of game theory that addresses the kind of games where misperception exists. The term hypergame was coined by P. G. Bennett in 1977 and seeks to explain how players in a game can have differing views of the conflict [11]. This advance in game theory shows how one player can believe decisions by the other to be irrational, but the opponent is actually making a rational decision based upon the perceived game setup. After its introduction it was used to model past military conflicts, where misperceptions existed, to show how the outcomes were achieved. Options selected by each side in the conflict are shown to be the rational choice by way of defining the game that each side perceived. In this manner, the hypergame analysis shows why unexpected results were obtained when one or both sides misconstrue the conflict. Hypergame analysis offers advantageous reasoning of strategy selection through situational awareness.

1.3 Application of Game Theory to Research

Although the name 'game theory' implies that the concepts expressed should apply to games, the ideas have been appropriated to many areas. At its heart, a game is just a conflict between two or more participants vying for the best possible outcome for themselves. The investigation of the competition between species has given rise to evolutionary game theory. Even evolutionary game theory itself has evolved to be applied to economics, sociology, and anthropology [7]. Application of game theory principles can be used wherever there is a contest over resources or something of worth. Even when there is cooperation between those involved in the conflict game theory can still model the outcome. At worst, game theory provides a rationale for the players to select strategies that offer avoidance of the least desirable results. This is where hypergame theory goes beyond standard game theory

by providing the means to outmaneuver an opponent, gain better utility, and still avoid unwanted results.

There have been studies applying game theory to terrorist activities of which certainly cyber attacks can be included. Terrorist activities are by nature covert and the conflict between them and governments can easily be seen as games of incomplete information [42]. Unfortunately much of the work that has been done in these studies fails to focus on incomplete and imperfect information games. This is an emerging field and the complexities involved in network security cannot be done by human analysis alone, computer assistance is required [33]. The inclusion of misperception and misinformation in this fight points to usefulness of hypergame theory because it provides strategy selection techniques that mitigate potential risks. Hypergame analysis is endowed with the tools that enable strategists to perform better than the worst case scenario and improve their position within the game [50]. The coupling of hypergame procedures with the speed and reliability of computer calculations can increase the ability of human administrators to protect networks.

It is clear that one of the biggest challenges in the cyber security game is establishing predictive analysis tools to assist against threats [56]. *This goal of this research is to provide a foundation for the use of hypergame theory for adaptable cyber defense systems able to adjust to the existence of misperceptions and misinformation*. A general game model envisioning a cyber attack and defense scenario that builds upon previous research efforts is expanded into a hypergame model. The model is used to demonstrate software written to give computer assistance for hypergame analysis. Application of the hypergame theory model is set against a game-based software simulated conflict by various types of attackers and defense techniques giving a quantifiable indication of hypergame analysis strategy selection. The game-based software simulation is achieved by running iterations of the game model where adaptability of strategy selection is the key focus. In the end,

this research gives initial evidence that using hypergame analysis allows network defense to adapt to the type of attacker faced in the fight for security.

1.4 Document Overview

In chapter 2, Literature Review of Game Theory Research, various efforts in the arena of cyber warfare analysis with game theoretic models are assessed. An in depth discussion on game theory models is provided with an emphasis on design techniques. Previous research in the application of game theory to cyber security is reviewed as evidence of its usefulness in this arena. Hypergame theory extends basic game theory principles and the advances it provides over standard game theory are discussed. Several forms of hypergame theory are reviewed showing the robustness of its relevance for conflict analysis. Finally, the plausibility of applying hypergame theory to cyber warfare is examined.

Chapter 3, Hypergame Model and Software Design Methodology, presents the reasoning behind the game architecture, software development, and the experimental plan. Various features of game theory model design are discussed and decisions on their use are made. The model, using the features selected, is formulated with a basis laid upon previous research efforts. Software design decisions, that result in the creation of the software used to instantiate the model and provide experimental output, are analyzed.

Chapter 4, HAT Software Experiment Design Methodology, describes the experiments that are performed with the game model and custom software. Pre-experiment tests are formulated that test the game model requirements and ensure that the software meets them. An experimental plan is devised that incorporates the game model and examines various game setups that may be encountered in cyber defense scenarios.

Chapter 5, HAT Software Experiment Results, reviews the outcome of the experiments outlined in the experimental plan. Pre-experimental test results, designed to ensure that the created software is functioning as required, are outlined first. Basic tests are next and provide a backdrop of base values upon which further tests are judged. Iterative

testing follows showing how game outcomes change after playing several in a row. Lastly, experiments are run to examine how updated values affect the player's strategy choices during game execution. The results of all experiments are then discussed to give credence to the applicability of hypergame theory in the arena of cyber warfare.

Chapter 6, Conclusions, gives discussion based on the experiments and what the future will hold for this area of research. The findings provided by the experiments are examined under the lens of their usefulness for cyber security. The effectiveness of the results and recommendations on their application are provided. Suggestions for future research using hypergame theory are prescribed. Also, ideas on the future use and expansion of the software used to perform the experiments herein, including any current known issues, is offered.

1.5 Summary

The state of cyber operations is discussed with emphasis on recent events and the possibilities of future catastrophes. The need for security of networks is shown to lie in the hands of trustworthy cyber defense mechanisms. The history of game theory shows that it is a reasonable approach to modeling decision making and determining the outcomes of conflicts. Game theory has been used to understand various applications in areas as diverse as military endeavors, economics and biology. Hypergame theory is a branch of game theory that focuses on conflict resolution when there are misperceptions by one or more players in a game. Most recently, game theory has begun to be applied to the area of cyber warfare. This new area of research shows promise in its ability to analyze situations that may allow for better cyber defenses. The application of game theory, and more specifically hypergame theory, to cyber defense of computer networks is the focus of this research.

II. Literature Review of Game Theory Research

THERE is a large body of work of studies involving game theory. Specific interest in the area of game theoretic models applied to cyber security is covered within this chapter. Additionally the focus of the research involves hypergame theory so much research in this area is examined. Other areas of game theory that involve cyber security applications are touched on.

2.1 Game Theory

Game theory is used to model diverse areas such as economics, natural selection, battles in past wars, and many other types of conflict [37]. The main influence behind the creation of game theory is the resolution of such competitions. Game theory models have many properties associated with them that influence the outcome and how game analysis proceeds. Cooperative games can exist where there is communication between players, but more often games are seen as non-cooperative where the players do not attempt to give up any information to their rivals. Simultaneous games are when players make decisions at essentially the same time and do not know of the opponent's move in advance (i.e. rock, paper, scissors). Conversely sequential games are when the opponent's move is known before a decision is made (i.e. betting in poker or chess). Perfect information versus imperfect information is when all previous moves are known to all players instead of some being hidden. These previous two concepts are often confused with complete information and incomplete information, which are actually intended to be the knowledge of all player's strategies and payoffs. The payoffs, also known as utilities, also have a common property of zero sum or non-zero sum. When the utility values for each player are directly opposite, it is considered a zero sum and a clear winner can be determined. When utility values are not opposite, non-zero sum, all players involved can gain or lose and it may be difficult to

ascertain the winner. These properties are just a few of the main defined for game theory, but a large portion of a game theory model definition comes from its form.

		Prisoner B: Stay Quiet	Prisoner B: Confess
Prisoner A	Stay Quiet	-1, -1	-5, 0
Prisoner A	Confess	0, -5	-3, -3

Figure 2.1: Prisoner's Dilemma : A strategic game example

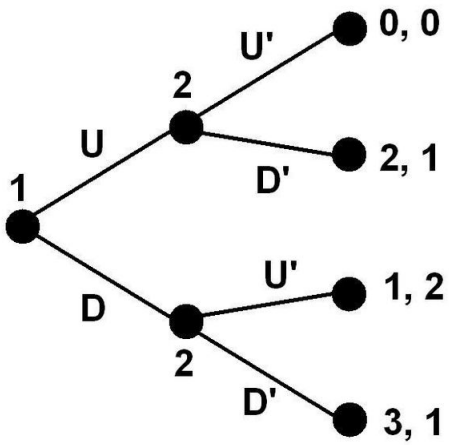


Figure 2.2: An extensive form game example [10]

The majority of game theory models are identified as either strategic, used to represent a simultaneous game, or extensive games, more often used to represent a sequential game even though it can represent simultaneous as well. Strategic games have utilities that are determined by which strategy is selected by each player. When a player's strategy

is selected all strategies are available to choose from and therefore the game is represented in a grid or matrix format (Figure 2.1). Extensive games' outcomes are instead represented as a tree structure where the initial player is at the top with branches leading off for each strategy available, and then the next player has strategies available from each branch creating branches for each of its strategies available (Figure 2.2). All strategies that a player can use, may or may not be available at a particular node of the tree dependent upon the previous player's selected strategy.

		Player 2	
		Swerve	Straight
Player 1	Swerve	0, 0	-1 , 1
	Straight	1 , -1	-10, -10

Figure 2.3: The game of chicken

The simplest way to analyze outcomes of games is to assign values to each player's utility values that are static and do not depend upon any variation. These static utility values allow the math to be straightforward and provide quick discovery of Nash equilibriums. The formal equation is:

$$\forall i, x_i \in S_i : f_i(x_i^*, x_{-i}^*) \geq f_i(x_i, x_{-i}^*) \tag{2.1}$$

Where i is the specific player, S_i is the player's strategy set, and $f_i(x)$ is the payoff function. Let x_i be the strategy profile of i and x_{-i} be the strategy profile of all other players. A weak NE means that for at least one player another strategy set leads to the same result. A strict NE is when there is only one strategy set to reach the result and no player can change their strategy, otherwise the result also changes. The pure strategy Nash equilibrium (PSNE)

in a strategic game, for example, has a relatively uncomplicated procedure for discovering whether or not one exists. Each strategy of a player is examined to see the maximum utility that is obtained given the other player's strategy choice. When the players' strategic choices coincide there exists a PSNE at those strategy selections. The game of chicken, see Figure 2.3, illustrates this concept, where the underlined utility values are the best choice for a player given that the other player has selected the corresponding strategy. Given that the other player chooses *swerve* there is no incentive to *swerve* as well, so the strategy choice is *straight*, $1 > 0$. However, when the other player chooses *straight*, the strategy choice most beneficial is *swerve*, $-1 > -10$. Therefore, in the game of chicken there are two PSNE, showing that with static utility values the operation can be straightforward. When no PSNE exist the calculation of mixed strategy Nash equilibrium (MSNE) is more complex, but with matrix operations even somewhat complex games can have at least one Nash Equilibrium (NE) found within a reasonable amount of time.

	Hawk	Dove
Hawk	(V/2) - C, (V/2) - C	V, 0
Dove	V, 0	V/2, V/2

Figure 2.4: Hawk-Dove game, illustrating the use of functions for utility values

Static utility for games are used as a way of simplifying a game model for analysis. Therefore, the replacement of these values with functions makes the analysis a complicated endeavor. An example of this is the Hawk-Dove game where the addition of variation in

the utility that a player will receive upon selection in most cases alters the equilibriums and thus the decision of the best strategy choice, see Figure 2.4. Both the hawk and the dove require the resource (V), food, but a fight over the resource by two hawks results in a cost (C), injury. In the analysis of NE it can be shown that if $V/2C > 0$ then the hawk strategy strictly dominates and both players will select it. However, when the opposite is true, $V/2 - C < 0$, the equilibriums are drastically different. There are two PSNE, hawk-dove and dove-hawk, but there is also the addition of the MSNE where each player chooses hawk $V/2C$ percent of the time and dove $1 - V/2C$ percent [47]. The advantage of variable utility modeling over static utilities is that it can describe a real world situation in more detail than the static values. As the value of a variable changes from game to game (i.e. food valuation could depend upon supply) it has impact on strategy selection.

A game worthy of note that encompasses some of the ideas discussed is that of cops and robbers, more formally known as pursuit-evasion games. The game consists of cop and robber teams where the robber attempts to evade the cop and win by never being caught, while the cop wins when the robber is captured. Usually this game is presented on a graph, G, where there are vertices which the players occupy and move between and if they ever occupy the same vertex the cop wins. This game is often modeled using graph theory, like in [5] to provide winning strategy analysis. However, as shown by Kahn in [30], a game theoretic model can be used to describe the competition. The game is formally defined by using the extensive form, zero-sum and perfect-information. It is shown how a non-game theory model of a game can be transformed into a game theoretic model. Also of interest here is that the idea of the pursuit-evasion game is applicable to real world situations. It is not unlike the detection of invaders present in network nodes by a system's defenses.

2.2 Game Theory Applied to Cyber Security

The increasing incidents of network intrusion are influencing the rapid evolution of both malware and defense techniques. As noted in [46] this coevolution is like a game

between the malware authors and security analysts, each attempting to outperform the other (thus winning the game). Logically, this leads to the interesting point of view that cyber threat attacks and the defenses that are put in place to deflect them can be modeled as a game between two players. Game theory is the tool that enables a mathematical understanding of the strategy choices that are selected by the attackers and defenders in this competition. A game that models the struggle between these two adversaries can lend insight into the best strategy choices for each side. It can even be stated that the side that is most proficient at understanding the game will gain the upper hand and emerge as the victor.

In general, game theory research conducted to analyze these cyber security situations focuses on the use of defense techniques to monitor the network for anomalies and mitigate suspected intrusions. The use of firewalls, passwords, access control lists and other outer defenses are generally regarded as static defense. These defense mechanisms are not usually updated dynamically and therefore do no lend themselves especially well to analysis by the use of game theory principles. Detection of anomalies within the network, however, requires response to findings and the implementation of changes to defend the network. These changes, being required at any time in order to combat an emerging threat, are an excellent reason to consider these operations as candidates for analysis. The detection of anomalies within the network is put in place to be a second line of defense to detect intrusions and consequently generate the appropriate responses to the threats that appear [38]. Malware and attackers that intrude into the system do not wish to be found, while the defenders are actively searching for these invasions. It is not generally considered enough to just detect the attack, but it must also be mitigated. Traditional intrusion detection schemes are not evolved enough for complex detection and are lacking in effective response strategies; therefore, efficient detection policies and appropriate response policies need to be logically formalized [6]. Game theoretic analysis is especially well suited for

formulating these responses because a decision on the best strategy to take follows from that analysis.

An overview of various approaches to the use of game theory reveals that although there are many differences in design of the models they all embody a central theme. Game theory is used to select those security measures providing the lowest incurred cost balanced with the highest level of security that is able to be achieved [44]. This principle is the main reason that game theoretic models should be used to influence the decisions that are made when deciding strategies for network defense. The more open a system is the more it becomes a liability, while the more security measures imposed on a system reduces its ability to be used by those that need it. When authorized users are unable to use the system because the security levels hamper their ability to access the network, it is tantamount to allowing the attackers victory in the game. Administrators of the system must realize that when a legitimate user is trying to access the system and cannot, eventually giving up, the cyber-battle has already been lost [6]. However, there are times when information is at risk and must be protected that it is crucial to limit access, even for the legitimate users. During these times, those that necessitate analysis of the trade-offs, game theory modeling can provide a strategy in which the best balance between security and availability is achievable.

A human administrator can make the decisions necessary that provide the network defensive capabilities which combat threats and allow users sufficient access to the system. Computers, on the other hand, can analyze all of the combinations and permutations of acquired data to find exceptions in general rules. This is in direct contrast to humans who are very prone to overlooking possibilities [41]. This is not to say that computers should make the decisions without human interaction, but instead that many factors should be included in the reasoning behind the resulting conclusions made by the computer's analysis of the intrusions. Most importantly, in any computer analysis, the game model must accurately depict the situation that is being encountered in order to provide an accurate

response; otherwise, an incorrect strategy is easily selected. A model that is able to at least come close to the representation of the condition of the network will provide invaluable information leading to a preferred strategy. This is because the computer analysis of game theory has the capability of examining hundreds of thousands of possible scenarios before selecting the best action. Computer analysis sophisticates the decision process of the network administrator to a large extent [44]. The modeling of the game being played is the most important aspect of using game theory analysis to implement cyber security. To this end, much of the research conducted in game theory's use for cyber operations focuses on the creation of an applicable model.

The difficulty in creating the appropriate model that represents the attack and defense of a network is that the number of variables involved can be immense. Thus, most models simplify the situation so the ideas they focus on can be examined within a reasonable timeframe and without a multitude of variables affecting the results. This does not mean that valuable information cannot be gleaned from these examinations, but only that models will fall short of how many real world situations occur. The simplifications lead to the majority of models describing two player games, the attacker as one player and the defender as the other. This is a reasonable expectation because the more players (P) and number of strategies per player (S) involved in game theory models requires increasing the number of game results (r) as shown here in Equation (2.2). The addition of just one player to a game

$$r = P_1S * P_2S * ... * P_nS \tag{2.2}$$

with two strategies for that player results in doubling the number of possible outcomes. Generally the number of strategies in the model is also kept low because each strategy added for a player increases the number of outcomes by the total number of other players' strategies. Therefore, many models tend to be two-player with two strategies each like in [15] where the attacker can attack or not attack and the defender can defend or not

defend. This seemingly over-simplification allows an examination of other variables, such as multiple nodes being attacked or utility values that are equations instead of set values.

Although some models researched do use the extensive model like [43] [8], and some even examined both extensive and strategic [23] [9], the majority of those examined use the strategic model as their backbone. The simplicity of strategic models allows for a more straightforward analysis, but this is not the only reason for the preference over extensive models. When considering cyber security one must understand that in most cases when an attack is being precipitated against a network node that is not defended at the time of the attack it can be difficult to mitigate it. Therefore, an extensive game where moves are made one at a time does not adequately describe the situation. In the remainder of the research examine herein, the focus is set on the models that use the strategic approach.

The design of a strategic model consists of two parts, strategy selection and utility outcome assignment. A typical model views the game as a strategy decision between two players where each player's strategy is listed on one side of the grid (Figure 2.1). Once the strategies are determined, the resulting cells of the framework can be populated with the utility, or value, that each player will receive when the two outlying strategies have been used. Commonly when this setup is used in cyber defense analysis the two players are typically the attacker and defender of a network. A simple design will consist of two strategies each, attacker can attack or not attack, while the defender can defend or not defend. A variation of this was used in [9] where the defender actually had a high level of defense or low level instead of none at all. Also in this variation the game was designed with static outcome values and there are no variables to change the values over time or instance. Static utilities provide an easy analysis of game results with rational players because the NE can easily be found, incidentally the simplicity of the model also impacts the ease of analysis. Bandyopadhyay and Sebalia show this by easily determining the

strictly dominated strategy and the NE of their model. However, as the complexity of a model is increased, so does the complexity of NE discovery.

Table 2.1: Chen and Lenutre *Variables*

Variable Symbol	Meaning
a	Detection Rate
b	False Alarm Rate
C_a	Cost of Attack
C_f	Cost of False Alarm
C_m	Cost of Monitoring
W_i	Value of Target

The variable utility approach was selected by Chen and Leneutre [15] in their examination of intrusion detection for heterogeneous networks. Their model retains the simplicity of using a two-player game, attacker and defender, and two strategies each, attack/not attack and defend/not defend, but the utility values are based upon variables. The variables they selected to determine the utilities are shown in Table 2.1. The assignment of functions to the game grid using these values allows the authors to show how in a real world situation a rational attacker will only have an inclination to attack the higher value targets. The subsequent analysis of the NE proves this by showing that whenever the target is not in the rational target set, the MSNE percentage for attack or defend drops to zero. Under the assumption that both the attacker and defender have sufficient resources to engage each other over the entire target set the NE becomes:

$$\begin{cases} U_A(\mathbf{p}^*, \mathbf{q}^*) = 0 \\ U_D(\mathbf{p}^*, \mathbf{q}^*) = -\frac{bC_f + C_m}{2a + bC_f} \sum_{j=1}^{N} W_j \end{cases} \tag{2.3}$$

The conclusion of this analysis is that increasing attack or monitoring does not change the NE and the attacker does not gain from a decrease in the attack cost. In order to decrease costs the defender must increase performance by improving detection rate and/or decreasing false alarms. Improvement can also be implemented by a reduction of the costs involved. These factors show that the application of formulae to utility values can provide valuable insight on how to prepare for the game and improve a player's performance.

2.3 Hypergame Theory

The critical factor that standard game theory analysis is missing, however, is the ability to see past the simple game and model the fact that one player may have an advantage over the other. The previous research discussion does not focus on the fact that not all information may be present to all parties involved in the game. Indeed, there can be many levels to a game that involve a kind of meta-strategy that requires subterfuge allowing one player to disrupt the Observe-Orient-Decide-Act (OODA) loop of the other [29]. The concept of not allowing the other player to fully grasp the strategies employed by the other is what has given rise to the study of the meta-game, also known as hypergame theory. Hypergame theory examines the fact that a player may have the tactical advantage of knowing that there is in actuality a larger game with more strategies that can be selected, there is a reasonable expectation that their opponent will choose a specific strategy instead of others, or that the opponent expects a certain strategy to be chosen.

Hypergame analysis extends game theory by providing for the larger game that is really being played whether or not both players are aware of it. A different game model can represent each player's view of the conflict, but more often than not these models will overlap where their knowledge is common. Figuring out what strategy a player will use is dependent upon not only his or her observation of the game, but also how that player believes the opposition is viewing the game. This creates many different game models that are examined for the solution to be obtained. The goal of hypergame analysis is to provide

insight into real world situations that are often more complex than a game where the choices of strategy present themselves as obvious. The beginnings of hypergame analysis began by the examination of completed situations where perception and information differences existed between the parties involved. It is much easier in hindsight, where all choices and outcomes have been determined, to achieve understanding of these kinds of situation.

It comes as no surprise that the initial foray into hypergame theory examines past military conflicts, which are prone to having misperceptions and missing information in the process of their unfolding. Analysis of past conflicts also lends itself to ease of understanding since the fog of war has cleared and the outcome has been determined. In this regard, Bennett and Dando [12] examined the fall of France due to the attack of German forces in 1940. They began by asking the question of why were the Allies not prepared for such a bold attack by the Germans. Their determination was that the Allies had discounted the option of an attack through the Ardennes, which resulted in the Allies misperception of the situation. The Germans, on the other hand, were taking a broader view and perceived a meta-game that encompassed the strategy option that the allied forces had discounted. Thus by being able to determine that the French would discount such an action, the German's were able to turn what was a rational decision by the Allies against them (Figure 2.5). The French player, only able to "see" the game on the left follows the

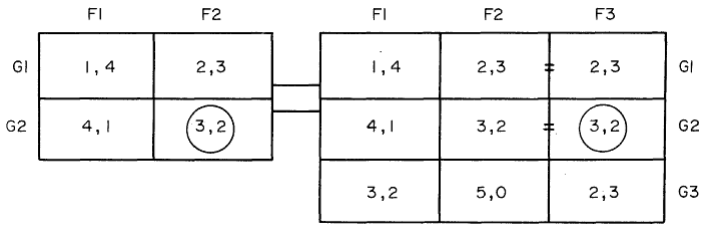


Figure 2.5: Bennett and Dando's Hypergame representation of the fall of France [12]

NE and determines that F2 is the acceptable choice. The German player, knowing that this

is the French player's rationale, realizes this and selects strategy G3 from the game on the right providing the best possible outcome for Germany. The outcome of such an analysis can show that each player can make what appears to be a rational decision on strategy choice, but that a misperception by a player can prove that choice to be a poor decision.

Hipel et al. [24] take another approach in their examination of the Falkland/Malvinas conflict between Britain and Argentina in 1982. They provide a hypergame analysis of the evolution of the conflict that shows that misperceptions by both sides can lead to an outcome that is unexpected by both players. Because each side makes choices without a view of the full hypergame neither player gains the benefit they were hoping to achieve through their strategy choice. The most interesting point of this analysis is that the game is viewed over time where choices made affect the next iteration of the game, providing different options and outcomes. Game changes based upon these changes are incorporated into the evolution of the game grid and information is gleaned at each step by the players. Eventually, the hypergame is reduced to a simple game because misperceptions fall away as both sides become aware of the remaining options and outcomes. However, this analysis and modeling, as well as that in čitehgtFall is relatively straight forward when one examines a conflict where all decisions and outcomes are known a priori. In order for hypergame theory to be useful when examining future conflicts, models which can represent these situations without previous knowledge require discussion.

The hypergame model attempts to create a vision of the players' understanding about the nature of the game being played. The model, once set forth, is then used to reason about the strategy selection of those players involved. Wang et al. [54] are at the forefront of the efforts to quantify how the hypergame model is designed. In their modeling the hypergame is divided into different levels of game perceptions. At the lowest (0 level) there exists the basic game where there are no misperceptions. The first-level hypergame introduces the first misperceptions, where the players have different games, but are unaware of the

other players' games (much like the game of the Falkland/Malvinas conflict from [24]). A second-level hypergame is where at least one player knows that there are different games being played (the Fall of France from [12] for instance). They also describe a third-level in which at least one player knows that other players know there are different games being played. The authors admit that a *nth* level hypergame can be described, but this would most likely be an over-complication of the model and unnecessary for analysis. The analysis of this model technique does not ensure that the outcomes have the same meaning for each player as a basic game theory analysis would. Instead the equilibrium can be transient as iterations of the game are played and the perceptions of the players change.

This modeling technique is further examined in [55] where the stability analysis of the hypergame is fully defined for *n*-players. Through these definitions the authors describe the various ways in which stability, a game equilibrium, is reached and why these outcomes exist. Nash stability exists when players make decisions that are rational (R) based on the outcome that suits the player. However, a player's perceptions can affect the true outcome and Nash stability is more likely to be achieved when misperceptions do not exist. There are also general metarational (GMR) and symmetric metarational (SMR) game outcomes for GMR where other players have a joint response for player *i* and that player can achieve no better outcome than the original and where SMR there exists one jointly sequential strategy selection to pin player *i* to the same result. Finally an outcome is considered to be sequential stable (FHQ) when there is a response to a particular player's strategy that puts that player in no better position, but also the responding player cannot be put in a worse position. A graphic representation of these definitions Figure 2.6 shows their relationships. The most significant finding in the author's discussion is that an FHQ outcome exists in every level of the hypergame, which also means that a GMR outcome must also exist.

In [52], Dr. Russell Vane III offers a different approach to hypergame modeling by providing the incorporation of a player's beliefs on opponent's possible actions. He

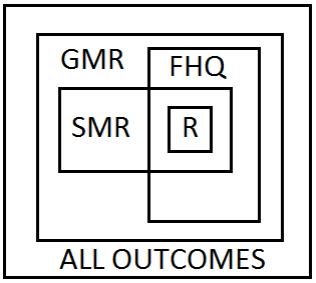


Figure 2.6: Individual Stability Concepts in n-person Hypergames [55]

also provides a graphic representation of the hypergame that is reminiscent of the normal strategic form used in standard game theory analysis. The new model is referred to as Hypergame Normal Form (HNF), see Figure 2.7. The full game is the familiar grid form with row and column strategies labeled and the utility values (u_{11} - u_{nm}) in the cells cross referenced from the strategies. The additional sections are the hypergame situational information. The Row Mixed Strategy (RMS)s are *hyperstrategies* gleaned from what the row player believes about the game being played by the column player. They are called hyperstrategies because they do not encompass the view of the full game, except for R_0 which is the full game NE. The Column Mixed Strategy (CMS)s are row's belief about the mixed strategy percentages that column will play when selecting a strategy. C_0 is column's NE view of the full game. When a CMS cell contains a 0, this is an indication that there is a sub-game that column is believed to be playing where the corresponding strategies are either unknown to column or discounted as not worthwhile. The final section is the belief-contexts, which correspond to the percentage of which row believes that the adjacent CMS will be played by column. Since they are percentages the belief-contexts will sum up to one, with the sum of values P_1 through P_{K-1} being at most 1 and the leftover constitutes the NE belief-context. Filling in the HNF with the values associated with the game provides the avenue for the hypergame analysis.

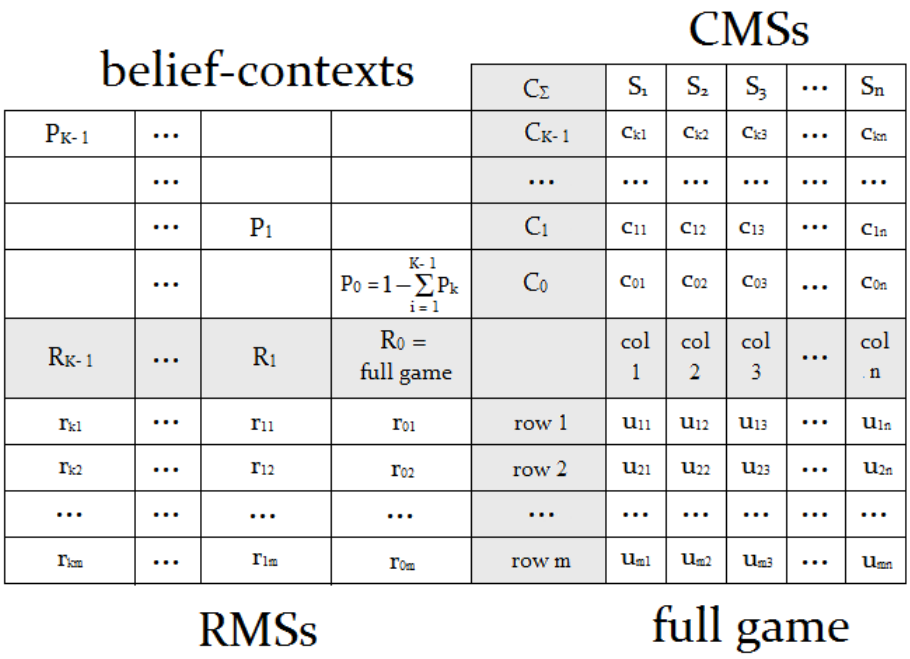


Figure 2.7: The Hypergame Normal Form [52]

Determining the utility values allows for a NE for the full game to be calculated which provides the input into R_0 and C_0. CMSs are then entered in the section above the full game. A CMS can be determined manually, i.e. knowing a player's preference for selecting rock in a game of rock, paper, scissors, or a NE for the column player can be used from the analysis of the sub-game that the column player is believed to be using. Each CMS is assigned a belief-context value which serves to weight the row player's belief that column will choose that CMS. These values are used to calculate C_Σ, the aggregate amount which directly affects the expected utility that row hopes to achieve. Row's hyperstrategies are then input into the RMS section. Expected utility values for each strategy set listed in the RMS section are calculated for the full game NE CMS, C_0, and the aggregate belief CMS, C_Σ. These values determine the effectiveness for which an RMS hyperstrategy is a practical selection for row to apply.

RMS effectiveness is categorized into three levels of usefulness: fully effective, partially effective, and ineffective (Figure 2.8). A fully effective strategy set will provide at worst case the same expected utility that row's R_0 strategy set achieves for C_0, but has a greater expected utility at C_Σ. Thus, given that row is viewing the game correctly a fully effective RMS is always a good choice. Partially effective strategy sets also provide a greater expected utility at C_Σ than R_0, but have a lower utility expectation at C_0. Given row's information, a partially effective RMS could provide a good outcome, but it is not always assured. The ineffective strategy set provides for no increase in utility and at best can only get to that expected by the NE, so there is no reason to select it. It is reasonable to assume that fully effective strategies sets should always be used, but that doesn't mean there isn't some inherent risks involved because the utility values are only expected and not foolproof. Worst case scenarios can also be included in this determination to help mitigate risk.

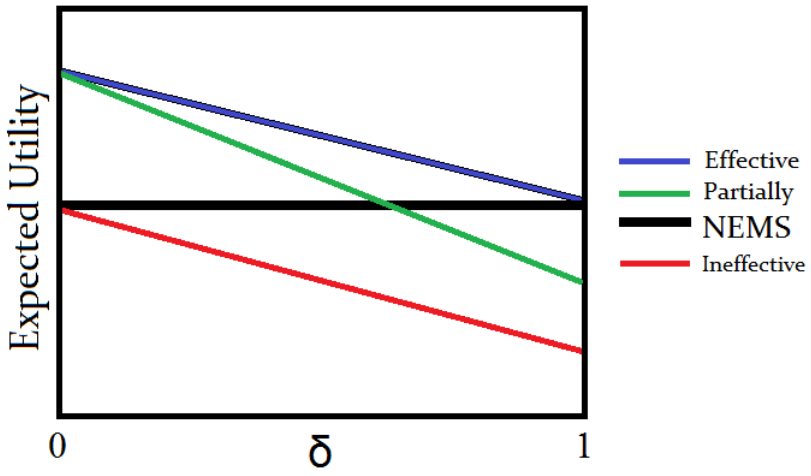


Figure 2.8: Effectiveness of hyperstrategies in HNF [52]

Risk assessment is built into the hypergame analysis through a method termed quantified outguessing. This method introduces the fear of the player that he or she

will be out maneuvered and the worst case utility will be the end result. Three types of hyperstrategy sets are described for this analysis: Modeling Opponent (MO), Pick Subgame (PS), and Weighted Subgame (WS). MO is simply selecting the strategy for row that will provide the highest utility given all of row's strategy selections and when considering the belief of how column views the game. In contrast, the PS strategy set consists of the NE for the same game view that was considered for MO. WS uses the PS strategy values multiplied by the belief-context percentage for that CMS and adds the R_0 multiplied by the belief-context for C_0, which results in a hybrid strategy set between PS and the MSNE for the full game. Each hyperstrategy is then assessed against the full game to quantify the worst case utility (G), or the utility value expected when column selects the correct counter strategy. The Expected Utility (EU) and G, once determined, allow the Hypergame Expected Utility (HEU) to be calculated by also considering g, the percentage chance row feels that he or she will be outguessed (Equation (2.4)). As can be seen the distance between EU and G has a quantifiable affect on the value of HEU for the

$$HEU(hs) = EU(hs) - (EU(hs) - G(hs)) * g \tag{2.4}$$

hyperstrategy (hs). As the fear of being outguessed increases the ability of any hyperstrategy to provide better utility that the MSNE solution to the full game decreases (Figure 2.9). The lower the fear of being outguessed MO is the best selection, but as that fear increases eventually PS dominates for a short period until the Crossover Point where MSNE for the full game is dominant (note that WS is always dominated and does not provide a suitable choice). Therefore, with good information on the intent of the adversary, hyperstrategy selection that provides better utility than standard game analysis is achievable.

Some further research in the use of the HNF have been performed after its creation. The ideas about hypergame analysis are expanded upon in [51] by Vane himself. The presence of luck and robustness of strategy plan are examined, but for the most part it

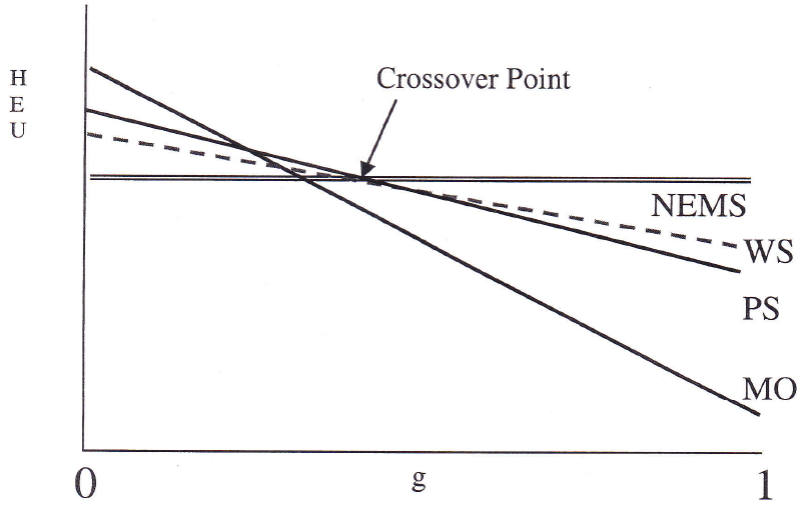


Figure 2.9: HEU value changes dependent on the value of *g* [52]

provides further evidence of the usefulness of hypergame analysis for the strategy selection process. A real world example of how to use the HNF is provided in [50], which examines a terrorist attack. The analysis entails applying belief-context values to expected types of attackers so that a strategic decision can be made to best protect first responders. This reiterates the idea that uncertainty exists and needs to be assessed when planning. Perhaps the most interesting application of the HNF is in čitehgtUsing where it is used to model the fall of France in 1940. The model is compared to the dual standard game model presented in [12] and to a preference vector model like that in [24]. Specifically it is outlined that using the HNF approach allows all information to continue to be presented and not removed from the model. Even when a strategy is completely discounted by applying a percentage chance of use of zero, it remains in the total game MSNE analysis and is not completely removed from the model. These research efforts show a glimpse of the usefulness of the HNF.

2.4 Hypergame Theory Applied to Cyber Warfare

Unfortunately very little in the way of hypergame theory application has been done in the arena of cyber warfare. Although much recent work has used game theoretical applications in an attempt to model network security, the use of hypergame theory is not considered on the same level. Hints to its effectiveness have been suggested. Certainly, a computer-based tool can provide easy access to information [50]. It is easily conceivable that an cyber defense system can be infused with a hypergame model in order to influence decisions on network defense. Even Vane, although not in a sense directed at defense agents, suggests that hypergame theory should be used for decision theory and game theory agents [49]. Evidence tends to point out that, given credible information, hypergame theory can do no worse than game theory models, but has the potential to outdo them. Providing a computer-based tool the ability to analyze a situation with a hypergame model creates a quick and efficient technique for strategy selection.

There has been at least a small amount of work performed in the use of hypergame theory applied to cyber warfare. House and Cybenko in their paper [25] use hypergame theory to model a generic cyber attack where the defender can choose a specific subgame or the full game dependent upon the experience level of the current defending administrator. Their model is designed using Dr. Vane's HNF, static utility values, and with the row player as the attacker. Using learning models it is shown that row can eventually determine the percentages, which in the HNF are the belief-contexts, that each subgame is being used. In simulations the results indicate that within 3000 to 5000 iterations these percentages are known within ± 5% of the actual usage percentages. These results show that there is at least some measure of confidence that using a learning strategy that one player can begin to understand the strategy selection of the other. This represents the ability to understand one's adversary in order to select one's own strategy to provide the best possibility at maximum utility.

The research does not conclude with these findings. Instead, the possibility of the defender (or column player in this case) obfuscating the learning ability of the attacker is examined. It is shown that there does exist at least a limited ability for the column player, using an obfuscation MSNE, to disrupt the ability for row to learn column's true usage percentages. This obfuscation MSNE is created by remodeling the utility values that were present in the initial row learning experiments. The result is that row will have a more difficult process by which to determine the strategy that has been played against it in each iteration. This approach is somewhat manufactured in order to provide column with the ability to select strategies that may be misinterpreted. However, the fact that payoffs and subgame definitions are the heart of any hypergame scenario is presented as the reason for this approach.

The results of this research effort are of most interest due to the usage of hypergame theory in the attempt to model a cyber warfare situation more than the actual content of the findings. House and Cybenko do not use the full HNF model, choosing to concentrate on learning the nature of the game through play instead of attempting to outguess the opponent as discussed in [52]. Despite this fact, the usage of the HNF to model the experiments shows the viability of the model to be incorporated into network defense initiatives. More avenues of the use of hypergame theory and specifically of models using the HNF are required to prove the effectiveness of the approach. The exploration of how hypergame theory can affect cyber defense and offense is only in its infancy.

2.5 Summary

Game theory has been used to model many things in the world. Although the roots of the theory lay in the 16th century, it wasn't until Jon von Neumann founded the field in 1928 with his proof of the minimax theorem of the zero-sum games with perfect information. From these humble beginnings it grew to encompass economics, business, biology, and many more areas. In the 1950s John Nash expanded the field to encompass

strategy selection of players for more generalized games with his description of strategy equilibriums (known as the NE). Games have many aspects to them, but each must have a type of form. The most common of these forms are extensive and strategic.

In recent years the concepts that embody game theory have begun to be applied in the area of cyber security. This is mostly due to the view that attackers and defenders of a network are in a sense playing a game, with each trying to gain the upper hand upon the other in the struggle to control the network. In order to keep analysis simple, game model designs are often basic, using two players with two strategies each and static utility values. Real world situations are usually more complex than this and Chen and Lenutre [15] use a more extensive model including variables to more adequately reproduce conditions that would be found in a real network defense system. Their approach still only examines a simple attack/not attack and defend/not defend strategy selection.

Hypergame theory expands upon the ideas that are the foundation of game theory. It models complex situations in which at least one party is unaware that there are different games being played by the players involved in the game. Work in this area has grown around the analysis of military conflicts where there exists much misinformation and deception. Models have been examined that allow for the analysis of hypergame conflicts in the effort to find why a situation was reached, or how a best outcome can be gained. Dr. Vane's HNF is introduced as not only a method for hypergame analysis, but also as a strategy selection tool that considers the risk of being outguessed. Research using the HNF is examined to show that it can be used to model previous work as well as real world situations. The benefits of hypergame theory over game theory is shown to be the possibility of better utility than that available by strictly following the MSNE and why this is achievable.

Only a small amount of research is currently being performed with the application of hypergame theory for cyber warfare. Although efforts have been few, it has been

recommended several times that more attempts using hypergame theory applied to cyber warfare should be made. House and Cybenko have provided some insight as to how this theory can give credence to the approach, even though their research was limited in its application. Through their efforts, the use of the HNF is shown to be a feasible modeling technique for the performance of experiments in this arena. Further research into the use of hypergame theory for cyber defense and offense needs to be accomplished because little has yet been done.

III. Hypergame Model and Software Design Methodology

Design of a hypergame theory model that shows the viability for the application of the theory for cyber security is a vital portion of this endeavor. The mix of game properties selected determines the structure of the model. Various software packages are considered as the component that allows for model evaluation and simulation. Software must meet the requirements ascribed by the model to be a viable option.

3.1 Game Theory Method Selection

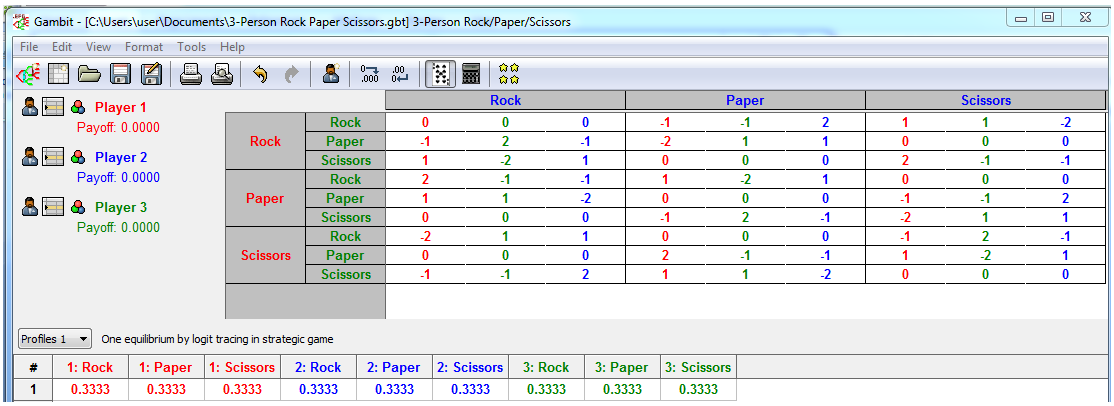
Game theory has been shown to be effective in its application to various kinds of real world situations from economics to biology [37]. The situation that arises in a battle over network control by the owners and those seeking to abuse it lends itself to a game perspective. This makes the decision to apply a game theoretic approach to an analysis of this confrontation an interesting endeavor, but there remains the question of how to proceed in modeling the conflict. There are many ways within game theory to model a game made up of players and strategies and surely there is not necessarily one "best" way to accomplish this goal. A design that incorporates these ideas in a way that is usable and practical is an important feature when a study is undertaken. Thus, the varying attributes of game theory are examined and the reasons for the selections used are discussed.

The most influential aspect of any game theory design is its form, which represents the players and strategies that are used to depict the conflict. As previously discussed, the most common of these forms are the extensive and strategic structures. Extensive games are modeled in a tree design and the more strategies available to players, the more expansive this representation becomes, see 2.1. A complex extensive game design can easily get out of control in terms of its size and ease of being understood. The strategic form offers a design that accommodates the addition of strategies more readily by way of its matrix layout. The

determination of utility outcomes is quickly available by the cross referencing of player's selected strategies. Extensive form games are also considered best for application to turn-based games like chess, while strategic form is better suited for simultaneous move games.

Although cyber security could be considered in a turn-based fashion because moves can be influenced by preceding events, the situation lends itself more to a simultaneous environment due to defensive measures already being in place as an attack occurs. Strategic form also lends itself to game play in which two players are involved, keeping the design in an easy to read matrix and this is the consideration herein. However, the strategic form does not limit the players to two. It can be expanded in a manner that is still understandable, as evidenced by a multi-player output used by the game analysis software Gambit, see Figure 3.1 [34]; however the complexity expands as the number of players increases. In a real world scenario it may be that there is more than a single party attacking a network, but it is often difficult to delineate attackers separately so considering all attacks as a single entity is not without merit. Also in this situation the defender is considered as a single entity, the network owner. Although limiting the players in the conflict is a simplification of reality, this abstraction allows for a more manageable and timely analysis [27]. These reasons lead to the decision to base this research investigation and its model on the strategic form with two players.

Another aspect of the model that bears much of its weight is that of how the utility values are determined; see the discussion on utility values in 2.1. Zero-sum and nonzero-sum games are both considered when designing the experimental model. Zero-sum games by their nature define a game in which one side always wins and the other side loses on any particular outcome. This kind of approach does not provide for the situation where each side can gain some from a particular combination of strategy choices. Certainly when one considers a real world situation it can be shown that this is a possibility. Consider a cyber attack in which damage is done or information is lost to the defender, but in which



		Rock			Paper			Scissors		
Rock	Rock	0	0	0	-1	-1	2	1	1	-2
	Paper	-1	2	-1	-2	1	1	0	0	0
	Scissors	1	-2	1	0	0	0	2	-1	-1
Paper	Rock	2	-1	-1	1	-2	1	0	0	0
	Paper	1	1	-2	0	0	0	-1	-1	2
	Scissors	0	0	0	-1	2	-1	-2	1	1
Scissors	Rock	-2	1	1	0	0	0	-1	2	-1
	Paper	0	0	0	2	-1	-1	1	-2	1
	Scissors	-1	-1	2	1	1	-2	0	0	0

Figure 3.1: Three-player zero-sum Rock/Paper/Scissors game in Gambit's form

the defender gains valuable insight into the attacker. In this instance the defender may have lost some utility value, but both sides have also gained and this does not fit the model of a zero-sum situation. The incorporation of this possibility requires that the game be based upon a nonzero-sum utility representation.

Consideration to designing the utility outcomes for the game must also take into account the quantities that are involved. Static representation of numbers allows a game designer to show not only that a certain strategy combination is better or worse for the players than others, but also by how much. However, the values to assign to the strategy choices may not reflect environmental influences. Like the situation in the Hawk-Dove game where resources can change over game iterations, see Figure 2.4 on page 13, a computer network is a landscape that can change over time. Indeed, nodes can be added and removed as computers go up and come down, even the value of a specific node can change due to the information stored on it or its position in a topology [15]. Utility values that are based upon functions grant the model with a simple way to be influenced by environmental changes. The use of variables in these functions shows how certain aspects of the defender and attacker control the values associated with the payoffs achieved. The functional utility

design is therefore more useful in the analysis of game outcomes for this experimental design.

As for the use of hypergame theory over that of basic game theory models, there are several reasons for this approach as discussed in 2.3. A typical game when designed for game theory does not recognize the possibility that each player may be incorrect in their assumptions about the game. When it comes to cyber defense, it is unreasonable to assume that full knowledge about one's attacker or their capabilities can possibly be known; even the attacker is doubtful to have full knowledge of the system that is being attacked or the defensive aptitude [42]. Hypergame theory provides a concrete way in which to insert this difference in knowledge into the analysis. This is done by providing different game views to each player. Strategies may be hidden from the other players through either discounting them as options or because the ability of the player is unknown to the others. In particular the Hypergame Normal Form (HNF) approach provides a built in risk assessor factor by weighing a player's fear of being outguessed. Also, through the use of hyperstrategies, the HNF affords the possibility of increasing utility over that of a standard mixed strategy Nash equilibrium (MSNE).

The previous discussion explains the reasoning behind the hypergame design decisions that are put together for this research. A two-player strategic form game is used for its simplicity in design and the simultaneous move system. Nonzero-sum utilities allow for the possibility that each side can gain or lose based upon strategic decisions. Functional utility values give environmental impact influence upon game conclusions. Hypergame theory contributes the ability to analyze the game allowing for the fact that each side may have differing perceptions on the game. The HNF is used as the model's form so that hyperstrategies can be analyzed for strategy selection. These design decisions are integrated into the experimental game model.

3.2 Model Design

Various approaches to the design of a game theoretical model are examined to provide insight on the procedure that would best suit the needs of a study of hypergame application to cyber security. Previous research, see 2.2, presents examples of model design that show the value that game theory has in a decision making process. A model based upon these ideas that seeks to recreate a real-world scenario provides the best possible source for insight into how to deal with threats to a computer network. However, complexity does not necessarily provide for better answers to questions about the usefulness of game theory in this arena. The addition of strategies and more levels of hypergames to provide a super model that correctly identifies all possibilities in an interaction between attacker and defender is not only improbable, but would be difficult to manipulate [27]. Therefore, it is necessary that the model be focused so it still provides tenable results, yet it needs to have complexity enough that the value of those results is applicable. A balance between usability and relevance to a cyber security situation is achieved through the combination of previous research models.

3.2.1 Influential Models.

The model design proposed by Chen and Leneutre from [15], see section 2.2, fits together well with the selection of game theory attributes described. They provide a basic two-player, attacker and defender, game in strategic form with functional utility values (Figure 3.2). The functions allow for changeable utility values that are based upon the variables they contain, see Table 2.1. The most interesting concept envisioned is that the costs of the various strategy selections, like C_m the cost of monitoring, are considered as percentages of the value of the target, W_i. As the percentage to defend or attack the target increases, the utility decreases. This is a logical design technique given that if the cost percentage to attack two particular nodes is the same, the higher value node returns a greater payoff. However, if cost percentages differ for two nodes it may be that a lower

value node could be more cost effective to attack. This concept also creates the capability of increasing the cost percentage above one hundred, indicating a node that is more costly to attack than it is worth. Therefore, the model examined in this research effort expands upon the Chen and Leneutre model by keeping this main concept intact by having percentage costs associated with new strategic options. The model is further enhanced by changing it into a hypergame.

	Monitor	Not monitor
Attack	$(1-2a)W_i - C_aW_i$, $-(1-2a)W_i - C_mW_i$	$W_i - C_aW_i, -W_i$
Not attack	$0, -bC_fW_i - C_mW_i$	$0, 0$

Figure 3.2: Chen and Leneutre's game model with functional utility values

There are various approaches to the design of hypergames such as those discussed in [12] [24], but the work by Dr. Vane in [52], discussed in 2.3, offers a robust modeling technique that also fits the selected game attributes. The main advantage the HNF has over the other models is that it combines the subgames into one viewable grid instead of having multiple game grids to view. Not only is this model visually appealing, it also offers a straightforward system for strategic decision analysis. The creation of the hyperstrategies, each examining the game from a different perspective, and the inclusion of the g value, as an indication of a player's fear of being outguessed, provides a definitive, quantitative reasoning approach to strategy selection. The HNF also uses values for utility results instead of a system of preferences like the other models. Although originally described using static utility values and zero-sum outcomes, the evolution of the HNF to using functional utility and nonzero-sum payoffs is not hindered because the row player's decision is already uncoupled from the view of the column player. In fact, Dr. Vane describes that the changing of the model to nonzero-sum in this manner should be relatively

straightforward [52]. The addition of the functional utility values does not change anything significantly about the HNF, only that those values need to be calculated prior to the hypergame analysis.

3.2.2 The Experimental Model Design.

In order to provide a quantifiable result on the adaptability of hypergame theory, a game model is introduced. The model is a scenario in which there is an attacker and defender vying for control over a network node. This is a specific example as to how the concepts of hypergame theory could be applied to a cyber defense situation. It demonstrates an overt, targeted attack against the network. Although the concepts introduced are an example of hypergame modeling, one cannot assume that these apply in every situation. Indeed, the purpose of the game model used in this study is only to give an example of how a model may be constructed. The construction techniques used in the creation of the model can be used to examine a wide variety of situations. The true benefit of hypergame analysis is the ability to be adapted to the situation for which the model is built.

Both the Chen and Leneutre, and Dr. Vane's models offer valuable tools for insight into hypergame strategic reasoning. The combination of the two allow for a design that has changeable nonzero-sum utility values and a quick process for the delineation of strategy selection. These two differing, yet compatible, models lead to the creation of a design that gains these benefits. The Chen and Leneutre game grid is expanded to increase strategic selection for both the attacker and defender, while the HNF allows these new strategies to be considered hidden or discounted by the other player. For the purposes of this investigation in the area of network defense, the game grid is also switched from attacker as row, to defender as row because the HNF analysis technique is designed for the row player's perspective of the game. Thus, the model design incorporates the functional utilities and nonzero-sum from [15], keeping the strategic form and two-player setup present in both

influential models, and upgrading the previous model to the HNF from [52]. The resulting game is shown in Figure 3.3.

Attacker / Defender	Not Attack		Attack		Zero-Day Exploit	
Not Defend	0	0	$-W_i$	$W_i - C_a W_i$	$-W_i$	$W_i - C_z W_i$
Defend	$-bC_f W_i - C_m W_i$	0	$-(1-2a)W_i - C_m W_i$	$(1-2a)W_i - C_a W_i$	$-W_i$	$W_i - C_z W_i$
Provide Ruse	$-W_i - C_r W_i$	0	$V_a - C_r W_i$	$W_i - C_a W_i$	$-W_i$	$W_i - C_z W_i$
Shut Down	$-W_i - C_t W_i$	0	$V_a - W_i - C_t W_i$	$-W_i$	$V_a - C_t W_i$	$-W_i$

Figure 3.3: Game model designed for use in this study

Although the attacker and defender players have switched positions the upper left corner of the game grid is recognizable as the Chen and Leneutre model (the top or left most utility values belong to the row player, defender) with additional strategies available to the players. These new strategies are each included in what is referred to as the *full game*. The reason for this distinction is that for the hypergame modeling not all strategies are seen or considered as options by the other player which creates *subgames*. A subgame of the full game is where some of the strategies of the players are removed from consideration because of misperception or through belief that a strategy would never be used. Each player has new alternatives available beyond the original model that creates the hypergame model. The attacker now has a new option, the *zero-day exploit*. The *zero-day exploit* is an attack for which there is no defense because the vulnerability is a previously undiscovered issue [4]. For the purpose of examining the effects provided by additional strategies, the defender can use the *provide ruse* and *shut down* strategies. Providing a ruse, a honeynet or honeypot, is a device for the defender to fool the attacker into thinking they are attacking a real part of the network, but it is a façade designed to give the attacker the appearance they are succeeding in the attack. The *shut down* option is removing access to the network node

entirely, but of course this limits access to even authorized users and therefore may not be a viable option.

The new strategies, as well as the new combinations of them with old strategies, require new functions for utility determination. New variables are also necessary for the calculation of the payoffs. They are summarized in Table 3.1, along with the holdovers from the Chen and Leneutre model (which are repeated here from Table 2.1). For the attacker the zero-day exploit has a cost, C_z, reflected as a percentage of the value of the target. The defender's shutdown option has a cost in amount of time the node is unavailable, C_t, and this can rise the longer the system is down. The assumption for this study is that the attacker's intent is not to have the system shut down and that it is an option available to the defender. Certainly in real-world situations the cost for shutting down could be raised to prohibitive levels to reflect the lack of availability for this strategy. The provide-ruse strategy has a similar cost, C_r, which can also be changed to reflect time and/or sophistication of the ruse. However, the most important change here is the addition of the value of the attacker, V_a. This greatly affects the utility values of providing a ruse or shutting the system down because the more dangerous the attacker, the more worthwhile it is to use these strategies. This enables the defender to delineate between nuisance attackers and those that seek to do real harm.

3.3 Custom Software Development

The process of hypergame analysis can be done by hand, but is a long tedious process that is better suited to a software tool. In order to provide the output required for analysis and testing the software needs to model the hypergame, provide calculation of utility from functions, run multiple games with static or random strategy selection, and be able to update variables and belief contexts between game iterations. Preexisting software is considered an option, but must be able to provide the aforementioned items to be viable. Microsoft

Table 3.1: Experimental Model *Variables*

Variable Symbol	Meaning
a	Detection Rate
b	False Alarm Rate
C_a	Cost of Attack
C_f	Cost of False Alarm
C_m	Cost of Monitoring
C_r	Cost of Providing Ruse
C_t	Cost of Time Down
C_z	Cost of Zero-Day Exploit
V_a	Value of Attacker
W_i	Value of Target

Excel is proficient at modeling the hypergame and can calculate the utility from functions easily. However, the difficulty of running multiple game iterations and updating variables removed it from consideration. A math program, such as Matlab, can do the required calculations and would even be able to run the multiple iterations with changes to variables. These types of programs are not directly related to game theory so software specific to a game theory application is more desirable in this instance. Therefore, software with direct game theory application is considered.

3.3.1 Examination of Game Theory Software.

Perhaps the most intriguing tool available for game theory widely available is the Gambit software. Gambit provides tools to do computations of finite, noncooperative games and can use strategic form [34]. The most promising aspect of Gambit is the simple user interface for the creation of games, Figure 3.4, allowing players and strategies to be

added quickly. It also offers a suite of solution concepts that includes Nash Equilibrium (NE) discovery as well as formats for storing and communicating games to external tools. These options it provides make it a candidate for use in experiments related to hypergame analysis. The source code of the software is written in C++, is available for use and has been upgraded through the years since its original release in BASIC during the mid-1980s [34]. The disadvantages for using Gambit begin with the lack of native hypergame support as it is designed for use as a standard game theory tool. Functional utility values are also not supported, although it does offer easy game creation with static utility values. The software is viable, but without an existing way to infuse hypergame support (specifically that of the HNF) and the requirement to update current code in numerable ways, it is removed from consideration as the experimental tool.

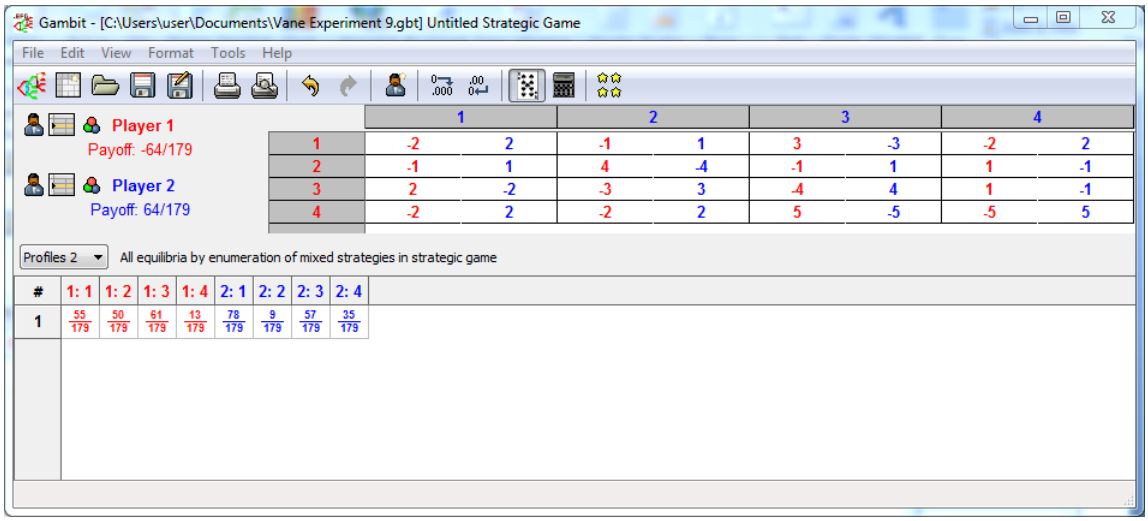


Figure 3.4: Two-player 4x4 strategic game example with equilibriums from Gambit

Also there is software package, called HYPANT, that is specifically designed as a hypergame analysis tool. Written by Lachlan Brumley [13] in 2003, it contains a proprietary language known as Hypergame Markup Language (HML) that allows for game

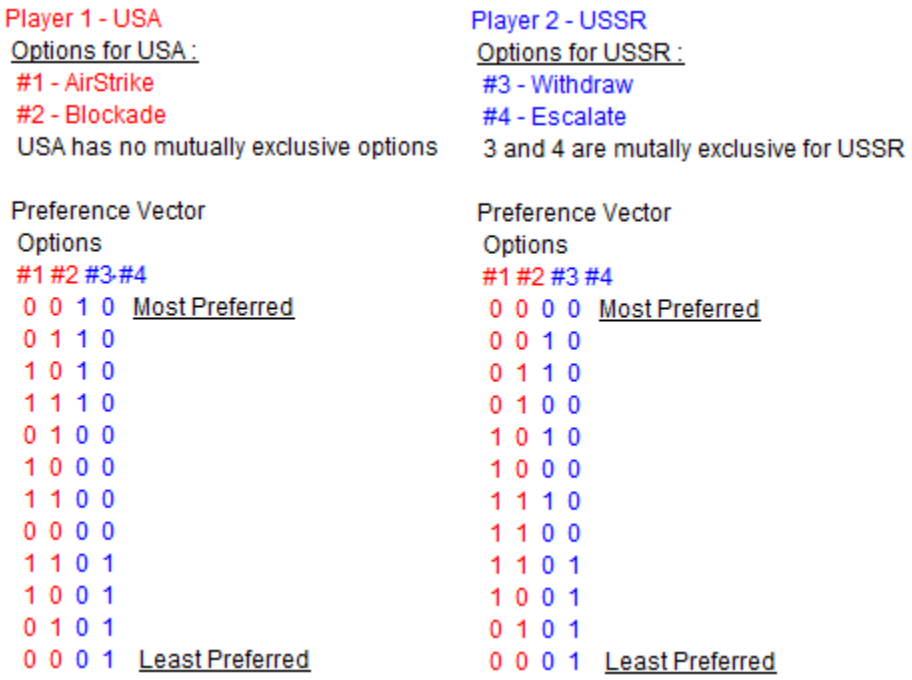


Figure 3.5: Example output of a two-player HYPANT game form (Cuba blockade)

player's specific views of the game being played. In other words, it contains the most critical aspect of hypergame analysis, full game versus subgame views. Several examples of the usefulness of the tool are provided including the hypergame example from [12] the Fall of France. These examples show that HYPANT does succeed in its goal as being a hypergame analyzer. However, it does not use the type of utility values required by the HNF and instead relies on stability and unilateral improvement values present in the Frasier and Hipel hypergame model in Figure 3.5. The differences in the analysis technique used by HYPANT and that in Dr. Vane's HNF are great enough that upgrading HYPANT in this regard would be more extensive that writing an HNF software tool from the ground up!

Another program using the HNF is in existence and is called Security Policy Assistant (SPA). SPA is designed to facilitate the decision making process involved in the release or withdrawal of classified documents from foreign disclosure [48]. It provides an input GUI

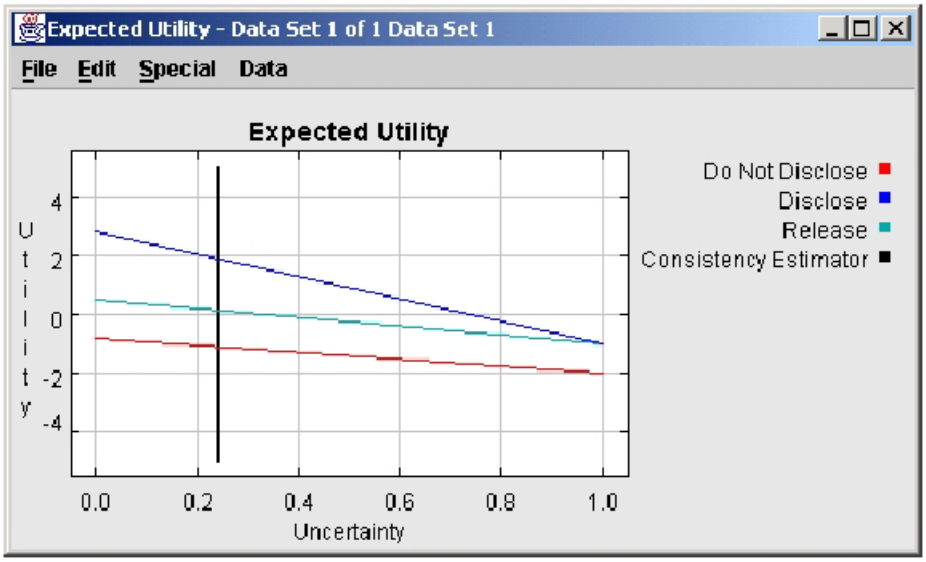


Figure 3.6: Expected utility analysis window from SPA [48]

that allows the entry of information relating to the HNF. Once the data is entered it gives output and graphs reminiscent of the outcomes present in [52]. A graph that is particularly familiar is the graph of expected utilities, Figure 3.6, which is much like Figure 2.9 on page 28. This software is most like the avenue being pursued and although is not precisely what is required does offer some of the features desired. Unfortunately, although the manual for SPA was obtained, at the time of this research investigation the software is unavailable and could not be examined or used. However, the fact that the concept has been used in an application for determining disclosure policies for classified documents is evidence that there is application for hypergame theory beyond previous military conflict analysis.

3.3.2 HNF Analysis Tool (HAT) Software Design.

The lack of currently available software that provides all of the aspects required for the testing and analysis motivates the decision to develop a custom piece of software using the Java programming language. This software incorporates the HNF by allowing the input of a game design via Extensible Markup Language (XML) Figure 3.7. This form of input is selected because unlike the proprietary HML used by HYPANT, it is a simple, flexible

text format with a defined schema [40]. Software tools for retrieving data in XML format already exist within Java. The game information present in the XML allows the software to instantiate objects with the data for creation of a game environment. The main components of the HNF are either included or derived from this information. Once the game is setup, game iterations can be run with either random or static strategy selections, and variables can be updated between games. The HAT customized software is designed to analyze the XML provided game with the principles outlined in [52] using an object-oriented design that encompasses its components.

```
<?xml version="1.0" encoding="UTF-8"?>

<root>
    <Game Name="Experiment1" HasSubLevels="False">
        <Player Name="Row">
            <PlayerStrategy Name="Not Defend" Usage=".33" Hidden="False"/>
            <PlayerStrategy Name="Defend" Usage=".33" Hidden="False"/>
            <OpponentStrategy Name="Not Attack" Hidden="False">
                <PlayerStrategy Name="Not Defend" Utility="0" Hidden="False"/>
                <PlayerStrategy Name="Defend" Utility="-0.24" Hidden="False"/>
            </OpponentStrategy>
            <OpponentStrategy Name="Attack" Hidden="False">
                <PlayerStrategy Name="Not Defend" Utility="-1" Hidden="False"/>
                <PlayerStrategy Name="Defend" Utility=".4" Hidden="False"/>
            </OpponentStrategy>
        </Player>
        <Player Name="Column">
            <Name>Attacker</Name>
            <PlayerStrategy Name="Not Attack" Usage=".8537" Hidden="False"/>
            <PlayerStrategy Name="Attack" Usage=".1463" Hidden="False"/>
            <OpponentStrategy Name="Not Defend" Hidden="False">
                <PlayerStrategy Name="Not Attack" Utility="0" Hidden="False"/>
                <PlayerStrategy Name="Attack" Utility="0.8" Hidden="False"/>
            </OpponentStrategy>
            <OpponentStrategy Name="Defend" Hidden="False">
                <PlayerStrategy Name="Not Attack" Utility="0" Hidden="False"/>
                <PlayerStrategy Name="Attack" Utility="-0.8" Hidden="False"/>
            </OpponentStrategy>
        </Player>
    </Game>
</root>
```

Figure 3.7: Example of the XML game format used by the HAT software

HAT is written in Java with the NetBeans Integrated Development Environment (IDE) 7.2.1 [3]. This IDE provides concurrent error checking while coding and compile time results. The versioning software Git is used within NetBeans to ensure proper backups are completed and so that changes made can be undone if necessary. Versioning is an important part of current software engineering practices because without it, changes that break performance of the program may be difficult to locate and reverse. Another reason to have versioning, although not employed because of there is a single developer, is that when changes are made by several designers, they can be merged together gracefully. Without the ability to merge updates, changes to the same code at the same time cause conflicts. NetBeans also has a refactoring solution built in. Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior [21]. This does not provide the user with any outward changes, but can be very important for the developers allowing the reuse of code, consolidation of structure, safe removal of old code, and easy renaming of code artifacts to better describe their role in the program. NetBeans IDE 7.2.1 is thus selected because of these code structure and time saving devices, which facilitates the software's object oriented design.

Individual objects encapsulate the components of the game mechanics used in the HNF (Figure 3.8). The game consists of the players involved, the players cross reference their opponent's strategies, and the opponent strategies cross reference that player's available strategy choices which lead to the player's utility values when these strategies are selected together. The current design allows for a two-player game, but the object-oriented design allows for the extensibility to become an n-player game. The HAT loads a game XML file, selectable by the user, and creates the objects based upon the data from it. This is done by a process of parsing through the components of the file which correspond to the object types present in the code. Each node of the XML file describes an object in the code, providing all the information required that creates the full game structure as shown

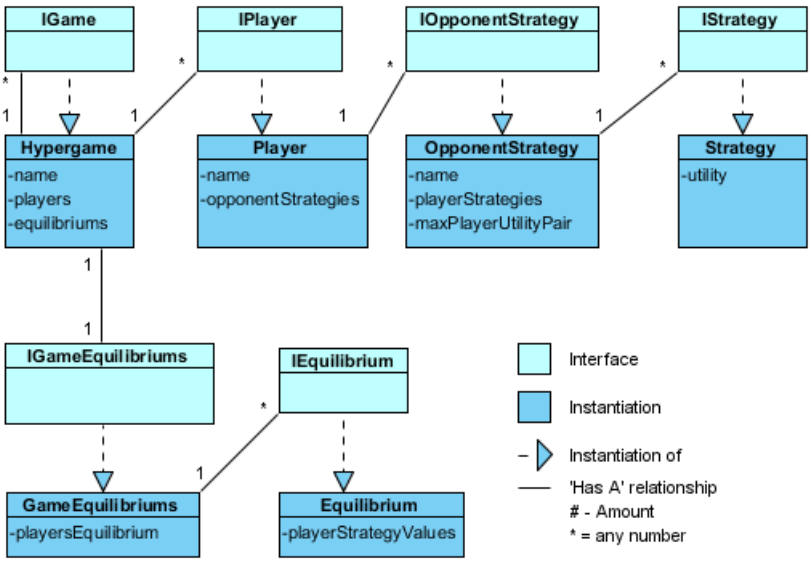


Figure 3.8: Core design of HAT software in Universal Modeling Language format

here output in the software window of Figure 3.9. The creation of these main components is done through instantiation by means of a design pattern called the *factory pattern*. The main reason to use a factory pattern is so that when changes need to be made for object creation there is only one location where modifications need to be done [22]. This facilitates quick and easy changes during design phase, but also allows for extendibility for later software updates.

Each strategy cross referenced with opponent strategies utility has the ability to use either direct values or functions. Static values are easily input directly into the object by parsing the characters in the XML, but functions require a more dynamic approach. The use of function utility values requires not only a scheme for input via the XML file, but also for the evaluation of them through software. The variables present in the functions are given their own section in the file that consists of identifying the symbol used and an initial value for calculations. These variable identifiers are then incorporated into utility functions by putting them into brackets for easy identification by the software, shown in the example

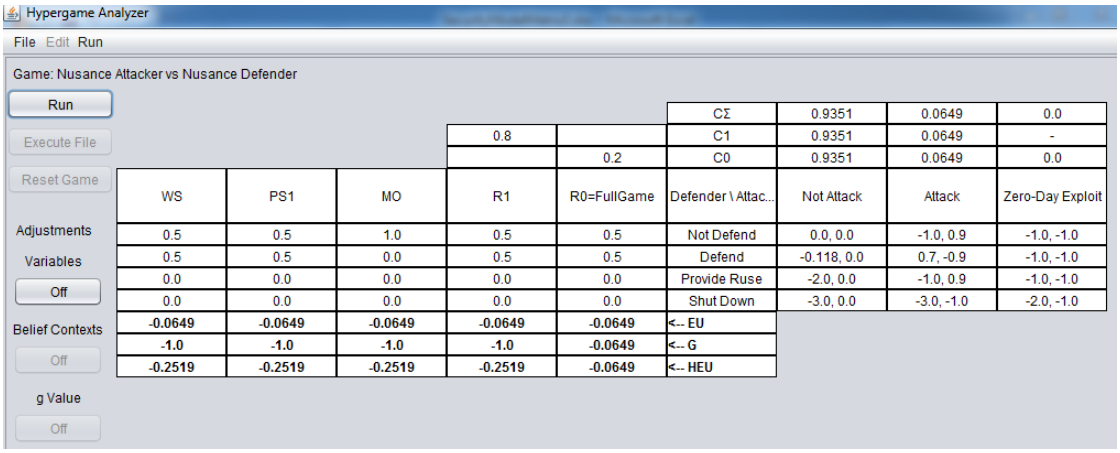


Figure 3.9: Example of the main view of the HAT software with game loaded

function of Equation (3.1). The letter combinations in the brackets correspond to the

$$-\{b\} * \{Cf\} * \{Wi\} - \{Cm\} * \{Wi\} \tag{3.1}$$

variable names defined by the XML file. The identifiers are replaced by the initial values for initial utility calculation, but the function form is retained for updating when variable values change. This allows a critical component of the experiments to be available to the game software. The functional utility code is a modified version of [18], where the initial function string with variables is replaced by the variable values. After variable replacement the function solver can provide concrete numbers for the utilities. Once utility values are tabulated, the calculation of NEs commences.

The NEs for the hypergame and subgames are calculated using the Lemke-Howson algorithm (3.1) because it works for non-zero sum bimatrix games. This is accomplished with Java code that is an altered version of the one created by Junling Hu in an applet [26]. The advantage of updating this code for use by the HAT software is that changes made can be validated as correct by entering the same information into the supplied applet. Also, although not checked by this study, Hu claims that the program outperforms Gambit on the speed of obtaining the NEs. There is no guarantee that the algorithm finds all NEs

because at most it finds $n+m$, but at a minimum one is found as stated by Nash's Existence Theorem [31] [36]. The NE is then used by the software as the initial belief context in the hypergame, C_0. Once determined, the rest of the game is built using additional belief contexts which allows for the creation of the hyperstrategies.

Algorithm 3.1 Lemke-Howson [39]

Require: P_1 to be the polytope of Player 1 utilities

Require: P_2 to be the polytope of Player 2 utilities

Let x be the all-zero vector 0 of length m from P_1

Let y be the all-zero vector 0 of length n from P_1

Let k_0 be any label of x

Let $k \Leftarrow k_0$

loop

In P_1, remove the label k from x; let x' be the new vertex and k' the label added

$x \Leftarrow x'$

if $k' = k_0$ **then**

stop looping

end if

In P_2, remove the label k' from y; let y' be the new vertex and k'' the label added

$y \Leftarrow y'$

if $k'' = k_0$ **then**

stop looping

end if

$k \Leftarrow k''$

end loop

print (normalize(x), normalize(y))

The belief contexts are a combination of a probabilistic weight value, row's belief that column chooses that specific context to which it is assigned, and the associated Column Mixed Strategy (CMS)s, the percentages of each strategy available to column. Each supplied belief context is a subgame where any column strategy given a percentage use of zero is removed. There are two ways in which subgames are determined: supplied belief contexts within the XML game file or removing strategies labeled as hidden. The column player's view of the game still removes any hidden row player strategy, but the row player sees the subgame differently as all his or her strategies are available. This concept holds true even when the belief contexts are not provided in the XML file, in which case only one belief context is built through the removal of column and rows hidden strategies. The percentages for the CMSs for a built belief context is determined by the NE values for the subgame, as seen by column. *Hyperstrategies* are created by examining the subgames that exist in the hypergame framework.

The hyperstrategies generated by the software are the R_x subgames, Modeling Opponent (MO), Pick Subgame (PS), and Weighted Subgame (WS). The R_x subgames are not technically hyperstrategies, but they are included in the design for completeness. Their values are the MSNE for the row player for the subgames as seen by column (any hidden row strategies are not used in the calculation). The MO is made by examining each strategy available to the row player and the expected utility it obtains versus the aggregate belief context, C_Σ. The best utility strategy is selected as the MO hyperstrategy, unless it is the same as the R_0 full game, then it is not included. A PS hyperstrategy is created for each subgame (i.e. belief context) included in the hypergame. The MSNE is assigned to row's strategy selections using row's view of the subgame (all row strategies available, but without column's unused strategies). Finally, the WS is made by aggregating the PS and the R_0 full game strategy selection percentages. The values are normalized through multiplication by the associated belief context's probabilistic value (the R_0 is the full game

NE from C_0), then they are added together to determine the strategy percentage value for the WS.

Each hyperstrategy, and the R_x subgames, calculates the Expected Utility (EU), worst case utility (G), Hypergame Expected Utility (HEU). The EU for each hyperstrategy is given by cross multiplication of utility values with row's strategy selection percentage and aggregate column strategy selection in $C_\sum$. Each strategy utility is added together for the final EU value. Determination of the G value is based on worst case outcome. The lowest utility available to row given a player's strategy selection is multiplied by that strategy's percentage use, which are added together to give the G value. Once the EU and G are finished, they enable the calculation of HEU. It is calculated by using Equation (2.4) on page 27. The value applied as the fear-of-being-outguessed, g, is instrumental in determining the outcome of the HEU for each hyperstrategy. The g value can be predetermined or changed after a game is executed so that the next game can have differing HEU values. This is important because the HEU is instrumental in hyperstrategy choice.

After the game is fully built, it can be executed. The execution of a game consists of selection of player strategies and determination of utility values obtained. A column player's strategy choice can be selected by means of a usage value or a usage file. Usage values incorporated into the XML game file are percentages assigned to any column strategy. The game executor then uses the usage percentages to choose a column strategy stochastically. On the other hand, if a usage file is provided it is a series of strategy names and the executor simply selects the next one in line. The row player's strategy choice is determined through selection of hyperstrategy. The hyperstrategy with the highest HEU value is used and a strategy is selected by random number. The two strategies used then determine the utility that each player receives in the game outcome. Game results are

saved to a file in a comma separated values (CSV) format that can easily be imported into supporting software.

The game has four additional *update* modes under which the games are executable: variables, g value, belief context, and all of them combined. The variables are updated by way of providing the game with an update technique for specific variables that can be affected by strategy choices. Updates to the variables create a more realistic scenario as costs for certain actions may change over time and as a result of previous choices. The XML file *variable node* is able to contain updates that tell the software how to update the variable between iterations. Inputting the player's names, one or more players, and strategy choice, up to one strategy per player, tells the software that the update to the variable occurs when all of the strategy choices appear in the results of the current iteration. Each set of player's strategy choices is coupled with a value that changes the variable and this is established as either an incremental change, up or down by this amount, or direct change, the variable is set to the value. In this manner as the software progresses through the iterations of the game, utility values update according to the settings it is given. These changes not only affect utility values, but also possibly the equilibriums and hyperstrategies. Game data is recalculated before the next iteration is run so that any changes made are incorporated into the next iteration execution.

The g value update is accomplished differently and does not have as dramatic effect on the whole game data as the variable updates because it can only change the HEU amounts. This update is accomplished with a simple check of whether or not the defender was able to obtain the utility that was expected during that game iteration. Since the g value is an indication of fear of being outguessed, when utility value is as expected, it goes down; similarly, when the utility value is not as expected, it goes up because the player has been outmaneuvered. The increase and decrease values are settable within the XML file, so they can be different for any game. The expected utility of the iteration is the max utility

available in the row corresponding to the defender's selected strategy. After the execution of the game strategy selections, the obtained utility of the defender is compared to the expected utility and the g value is updated accordingly. The lowering of a g corresponds to correct choices and hyperstrategies are more likely to be selected for choosing strategies in later games; conversely, the raising of g results from poor strategy selection causing the player to more likely pick the full game NE for strategy selection.

The belief context updates are affected by the variable changes. The initial belief contexts when based upon the utility values, in other words based upon a subgame view within the full game, can change when those values do. In essence, if the initial game changes then it can be assumed that a player's belief about how the game is being played would change along with it. So when utility values are changed the software reevaluates the MSNE of the subgame that resulted in the initial belief context. The new mixed strategy values for the opponent are input into the belief context. The updated context has direct influence on the $C£$ value and therefore can affect the hyperstrategy values for the row player. Once the changes are made to the belief contexts, the new values are applied to the hyperstrategy calculations. When the next iteration of the game is played, these changes may have an impact on row's hyperstrategy HEUs which in turn affects the strategy selection process for the current iteration of the game.

The entire process of the HAT software takes the form of an Observe-Orient-Decide-Act (OODA) loop. The initial observation phase is the loading of the XML game file into the software objects. The data entered is used to determine the utility values and the equilibriums as the orientation phase. Included in this phase is also the calculation of belief contexts and hyperstrategy mixed strategy percentages. Calculations made in the orientation phase are used to determine the strategy selection by the row player during the decide phase. The selected strategy is used for the game iteration during the act phase and the utility values obtained are established. Game results are then used by the software

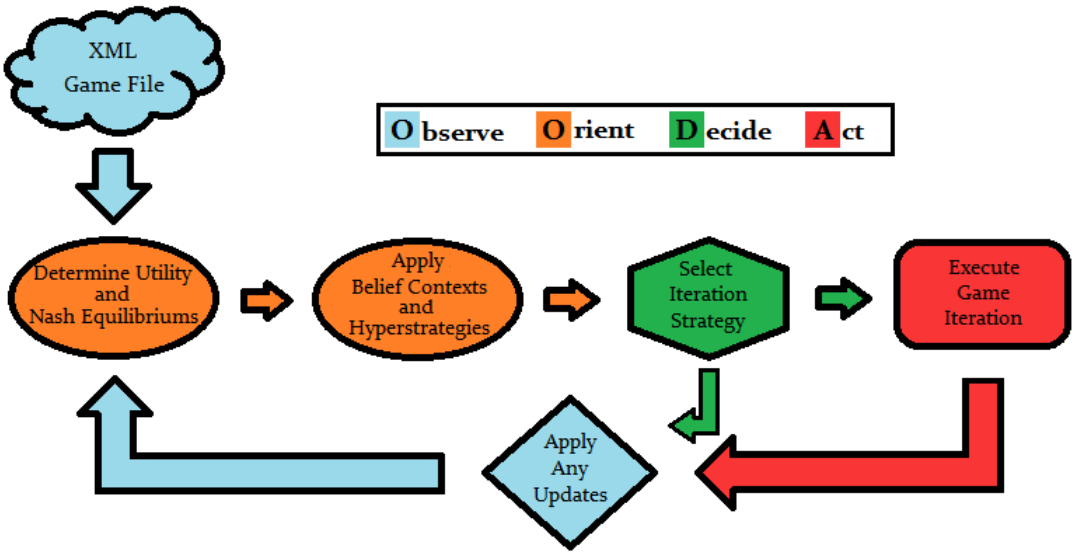


Figure 3.10: The HAT processes as an OODA loop

to apply any updates that have been mandated by the outcome, which is the return to the observation phase. It is at this point that the OODA loop has come full circle and the process repeats until the final game iteration is complete. *The coupling of the OODA loop and hypergame analysis is a unique aspect of the HAT software that is not available elsewhere.*

3.4 Summary

The valuable portion of the game model design is the demonstration of how an adaptable hypergame model is constructed. In the consideration between extensive and strategic form games, strategic form is the choice because of its matrix style layout, ease of control, and the applicability to the cyber conflict scenario. The number of players is kept at two, attacker and defender, to limit the scope of the study and maintain manageability. Game utility values are set as non-zero sum because in network defense there is rarely a clear-cut winner and non-zero sum games allow for these shades of grey. Functional utility values are incorporated into the design because they give the ability to apply changes over

time and update the payoffs accordingly. Finally, all these decisions are incorporated into a hypergame design that takes into account participant's differing views of the game. The model designation is a strategic form, two-person, non-zero sum, and functional utility hypergame. The model is a scenario example designed to introduce the concepts required for hypergame analysis and game adaptation.

The game model is a hybrid composition of two previous research efforts as described in [15] and [52]. The Chen and Leneutre game model closely fits the design decisions of the experimental model. Their model describes a simple attacker/defender scenario that uses functional utility values that create a changeable outcome based upon variables. The advantage is that it can more resemble a real-world scenario than static utility values. Dr. Vane's hypergame modeling technique is an expanded strategic form that shows the differing views of the player's without requiring multiple game grids. He also provides mathematical tools for analysis of the design that can account for the row player's risk in the strategic decision making process. The combining of these two models creates a new design that upgrades the Chen and Leneutre technique into the hypergame arena and Dr. Vane's HNF becomes more robust by using functional, non-zero sum utility values.

The requirements for the game model the directly affect the decision on what software to use for the experimentation. The Gambit software is considered an excellent tool for game modeling and NE calculation; however, it does not provide the necessary functional utility values or a native way to model the HNF, two primary model selections. HYPANT is an available hypergame analysis software that uses a proprietary game input language, HML, but it also lacks functional utility values, instead relying on the stability and unilateral improvement technique. The unavailable software package, SPA, does use the HNF for strategy decision analysis as shown by the manual that is available. Although it does appear that access to the software would be useful, the idea behind it is not network

defense techniques. Thus, the decision to create a custom software tool is the choice that is made.

The HAT software is designed with the goal of providing a tool that allows the analysis of the experimental hypergame model design. A game is entered via XML file and the Java code turns the data into objects within the tool that constitute the game. Software objects are made to coincide with the aspects of a game model: hypergame, players, strategies, and equilibriums. The game has the ability to create utility values from functions and variable values. MSNE results are calculated within the software tool using the Lemke-Howson algorithm. Any pure strategy Nash equilibrium (PSNE) found with this algorithm appears as an MSNE, but with a percentage of one-hundred. Belief contexts are either entered into the XML file, or can be created via subgames which are player views minus the hidden strategies. The Row Mixed Strategy (RMS)s, MO. PS, and WS are calculated using the techniques outlined in [52]. Each one provides its own EU, G, and HEU values that are used to determine strategy choice. The fully built game can be executed with usage values assigned to the column player's strategies and the row player's strategy percentages determined with the RMSs. The players' strategy choices determine the utilities each obtains and thus the outcome of that game. Multiple iterations of a game are also able to be executed with subsequent iterations being affected by previous results. This software progression follows the OODA loop strategic level decision process.

IV. HAT Software Experiment Design Methodology

EXPERIMENTS for the software are designed based upon the hypergame model. Initial experiments are created to show that the HNF Analysis Tool (HAT) software functions as needed by the model requirements. Games are designed that use the model in a software simulation of conflict. Data collected from experiments is used to evaluate the hypergame theory model's utility as a tool for network defense.

4.1 Experiment Design

4.1.1 Pre-Experiment Tests.

Software use requires that expected output is produced with any given input. In order to validate that the software is operational, simple sample games are created and input into the game software. This is done concurrently with software development at certain stages to ensure that the software continues to provide proper output. The sample games are designed incrementally to test the most recent developments in the code which prevents errors from becoming more difficult to locate. The initial games are taken from basic game theory so that a solid base is constructed on which the more complex games can be built. The Extensible Markup Language (XML) file that inputs a basic game is simple, straightforward, and human readable as can be seen here (Figure 4.1). When compared with the Prisoner's Dilemma game grid (Figure 2.1, page 11), easily recognizable features are: player names, strategy names, and utility values. A test output is prepared and produced when the software is run and a visual check that the display contains the correct information is performed. These tests give confidence that the software is behaving in the manner for which it is designed.

The initial tests show that the required information is input into the program from the XML game file. They are a simple validation by text output that not only was the correct

```
<root>
    <Game Name="Prisoner's Dilemma" HasSubLevels="False">
        <Player Name="Prisoner A">
            <PlayerStrategy Name="Stay Quiet" Hidden="False"/>
            <PlayerStrategy Name="Confess" Hidden="False"/>
            <OpponentStrategy Name="Stay Quiet" Hidden="False">
                <PlayerStrategy Name="Stay Quiet" Utility="-1" Hidden="False"/>
                <PlayerStrategy Name="Confess" Utility="0" Hidden="False"/>
            </OpponentStrategy>
            <OpponentStrategy Name="Confess" Hidden="False">
                <PlayerStrategy Name="Stay Quiet" Utility="-5" Hidden="False"/>
                <PlayerStrategy Name="Confess" Utility="-3" Hidden="False"/>
            </OpponentStrategy>
        </Player>
        <Player Name="Prisoner B">
            <PlayerStrategy Name="Stay Quiet" Hidden="False"/>
            <PlayerStrategy Name="Confess" Hidden="False"/>
            <OpponentStrategy Name="Stay Quiet" Hidden="False">
                <PlayerStrategy Name="Stay Quiet" Utility="-1" Hidden="False"/>
                <PlayerStrategy Name="Confess" Utility="0" Hidden="False"/>
            </OpponentStrategy>
            <OpponentStrategy Name="Confess" Hidden="False">
                <PlayerStrategy Name="Stay Quiet" Utility="-5" Hidden="False"/>
                <PlayerStrategy Name="Confess" Utility="-3" Hidden="False"/>
            </OpponentStrategy>
        </Player>
    </Game>
</root>
```

Figure 4.1: Basic game XML example (Prisoner's Dilemma)

information received, but also that the data can be retrieved and output to the console. This is an important step in the process because without this test there would be no confidence that data could be manipulated correctly if the information does not begin with the right input. The software loads the XML game file and generates the text output to the console. The game is output in grid format as is expected with a strategic form game. The console displays the game name, the player names with row player on the left and column player at the top, each player's strategies and the utility values cross referenced from the strategies. The output is then visually validated that it outputs the model of the game that was intended.

Once the software is guaranteed to have the proper data input through XML manipulation, the next test is designed to show that pure strategy Nash equilibrium (PSNE)s can be produced. The code implements the simple method for finding the PSNE as shown in 2.1 on page 12. This method involves marking the highest utility available to a player for

each strategy available. After each player's high utility values have been found, those that meet in the same cell of the matrix are PSNE. The software uses the process of saving the max utility value for a given player's strategy and the opposing player's strategy as the data is input into the system; thus, quick retrieval of strategy and utility values for comparison to the other player is possible. This is not an extremely useful technique because it cannot locate any mixed strategy Nash equilibrium (MSNE) while the methods that can locate MSNEs will also find the PSNEs (tagged as MSNE with the selected strategies at 100% usage). However, it is another important step in making sure that the data is entered, retrieved, and manipulated correctly within the software code. When the PSNE tests are complete, the Nash Equilibrium (NE) method is tested.

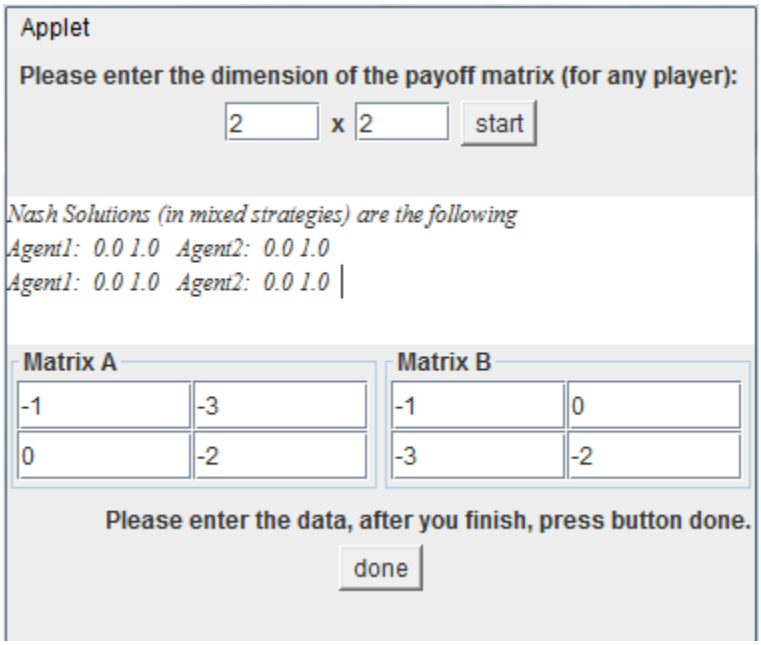


Figure 4.2: Hu's applet showing Prisoner's Dilemma entered

The NE portion of the code uses the Lemke-Howson algorithm adapted from [26], see 3.3.2. Again, a text visual is used to validate the output, but because the NE contains calculated values an external validating source must be used. The Lemke-Howson

algorithm code is included in an applet that allows a game bimatrix to be input and produce NE outputs (Figure 4.2). The applet's output is not enough to ensure that the developed software provides the correct values for NEs because the applet's output is not known to be correct. Instead, the Gambit software is used to check the applet. Once the applet is determined to be operating correctly, necessary changes to make the code work within the software are shown to be correct by comparing output of the HAT software and the applet with identical game input. This is done by having the HAT output its derived NEs to the console in text form for visual comparison with Gambit and the applet. When all three sources of MSNE results coincide, this shows that the software is operating within reasonable expectations for NE calculations.

After the NE is determined to be operating, the function utility portion of the software is tested. The first test is a suite of selected functions that check the mathematic operation of the function code. Each function example is designed to check various aspects of function build: addition, subtraction, multiplication, division, order of precedence, and parentheses overriding of precedence. Predetermined values via calculator are checked automatically by test code with function code output. Next sample games using functions for the utility values are tested. These tests require that utility functions replace the variable identifiers with the values given to the software from theXML file. Like the NE tests, due to the calculation of the utilities an outside source must be used for validation of output. An Excel spreadsheet is designed to emulate the utility functions and variable inputs so it can calculate expected values for comparison (fig:utilityFunctionSpreadsheet). The utility results from the HAT, which are output to the console via text, are visually compared to those given by the spreadsheet. The combined results of the function code test suite and the sample games give the evidence required that ensures accuracy of the HAT function calculator.

a	detection rate for attack	0.83									
b	false alarm rate	0.05									
C_a	cost of attack	0.25									
C_f	cost of false alarm	0.75			Not Attack		Attack		Zero-Day Exploit	
C_m	cost of monitoring	0.25		Not Defend	0	0	-1	0.75	-1	0.2
C_r	cost of providing ruse	0.5		Defend	-0.2875	0	0.41	-0.91	-1	0.2
C_t	cost of time down	1		Provide Ruse	-1.5	0	0.3	0.75	-1	0.2
C_z	cost of zero day exploit	0.8		Shut Down	-2	0	-1.2	-1	-0.2	-1
V_a	Value of Attacker	0.8								
W_i	Value of Target	1								

Figure 4.3: Spreadsheet for HAT utility function check

Finally a test is performed to ensure the Hypergame Normal Form (HNF) is properly structured and calculations of the Row Mixed Strategy (RMS)s and belief contexts are done according to requirements. An example as outlined in [52] is used as an example for a visual check of the form data. An XML game file is built according to the visual example. Once the data is read in by the HAT software it is visually displayed in the main window. Output is shown in a fashion very similar to that of the HNF with the belief contexts on top (with their probabilistic values and Column Mixed Strategy (CMS)s), the RMSs to the left, the main strategic form game on the right, and the Expected Utility (EU), worst case utility (G), and Hypergame Expected Utility (HEU) values below. A visual check of the numbers entered in the graph versus the values in the example determines the validity of the software modeling. After this final check experiments involving games created in the experimental model form are conducted.

4.1.2 Experimental Games.

The experimental games involve pitting a defender and an attacker against each other with strategy selections using the hypergame model. Games are run between differing types of attacker classes and defender classes. The attacker classifications reflect levels of intensity and resourcefulness that can be encountered in a cyber defense scenario. The defenders are built in terms of expected level of attacks. Games are run with the various attackers and defenders interacting and show how effective the hypergame model is in the

given situations. The various games are developed in an increasing sophistication manner which presents how updates in values between iterations of a game affect the outcome.

Table 4.1: Attacker types and their usage percentages

Attacker Type	No Attack	Attack	Zero-Day Exploit
Nuisance	93.51%	6.49%	-
Low-Level	90.17%	9.83%	-
Mid-Level	86.61%	13.39%	0%
High-Level	0%	0%	100%
All-Out	0%	0%	100%

The attackers are decomposed into five different categories of threat: nuisance, low-level, mid-level, high-level, and all-out (Table 4.1). These attacker examples are designed to cover a wide variety of attack styles for the purposes of this study. A nuisance attacker presents an opponent with an unsophisticated level of attack, or one that seeks to do damage that is not considered of much consequence to the defender. The low-level attacker is attempting to do real harm to the system by stealing information or reducing access. The mid-level attackers denote a more dangerous opponent, one that is consistently barraging the network in an attempt to get critical information or deny service. The high-level attack is a sophisticated attempt to destroy, steal, and deny. The all-out attacker is designed to reflect the scenario where the attacker uses any means necessary to damage the network. This opponent is likely to unleash a deadly barrage of attacks and use all the zero-day exploits at the attacker's disposal. All these amounts are based upon NE amounts given by the attacker's view of the game and do not include the hidden strategies for the defender, provide ruse and shut down. Thus, although it appears incorrect that the high-level and all-out attackers both expend 100% of their attacks on zero-day exploits, this is the result

of initial game setup and does not reflect changes that may happen during game play. Also these are examples of how a game may be designed, real-world models may differ depending upon circumstances such as the inability to shut down the network. Each of these attackers are pitted against defenders, which are similarly classified by the style of attack they are expecting to occur.

Table 4.2: Defender types and their initial belief contexts

Type	Probabilistic Value	No Attack	Attack	Zero-Day Exploit
Nuisance C1	0.8	93.51%	6.49%	-
Nuisance C0	0.2	93.51%	6.49%	0%
Low-Level C1	0.8	90.17%	9.83%	-
Low-Level C0	0.2	90.17%	9.83%	0%
Mid-Level C1	0.8	86.61%	13.39%	0%
Mid-Level C0	0.2	0%%	100%	0%
High-Level C1	0.8	0%	0%	100%
High-Level C0	0.2	0%	100%	0%
All-Out C1	0.8	0%	0%	100%
All-Out C0	0.2	83.33%	0%	16.67%

The defenders are also named in the same fashion as the attackers: nuisance, low-level, mid-level, high-level, and all-out. The defender classes have belief contexts that are equal to that of the reciprocal attacker type (see Table 4.2) with probabilistic values set at 0.8, which means the defender suspects with an 80% surety that type of attack is occurring. This setting still allows for a 20% possibility that the attacker is taking into account the full game by applying this to the probabilistic value of the $C0$ belief context. The defender is reasonably sure that the attack is going to be as expected, but reserves some

expectation that the assumption is incorrect. However, since the hypergame is seen from the defender player's point of view, various updates to the variables are apparent with the different classes Table 4.3. The changes in variables indicate differences in costs associated with the game functions due to the different level of attacks expected.

Table 4.3: Defender type initial variable values

Variable	Nuisance	Low-Level	Mid-Level	High-Level	All-Out
Detection rate for attack (a)	0.90	0.88	0.85	0.83	0.80
False alarm rate (b)	0.02	0.03	0.04	0.05	0.06
Cost of attack (C_a)	0.10	0.15	0.20	0.25	0.30
Cost of false alarm (C_f)	0.90	0.85	0.80	0.75	0.70
Cost of monitoring (C_m)	0.10	0.15	0.20	0.25	0.30
Cost of providing ruse (C_r)	1.00	0.75	0.50	0.25	0.00
Cost of time down (C_t)	2.00	1.50	1.00	0.50	0.00
Cost of zero-day exploit (C_z)	1.00	1.00	1.00	0.80	0.50
Value of Attacker (C_z)	0.00	1.00	2.00	3.00	4.00
Value of Target (C_t)	1.00	1.00	1.00	1.00	1.00

Detection rate, a, is considered to decrease as the attacker sophistication increases, reflecting a craftier foe. False alarm rate, b, increases because more scrutinization of anomalous behavior in the system. The cost of attack, C_a, for the attacker and the cost of monitoring, C_m, for the defender are both higher with each level due to the increased complexity required. The cost of a false alarm, C_f, is decreased as the defender is more willing to accept these in the more dangerous situations. Cost of a zero-day exploit, C_z, drops for the high-level and war attacker because they are willing to use them and have resources to obtain them. It is obvious why the value of the attacker, V_a, increases with

each level, the opponent is more dangerous. Finally, cost of providing ruse, C_r, and cost of time down, C_t are both decrease with the attacker type that is suspected because a defender is more willing to accept these options as threat level increases. The value of the target, W_i, is kept at one because as shown in [15] the value is affected by other variables on a percentage basis and is used to determine if a node is in the attacker's set of worthwhile targets. For the purposes of these experiments it is assumed that the node is being attacked.

Each type of defender is matched against each type of attacker. These game runs should show that the defender that is designed against the attacker is effective, but that not all defenders are effective against all attackers. When higher level attackers are pitted against lower level defenders, the low level defenders are outmatched. However, even some high level defenders against low level attackers fare worse than the defenders specifically designed to combat them. This is due to incorrect strategy selection; in other words, defending too heavily against a threat that does not exist. Each game will be run 10 times for 100 iterations each in two fashions, stochastic and static attacker strategy selection. The stochastic game allows a random number based upon the usage percentages provided the attacker class to determine which strategy is used. The static game uses a file for providing the strategies in a set order, but still maintaining the usage percentage. Both types still have the defender strategy selection based upon the percentages in the hyperstrategy selected by the Hypergame Expected Utility (HEU) by using a randomly generated number. The two types are used to show if model behavior changes with constant input versus randomized. These initial experiments are done without any updates to variables, g value, or belief context amounts.

Incorporating variable updates into the game iterations allows the game utilities to change reflecting the previous course of actions. These experiments concentrate on updates to the cost of providing a ruse, cost of time down, cost of zero-day exploit, and value of the opponent, see Table 4.4. The cost of providing a ruse and cost of time down increase

Table 4.4: Variable update changes

Variable ID	Variable Name	Strategy Selected Defender	Strategy Selected Attacker	Value Changes Amount
C_r	Cost of Providing Ruse	Not Defend	Any	-0.5
		Defend	Any	-0.5
		Provide Ruse	Any	0.1
		Shut Down	Any	0.1
C_t	Cost of Time Down	Not Defend	Any	-0.02
		Defend	Any	-0.02
		Provide Ruse	Any	0.5
		Shut Down	Any	1.0
C_z	Cost of Zero-Day Exploit	Not Defend	Zero-Day Exploit	0.1
		Defend	Zero-Day Exploit	0.4
		Provide Ruse	Zero-Day Exploit	0.1
V_a	Value of Attacker	Not Defend	Not Attack	-0.01
		Defend	Not Attack	-0.01
		Provide Ruse	Not Attack	-0.01
		Not Defend	Attack	0.5
		Defend	Attack	0.1
		Provide Ruse	Attack	0.5
		Not Defend	Zero-Day Exploit	0.5
		Defend	Zero-Day Exploit	1.0
		Provide Ruse	Zero-Day Exploit	1.0

when used as these affect legitimate network users' access. They both lower slowly when not used to indicate that user access has been restored for the time being, and it can be beneficial to use them again in the future. The cost of the zero-day exploit will lower when used successfully (i.e. the network was not shut down) as using it increases the likelihood of its discovery and the eventual protection against it. The value of the opponent is the most important aspect as it denotes to the defender the importance placed on protecting the network against this specific opponent. Successful attacks, that are detected while the system is up and running, raise the value of the opponent quickly. No attacks detected slowly lower this value over time because the opponent has become less of a threat. Variable updates affect attacker play as well as defender play; however, the attacker utility values change according to how the attacker sees the game and not the defender's view. The experiments keep a separate view of the game for the attacker for the purpose of keeping the attacker type with the correct attack percentages. These variable changes provide a more apt simulation of a real-world scenario than restricting the game to the initial values.

The two remaining update types are used to affect strategy selection for the defender. The belief context update changes the player's belief context amounts according to the defender's subgame that the player believes the attacker is playing. When subgame utilities are updated because of variable changes, the corresponding belief context updates to the attacker's new MSNE values. Since the initial belief context values are based on the initial utility values, it is a reasonable conclusion that changes in the utilities should be reflected by updating the belief contexts. The g value changes independently of the belief contexts and affects the hyperstrategy selection for the defender. Values are entered in the XML game file for the increasing and decreasing amounts when a defender feels outmaneuvered. This check is made at each iteration for g adjustment value and it is applied before the next iteration. Games are run with these updates applied independently of one another and finally games are run with both types applied.

4.2 Performance Metrics

Game experiments save information from all iterations of the game in a comma separated values (CSV) file format where the header line indicates the data name separated by commas, and lines beneath contain the values. The data saved for each game run: the current utility values for that run for each player's strategy cross referenced with each opponent strategy, each row RMS (including their Expected Utility (EU), worst case utility (G), and Hypergame Expected Utility (HEU) values), the belief contexts (probabilistic values and column strategy percentages), each player's selected strategy with the utility it obtained, the row player's max and min strategy as compared to the opponents along with the utility it provides, and updatable values. The game data (utilities, RMSs, and belief contexts) is collected each round so that updates from one round to the next may be tracked. The game iteration data (strategies and utilities obtained, plus row player's max and min) is collected because it is the results of that round's conflict. The update values (variables, g) are saved to track any changes that may have occurred in these values per round. The data collected gives insights into changes that are occurring while the game iterations progress. All data is saved into a single file named after the game with *Results* tacked onto the end of it, (i.e. Nuisance Attacker vs Nuisance Defender Results#.csv). Multiple game run files are numbered in increasing order to differentiate between them.

In addition to the game round data each full run of game iterations collects final results data in a separate CSV file. In a similar fashion to the game round file, a header line names the data values that are located below it, but there is only a single data line entered for each collection of game iterations. All game run final results are saved into a single file with each run appearing on a successive line beneath the header. The data collected in this file is: number of iterations, row player statistics, and each player's strategy usage percentage. The number of iterations is an indication of how many executions there are for this line of data and the usage percentages keep track of how often each player used the

specific strategy. The row player statistics include: total max utility obtainable, total utility obtained, total min utility obtainable, utility ratio, and utility per game. The max and min utilities are an upper and lower bound of how well or poorly the player does if the best or worst strategies are always selected (when compared to the opponent strategy selection per iteration). The total utility obtained is what is actually achieved by the strategy selection choices made during the game iterations. The utility ratio is a measure of how the player fared on total utility versus the upper and lower bounds set by the max and min obtainable values. This ratio is an important factor when considering the success of the game run because the higher the value the closer row came to utility maximization. The last row statistic is the utility per game; it is the average utility per iteration and, when compared to the HEUs, is another indication of successful strategy selections. Finally, the percentage of use for each player's strategy is saved as the number of times used divided by total iterations of the game.

4.3 Evaluation Technique

The data that is collected is examined to provide evidence as to the effectiveness of strategy selection during the game. As was discussed momentarily, a high utility ratio is an indication of a strategy selection technique that is able to maximize potential utility because it more often picks the strategy that provides the highest utility versus the opponent strategy selection. A value of more than 50% is considered as successful because it has more often avoided the pitfalls of selecting poor competing strategies. The closer utility ratio is to 100% the better the defender's strategy selection. The average utility, utility per game iteration value, is also a measure of successful game play. The closer utility per game comes to the highest value HEU the better the defender's strategy selection. Although it is difficult to imagine that it could be often greater than the highest HEU for all hyperstrategies, certainly it is possible on any given run that a decent strategy selection system could outmaneuver an opponent enough times to create this opportunity. However,

any result that finishes above the initial R_0 full game HEU shows improvement over the initial NE expected outcome. Finally, total utility is taken into account although this can vary greatly from experiment to experiment due to the stochastic nature of strategy selection.

Since the initial game experiments are run without any update system in place (variable, g, or belief context), those results are compared to the update experiment results. The key is the improvement in the various data that is collected; when improvements are made it is an indication of a more robust defense strategy. The updates to the items involved are considered better utility ratios increase, utility per game iteration comes closer to or above the HEU, and when total utility values outdo their comparative defense types. The first update applied is the variable update, so the software simulation can reflect a more realistic scenario. These are compared with the base game results. Improvements to results are ranked in order of precedence with ratio being first priority, then utility per iteration, and lastly total utility. The ratio is most likely to show a good strategy selection because it maximizes available utility. In certain situations utility ratio can appear to be superb when utility per iteration and total utility are low. When this occurs it is an indication that the player has achieved the greatest utility, but that the player is limited by the outcomes available. Utility per iteration is important because it is an indication of how well the strategy selections are able to meet or exceed the HEU. Total utility is less of a concern because this value is game dependent; it is affected by the utility values present. Experiments are then performed with the variable updates intact with the g and belief context updates included separately. The final update tests include all update types together and is compared to the other update results determining if the changes, when made all together, are able to give the best overall performance.

4.4 Summary

The new software requires initial pre-experiment tests to ensure that it is working according to design and game theory principles. Basic game theory games are used to ensure that the initial stages of development are behaving correctly. Simple visual tests are performed via text to console showing that data from the XML game file is entered and retrieved correctly. PSNE and MSNE tests confirm that the NEs are being calculated correctly, with output validated by comparison to Hu's Applet and the Gambit software. A test suite performs a check of the function calculating code by comparing known results to the ones provided by the method calls. Functional utility values are also checked after being entered into the XML game file by visual check with game output utilities and Excel spreadsheet calculations. A validity check of the HNF form provided by the software is done to ensure calculations of required values for the experiments are correct.

Experiments are performed on an array of different attacker and defender types, with each being pitted against each other. Both are classified as: nuisance, low-level, mid-level, high-level, and all-out. These classifications reflect different levels of sophistication in defense and attack modes. Standard games are executed without any updates between game iterations. Update iterations adding changes in variables are compared to the initial game runs. Belief context updates and g value updates are coupled with the variable updates for comparison with the standard variable update execution. The games are run with g value update alone, belief contexts update alone, and both updates together. These results are compared to determine the update type which provides the best utility ratio, utility per iteration, and total utility.

Total utility is a worthwhile metric to examine because it is a measure of game success, but this can vary greatly between games and is best suited for comparing similar attacker/defender combinations. Utility per iteration is valuable because the higher the number the better total utility will be over successive iterations, and it can be compared

in a percentage way to the HEU values of the hyperstrategies. The utility ratio is the indicator that is considered the most influential because this value shows how close the player gets to the maximum utility available for that game; this calculated value is independent of game utility values and can be compared across games as a measure of success. These quantifiable metrics are the determining factors for the outcome of the experiments conducted.

V. HNF Analysis Tool (HAT) Software Experiment Results

The results of various experiment's performances are presented. Initially pre-experiment tests are performed in order to assure that the software is functioning as required for accuracy of the core experiments results. These core experiment results are provided and an analysis of their results is discussed. Additionally a summary of the experimental findings is given.

5.1 Pre-experiment Results

The hypergame analysis software developed for the core experiments is required to be validated so that the resulting data can be considered legitimate. These validations are performed incrementally as the software is written to assure that each step can build upon the previous work. The tests conducted are validated through various means, but each has a corresponding known good result to check against. Some of these results are from literature sources, while others are verified by mathematical analysis using software such as Microsoft Office Excel. These results show that the hypergame analysis tool has been developed in accordance with the principles of game theory analysis, section 2.1, and correctly emulates the HNF, section 2.3 on page 23.

Initial tests for game theory compliance involve the insurance that Extensible Markup Language (XML) data is entered and retrieved by the software correctly. Three relatively simple and common games are used to examine this: Prisoner's Dilemma (11), Chicken (12), and Matching Pennies. These games are used because of their two-player two-strategy models which keeps this stage of testing to a minimum, yet providing all the information required to ensure correct results. A "simple" text output is designed to show that the games are input and the able to be output correctly. In fact, the initial output did reveal that at first data was being retrieved from the software objects incorrectly because the row

player's utility values were being output second contrary to the common practice of putting them before the column player's utility. After code modification, all three games output the results as they are designed in their respective XML game files. The sample output (Figure 5.1) includes all three game outputs. The Prisoner's Dilemma reflects both its game grid, see Figure 2.1, and the its XML game file, see Figure 4.1. The passing of this test allows continuation of development to the calculation of the Nash Equilibrium (NE).

```
Game : Prisoner's Dilemma
                                Prisoner B
                          Stay Quiet        Confess
        Stay Quiet        -1.0,-1.0         -5.0,0.0
Prisoner A
        Confess           0.0,-5.0          -3.0,-3.0
```

```
Game : Chicken
                                Player2
                           Swerve          Straight
        Swerve             0.0,0.0         -1.0,1.0
Player1
        Straight           1.0,-1.0        -10.0,-10.0
```

```
Game : Matching Pennies
                                Oppose
                          Heads             Tails
        Heads           1.0,-1.0          -1.0,1.0
Match
        Tails           -1.0,1.0          1.0,-1.0
```

Figure 5.1: Example HAT game grid test output (Prisoner's Dilemma)

Selection of these games is also based upon the pure strategy Nash equilibrium (PSNE) that each provides, 12 Prisoner's Dilemma has a single PSNE, Chicken has two PSNEs, while the Matching pennies game has none at all. These games are used at this stage of testing because they are all basic game theory examples with well known PSNEs [37].

The variety of selections of PSNE provides confidence that the test indicates that the code manipulation of data is correct. These equilibriums are found using the simple technique of comparing the max utility value for each player's strategies where row and column match as maximum. The software does find the correct number of PSNEs as well as the proper strategies for all three of the test games and this is validated by the output, see Figure 5.2. This test along with the initial game grid output test shows that data is entered correctly from the XML game file and precise data retrieval is achievable. Calculation of the mixed strategy Nash equilibrium (MSNE) is now possible with the confidence that input data is correctly entered and maintained.

```
Game : Prisoner's Dilemma

Nash Equilibrium : Prisoner A:Confess,Prisoner B:Confess
```

```
Game : Chicken

Nash Equilibrium : Player1:Straight,Player2:Swerve
Nash Equilibrium : Player1:Swerve,Player2:Straight
```

```
Game : Matching Pennies

There are no pure strategy Nash Equilibriums.
```

Figure 5.2: HAT text output showing PSNEs of Prisoner's Dilemma, Chicken, and Matching Pennies

The testing of the portion of the software that is able to calculate the MSNEs is imperative because without this ability the hypergame software would not be able to analyze any game data correctly. The test makes sure that at least one MSNE is found because the Lemke-Howson algorithm (49) is designed to produce at a minimum one MSNE; although, the algorithm can produce up to the number of the row player's strategies

plus the number of the column player's strategies. Prisoner's Dilemma, Chicken, and Matching Pennies, and are also used in this test to show the software is operating within expectations. In addition, a sample non-zero sum game is input to ensure that this type of game also correctly calculates MSNEs.

Each game is first input into Gambit (section 3.3.1) and the MSNE is calculated using Gambit's recommended method. The sample non-zero sum game is shown in Figure 5.3. Then the games are manually input into the applet (section 3.3.2) and its calculations are compared with Gambit's, see Figure 5.4. Gambit outputs in fractions, but the applet outputs with decimal values; however the two agree on the amounts for each strategy. The applet values (rounded) read from left to right equal the Gambit fractional amounts for each player: row strategy 1 = 0.5572 = 185/332, row strategy 2 = 0.2761 = 275/996, row strategy 3 = 0.0 = 0, row strategy 4 = 0.1667 = 1/6, column strategy 1 = 0.2663 = 2256/8471, column strategy 2 = 0.0543 = 460/8471, column strategy 3 = 0.6794 = 5755/8471. Both software packages agree on the MSNEs for all games tested, so comparison is then made with the HAT software output. The software is able to calculate the MSNEs for all games tested and concurs with the results from both Gambit and the applet, see Figure 5.5. The successful test shows that changes incorporated for the HAT software made to Hu's applet Lemke-Howson algorithm code do not alter the MSNEs obtained. This confidence check is successful, so the software development testing continues with the functional utility requirement.

Properly calculated functional utility values are another requirement of the model, so its validation is a necessary step for completion of the software. Since the code for function calculation is obtained from an outside source, it is an important step to validate that it completes its task successfully. Thirty-nine different tests are performed as a test suite for this purpose, including some provided by the code designer [18]. Each test

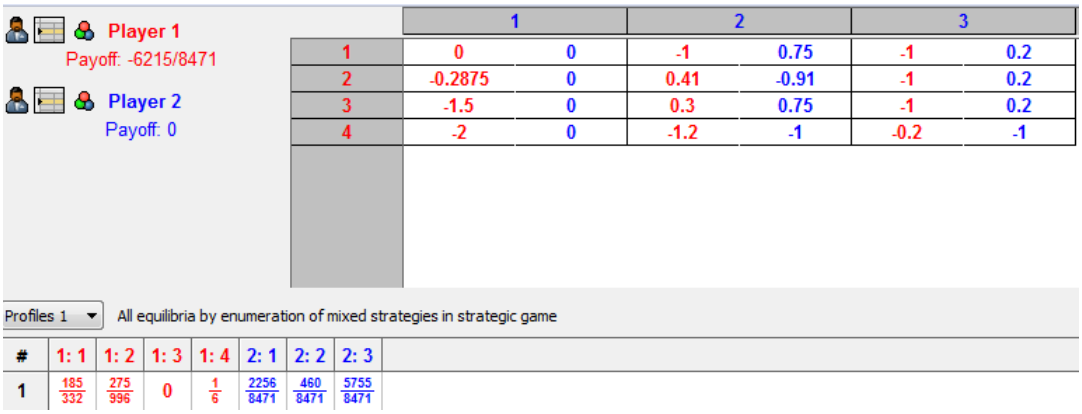


Figure 5.3: Gambit MSNE results for non-zero sum sample game

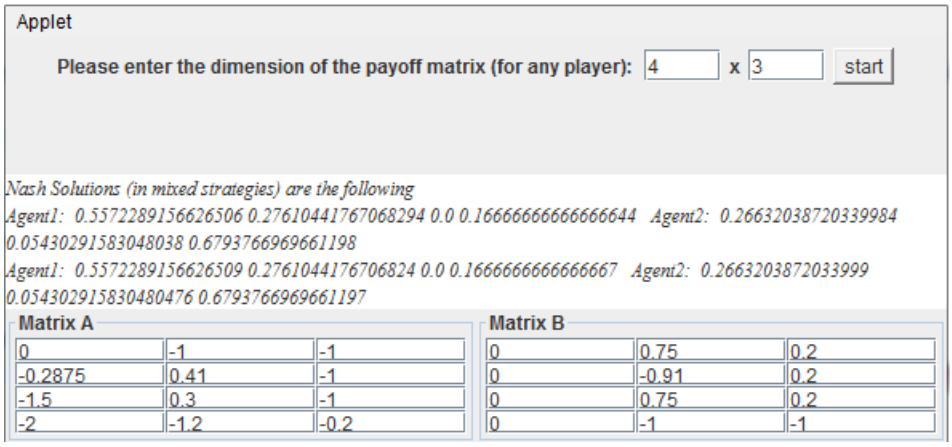


Figure 5.4: Applet MSNE results for non-zero sum sample game

consists of a function designed to test a specific operational outcome (addition, subtraction, multiplication, division, order of precedence, and parentheses overriding of precedence) as well as operations with zero and leading negative signs. Each test contains the proper result of the function tested; this result is compared to the answer given by the software for the same function. All tests in the suite use the supplied code and complete successfully, see Figure 5.6. However, this is only a portion of the testing done on the functional

```
Game : Sample Non-Zero Sum Game
                                          Player 2
                    Col 1                  Col 2                  Col 3
        Row 1       0.0,0.0                -1.0,0.75              -1.0,0.2

        Row 2       -0.2875,0.0            0.41,-0.91             -1.0,0.2
Player 1
        Row 3       -1.5,0.0               0.3,0.75               -1.0,0.2

        Row 4       -2.0,0.0               -1.2,-1.0              -0.2,-1.0

There are no pure strategy Nash Equilibriums.

Mixed Strategy Equilibrium:
Player 1: Row 1 0.5572, Row 2 0.2761, Row 3 0.0, Row 4 0.1667
Player 2: Col 1 0.2663, Col 2 0.0543, Col 3 0.6794
```

Figure 5.5: HAT MSNE results for non-zero sum sample game

utilities because they must be able to be entered with variable names and then translated into functions.

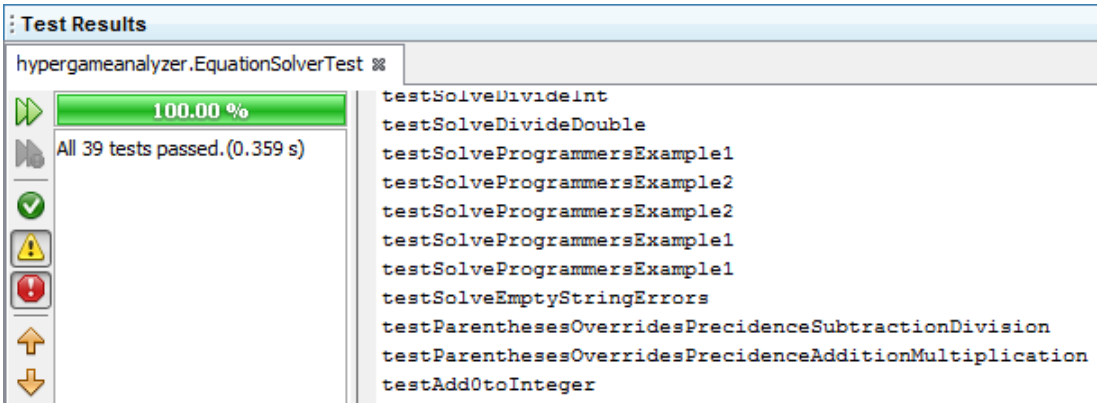


Figure 5.6: Function code test suite completion

Two example games are used to test this functionality, Hawk-Dove from Figure 2.4, 13, and the Chen and Leneutre model from Figure 3.2, 38. Each game is given two different

sets of initial values and entered into the software through their XML game files. A visual comparison with Excel spreadsheets versions of the games with functional utilities shows that they are able to be calculated successfully. The tests for Hawk-Dove check $V > C > 0$, which emulates the Prisoner's Dilemma (Figure 5.7, and $C > V > 0$, emulating the game of Chicken (Figure 5.8. The test done with Chen and Leneutre's model is a more complex utility function and works correctly (Figure 5.9). Once complete and functional utilities are confirmed to operate according to specification, the HNF is confirmed to be calculating required data correctly.

			CΣ	1.0	0.0
		1.0	C0	1.0	0.0
WS	MO	R0=FullGame	Row \ Column	Hawk	Dove
1.0	1.0	1.0	Hawk	2.5, 2.5	10.0, 0.0
0.0	0.0	0.0	Dove	0.0, 10.0	5.0, 5.0
2.5	**2.5**	**2.5**	**<-- EU**		
2.5	**2.5**	**2.5**	**<-- G**		
2.5	**2.5**	**2.5**	**<-- HEU**		

Nash Equilibrium : Row:Hawk,Column:Hawk

V	10			Player 2			
C	5			Hawk		Dove	
		Player 1	Hawk	2.5	2.5	10	0
			Dove	0	10	5	5

Figure 5.7: Hawk-Dove game $V > C > 0$ (HAT and Excel Output)

The HNF, as described by Dr. Vane in [52] and as discussed in 2.3 is the backbone of the HAT software and it is necessary that is built according to specification. It must calculate the required hypergame values: belief contexts (probabilistic value and Column Mixed Strategy (CMS)), Row Mixed Strategy (RMS), Expected Utility (EU), worst case utility (G), and Hypergame Expected Utility (HEU). An XML game file

			CΣ	0.0	1.0
		1.0	C0	0.0	1.0
WS	MO	R0=FullGame	Row \ Column	Hawk	Dove
1.0	1.0	1.0	Hawk	-2.5, -2.5	5.0, 0.0
0.0	0.0	0.0	Dove	0.0, 5.0	2.5, 2.5
5.0	**5.0**	**5.0**	**<-- EU**		
-2.5	**-2.5**	**5.0**	**<-- G**		
3.5	**3.5**	**5.0**	**<-- HEU**		

```
Nash Equilibrium : Row:Dove,Column:Hawk
Nash Equilibrium : Row:Hawk,Column:Dove
```

V	5			Player 2			
C	10			Hawk		Dove	
		Player 1	Hawk	-2.5	-2.5	5	0
			Dove	0	5	2.5	2.5

Figure 5.8: Hawk-Dove game $C > V > 0$ (HAT and Excel Output)

			CΣ	1.0	0.0
		1.0	C0	1.0	0.0
WS	MO	R0=FullGame	Attacker \ Defen...	Monitor	Not Monitor
1.0	1.0	1.0	Attack	0.3, -0.6	0.9, -1.0
0.0	0.0	0.0	Not Attack	0.0, -0.208	0.0, 0.0
0.3	**0.3**	**0.3**	**<-- EU**		
0.3	**0.3**	**0.3**	**<-- G**		
0.3	**0.3**	**0.3**	**<-- HEU**		

```
Nash Equilibrium : Attacker:Attack,Defender:Monitor
```

a	0.3						
Ca	0.1			Monitor		Not Monitor	
Cm	0.2		Attack	0.3	-0.6	0.9	-1
b	0.01		Not Attack	0	-0.208	0	0
Cf	0.8						
Wi	1						

Figure 5.9: Chen and Leneutre model game (HAT and Excel Output)

created with Vane's example 6.5 is loaded by the software for a visual check of its validity, see Figure 5.10. The software produces an output that is visually similar to that of the HNF and is compared to the example, see Figure 5.11. Although example 6.5 does not include the G and HEU calculations, a simple check by hand confirms the output is accurate. All other components that are calculated by the software are confirmed to be identical to the example within rounding error. This test gives confidence that the software can properly display and use the HNF form for use in running games. Experiments involving the designed experimental model are now able to be performed because the software operation is validated.

					C_Σ	.568	.130	.263	.039
			.8		C_1	.6	.15	.25	-
				.2	C_0	.4358	.0503	.3184	.1955
WS	PS	MO	R_1	R_0 = full game		col 1	col 2	col 3	col 4
.061	0	0 (-.555)	0	.3073	row 1	-2	-1	3	-2
.336	.35	0* (-.272)	.35	.2793	row 2	-1	4	-1	1
.349	.35	0* (-.272)	.35	.3408	row 3	2	-3	-4	1
.254	.3	0* (-.272)	.3	.0726	row 4	-2	-2	5	-5
-.289	-.272	-.272		-.358	EU(*, C_Σ)				
-.358	-.358	-.358		-.358	EU(*, C_0)				

Figure 6.5. Experiment 9, 4x3 subcase with WS hyperstrategy

Figure 5.10: Example 6.5 from [52] for confirmation of HNF calculations

5.2 Experiments Results

The experiment games are run using the execute file option which is a script that describes the various settings that can be used for the iterations. The execution file runs all the defender types against all the attacker types and in each update mode: none, variable

					CΣ	0.5672	0.1301	0.2637	0.0391
			0.8		C1	0.6	0.15	0.25	-
				0.2	C0	0.4358	0.0503	0.3184	0.1955
WS	PS1	MO	R1	R0=FullGame	Row \ Column	col 1	col 2	col 3	col 4
0.0615	0.0	0.0	0.4054	0.3073	row 1	-2.0, 2.0	-1.0, 1.0	3.0, -3.0	-2.0, 2.0
0.3359	0.35	1.0	0.2568	0.2793	row 2	-1.0, 1.0	4.0, -4.0	-1.0, 1.0	1.0, -1.0
0.3482	0.35	0.0	0.3378	0.3408	row 3	2.0, -2.0	-3.0, 3.0	-4.0, 4.0	1.0, -1.0
0.2545	0.3	0.0	0.0	0.0726	row 4	-2.0, 2.0	-2.0, 2.0	5.0, -5.0	-5.0, 5.0
-0.2888	**-0.2715**	**-0.2714**	**-0.3851**	**-0.3576**	**<-- EU**				
-0.7114	**-0.8**	**-1.0**	**-0.392**	**-0.3576**	**<-- G**				
-0.3733	**-0.3772**	**-0.4171**	**-0.3865**	**-0.3576**	**<-- HEU**				

Figure 5.11: Example 6.5 output displayed after loading by HAT software

only, variable with belief context, variable with *g* value, and all modes on. Each paring of attacker and defender is run for 10 separate executions with 100 iterations each. The pairings are also run in stochastic (random) and static (text file) attacker strategy selection forms. The HAT software saves each execution run in a file as well as putting overall statistics into a final results file per pairing. This creates a total of 2500 game runs (10 executions * 25 attacker/defender pairings * 5 update modes * 2 strategy selection forms) and a total of 250,000 iterations (2500 game runs * 100 iterations each). These executions take a total of 6 minutes on an Intel i7 M640 quad core 2.80GHz processor with 8GB RAM running Windows 7 Enterprise 64-bit. The data collected provides insight into the actions taken by the players during game play.

The defender's view of the game determines the initial setup of utility values, belief contexts, and hyperstrategies that make up the HNF game grid. Therefore, each defender, no matter which attacker it is paired with, begins the execution at the same starting values. This indicates that the defender is unaware of the attacker position and is expecting the type of attacker that it is set up to defend against. The starting defender game grids are: All-Out (Figure 5.12), High-Level (Figure 5.13), Mid-Level (Figure 5.14), Low-Level (Figure 5.15), Nuisance (Figure 5.16). As can be seen by these game setups, the higher threat levels create more escalated attacks; while the higher the alert level of the

defender, the more these attacks are expected. The most interesting observation is the differences in strategy selection between these starting positions. The All-Out and High-Level defenders choose the *shut-down* strategy, because the HEU for Modeling Opponent (MO) is highest. Mid-Level defender also chooses its MO, but strategy best suited for this defense is to *defend*. The Low-Level and Nuisance defenders prefer the full game view and use its *mixed strategy percentages* for strategy selection. The game executions with no updates keep these views of the game intact.

					CΣ	0.1667	0.0	0.8333
			0.8		C1	0.0	0.0	1.0
				0.2	C0	0.8333	0.0	0.1667
WS	PS1	MO	R1	R0=FullGame	Defender \ Attac...	Not Attack	Attack	Zero-Day Exploit
0.5	0.5	0.0	1.0	0.5	Not Defend	0.0, 0.0	-1.0, 0.7	-1.0, 1.0
0.0	0.0	0.0	0.0	0.0	Defend	-0.342, 0.0	0.3, -0.9	-1.0, 1.0
0.0	0.0	0.0	0.0	0.0	Provide Ruse	-1.0, 0.0	4.0, 0.7	-1.0, 1.0
0.5	0.5	1.0	0.0	0.5	Shut Down	-1.0, 0.0	3.0, -1.0	4.0, -1.0
1.1666	**1.1666**	**3.1665**	**-0.8333**	**1.1666**	**<-- EU**			
-0.5	**-0.5**	**-1.0**	**-1.0**	**1.1666**	**<-- G**			
0.8333	**0.8333**	**2.3332**	**-0.8666**	**1.1666**	**<-- HEU**			

Figure 5.12: Starting game model for All-Out Defender vs. Attackers

					CΣ	0.0	0.2	0.8
			0.8		C1	0.0	0.0	1.0
				0.2	C0	0.0	1.0	0.0
WS	PS1	MO	R1	R0=FullGame	Defender \ Attac...	Not Attack	Attack	Zero-Day Exploit
0.0	0.0	0.0	0.8494	0.0	Not Defend	0.0, 0.0	-1.0, 0.75	-1.0, 0.5
0.0	0.0	0.0	0.1506	0.0	Defend	-0.2875, 0.0	0.41, -0.91	-1.0, 0.5
1.0	1.0	0.0	0.0	1.0	Provide Ruse	-1.25, 0.0	2.75, 0.75	-1.0, 0.5
0.0	0.0	1.0	0.0	0.0	Shut Down	-1.5, 0.0	1.5, -1.0	2.5, -1.0
-0.25	**-0.25**	**2.3**	**-0.9575**	**-0.25**	**<-- EU**			
-1.25	**-1.25**	**-1.5**	**-1.0**	**-0.25**	**<-- G**			
-0.45	**-0.45**	**1.54**	**-0.966**	**-0.25**	**<-- HEU**			

Figure 5.13: Starting game model for High-Level Defender vs. Attackers

					CΣ	0.6929	0.3071	0.0
			0.8		C1	0.8661	0.1339	-
				0.2	C0	0.0	1.0	0.0
WS	PS1	MO	R1	R0=FullGame	Defender \ Attac...	Not Attack	Attack	Zero-Day Exploit
0.0	0.0	0.0	0.5294	0.0	Not Defend	0.0, 0.0	-1.0, 0.8	-1.0, 0.0
0.0	0.0	1.0	0.4706	0.0	Defend	-0.232, 0.0	0.5, -0.9	-1.0, 0.0
1.0	1.0	0.0	0.0	1.0	Provide Ruse	-1.5, 0.0	1.5, 0.8	-1.0, 0.0
0.0	0.0	0.0	0.0	0.0	Shut Down	-2.0, 0.0	0.0, -1.0	1.0, -1.0
-0.5787	**-0.5787**	**-0.0072**	**-0.166**	**-0.5787**	**<-- EU**			
-1.5	**-1.5**	**-1.0**	**-1.0**	**-0.5787**	**<-- G**			
-0.763	**-0.763**	**-0.2058**	**-0.3328**	**-0.5787**	**<-- HEU**			

Figure 5.14: Starting game model for Mid-Level Defender vs. Attackers

					CΣ	0.9017	0.0983	0.0
			0.8		C1	0.9017	0.0983	-
				0.2	C0	0.9017	0.0983	0.0
WS	PS1	MO	R1	R0=FullGame	Defender \ Attac...	Not Attack	Attack	Zero-Day Exploit
0.517	0.517	0.0	0.517	0.517	Not Defend	0.0, 0.0	-1.0, 0.85	-1.0, -0.5
0.483	0.483	1.0	0.483	0.483	Defend	-0.1755, 0.0	0.61, -0.91	-1.0, -0.5
0.0	0.0	0.0	0.0	0.0	Provide Ruse	-1.75, 0.0	0.25, 0.85	-1.0, -0.5
0.0	0.0	0.0	0.0	0.0	Shut Down	-2.5, 0.0	-1.5, -1.0	-0.5, -1.0
-0.0983	**-0.0983**	**-0.0983**	**-0.0983**	**-0.0983**	**<-- EU**			
-1.0	**-1.0**	**-1.0**	**-1.0**	**-0.0983**	**<-- G**			
-0.2786	**-0.2786**	**-0.2786**	**-0.2786**	**-0.0983**	**<-- HEU**			

Figure 5.15: Starting game model for Low-Level Defender vs. Attackers

					CΣ	0.9351	0.0649	0.0
			0.8		C1	0.9351	0.0649	-
				0.2	C0	0.9351	0.0649	0.0
WS	PS1	MO	R1	R0=FullGame	Defender \ Attac...	Not Attack	Attack	Zero-Day Exploit
0.5	0.5	1.0	0.5	0.5	Not Defend	0.0, 0.0	-1.0, 0.9	-1.0, -1.0
0.5	0.5	0.0	0.5	0.5	Defend	-0.118, 0.0	0.7, -0.9	-1.0, -1.0
0.0	0.0	0.0	0.0	0.0	Provide Ruse	-2.0, 0.0	-1.0, 0.9	-1.0, -1.0
0.0	0.0	0.0	0.0	0.0	Shut Down	-3.0, 0.0	-3.0, -1.0	-2.0, -1.0
-0.0649	**-0.0649**	**-0.0649**	**-0.0649**	**-0.0649**	**<-- EU**			
-1.0	**-1.0**	**-1.0**	**-1.0**	**-0.0649**	**<-- G**			
-0.2519	**-0.2519**	**-0.2519**	**-0.2519**	**-0.0649**	**<-- HEU**			

Figure 5.16: Starting game model for Nuisance Defender vs. Attackers

5.2.1 No Update Experiments.

The game executions without any updates are a basic case and are mostly akin to simple game theory models. In fact, four out of the five hypergames choose the hyperstrategy that is the same as the full game MSNE, with the Mid-Level defender being the lone difference. Thus these executions provide little in the way of surprise results. The defender that is designed to take on the equivalent type of attacker provides the best defense in terms of total utility obtained, utility per game, and utility ratio. Like shown in Figure 5.17 and Figure 5.18 where the Nuisance defender outperforms all the other defender types against the Nuisance attacker. Note the same totals achieved by Mid-Level, High-Level, and All-Out defenders due to the non-update game runs not changing strategy selection, where the remaining two defenders still use mixed strategy selection. These non-update games are not very realistic, however, because they do not take into account any changes in values that occur over time. After all, an All-Out defender chooses to shut down the entire system the whole time and that is certainly not the goal of anybody's network defense.

5.2.2 Variable Update Experiments.

The variable run games are when the games become more realistic because costs change as strategies are used more often and the value of the attacker can change a defender's view of the situation. In the case of High-Level and All-Out defenders, values drop in these initial variable update only tests, but that is expected because the attackers also change their tactics because of the variable changes. Although, these defenders still do best against their attacker equals. This drop in utility is especially true for the higher threat opponents who are unable to attack with *zero-day exploits* in every game iteration, see Figure 5.19 versus Figure 5.20, where total utility is no longer positive for any attacker type. There is no doubt that the variable updates make a difference in the outcome of the

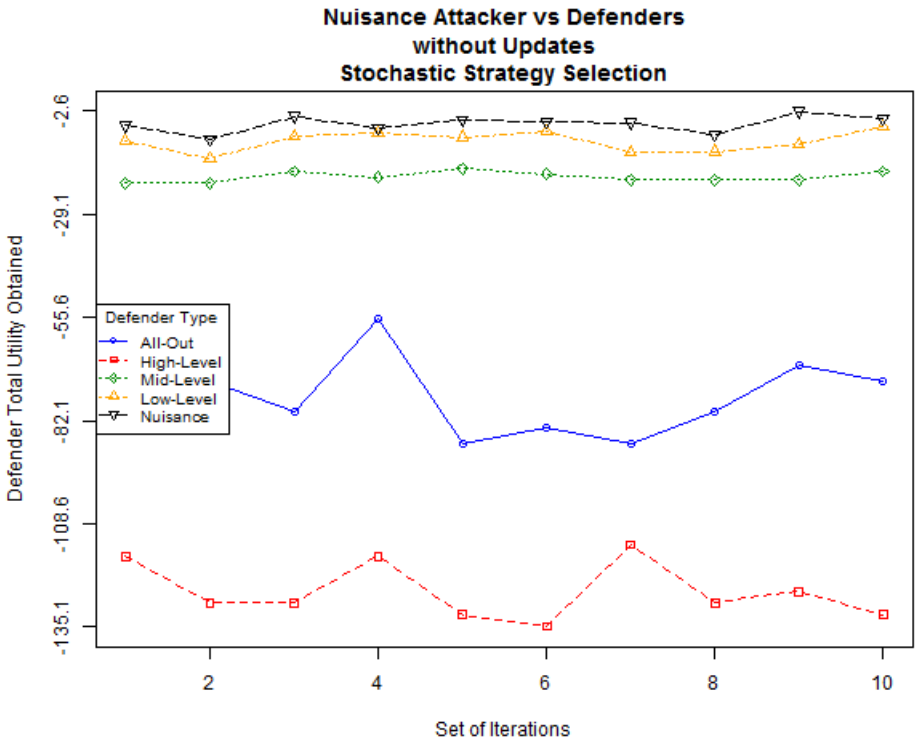


Figure 5.17: Defender total utility obtained vs. stochastic Nuisance attacker

game iterations. The question remains whether or not updates to the other portions of the HNF provide a better solution to the problem. This can be answered with the examination of the remaining experimental data involving belief context and g value updates.

5.2.3 Belief Context and g Value Update Experiments.

The further updates seek to gain an advantage over the variable updates by providing the hypergame with information gained during the game iterations. This begins to show the true power of applied hypergame theory to adapt a player's strategy choices to the situation that is presented. The belief context update changes what the player believes is the mixed strategy choices of the opponent in order to reflect the previous iteration opponent strategy. The g value update is a confidence check that changes when dependent upon whether or

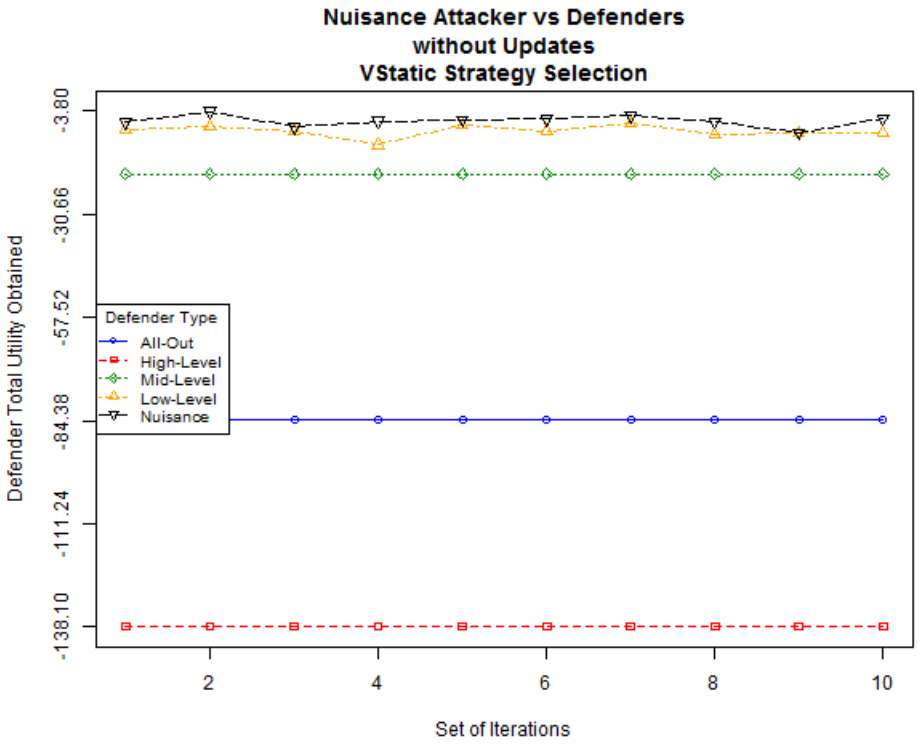


Figure 5.18: Defender total utility obtained vs. static Nuisance attacker

not the opponent outmaneuvers the player in the previous iteration. The updates to the mixed strategy values of the belief context and the HEU values because of *g* value changes lead to a reassessment of the hyperstrategy selection process. Evidence of these differences is shown in Figure 5.21, where the perfect result in utility per game from the no updates (which is unreasonable) is unattainable, but improvements with belief context or belief context and *g* value updates is achieved over variable updates alone.

5.3 Results Analysis

The results for the initial tests are able to confirm what is expected by the hypergame setup, that when the attacker is as the defender expects all the statistics are at or near maximum. Therefore the hypergame performs as intended by providing at worst the

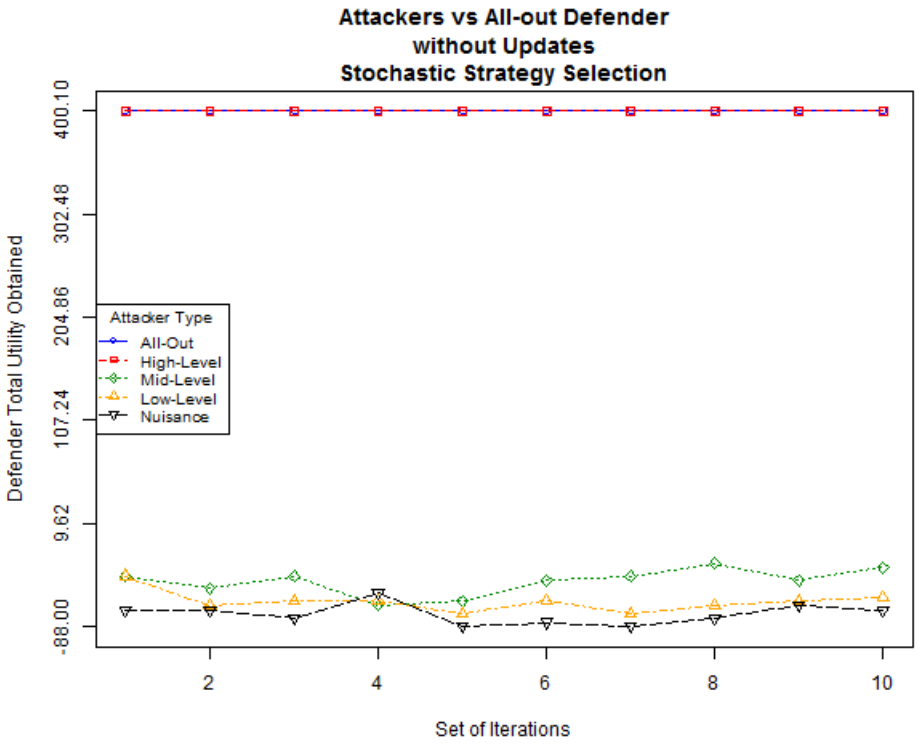


Figure 5.19: All-Out defender total utility obtained vs. stochastic attacker with no updates

expected NE when the actual game being played by both parties is realized. The results also indicate that the All-Out and High-Level defenders are equal in their utility ratios for all attackers because they have the same strategy choice. They also do poorly against the remaining attackers, but this is due to the cost of having the system shut down against a weaker opponent. The converse is also true, the lower level defenders, Mid-Level, Low-level, and Nuisance, are unable to defend against the upper attackers because the *zero-day exploit* strategy is always used against them. They are able to obtain high total utility, utility per game, and utility ratio against their counterpart attackers. The only reason these results are not completely beneficial is because over the iterations it is basically playing the same game over again. The lack of the updates does not provide a realistic scenario, for instance

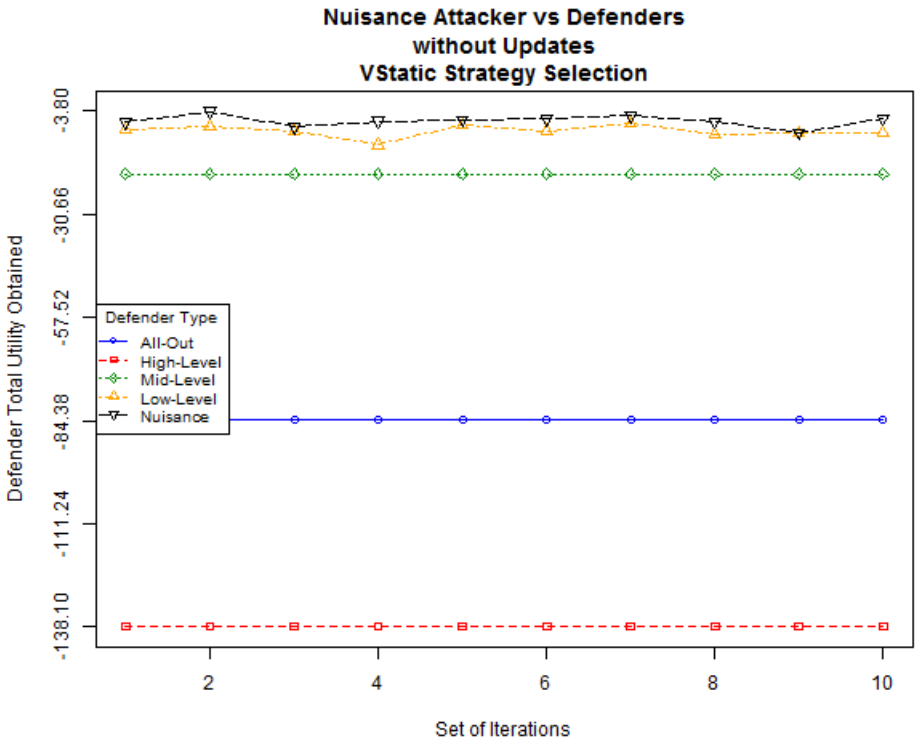


Figure 5.20: All-Out defender total utility obtained vs. stochastic attacker with variable updates

a *zero-day exploit* can hardly be used indefinitely and keeping the system shut down forever is not a winning strategy and certainly may not be a viable option in the first place.

The introduction of the variable updates begins to show how the hypergame strategy selection techniques can adapt to the adversary they face. The All-Out defender, who does poorly against the Nuisance attacker shows vast improvement when applying the variable updates in both the stochastic and static experiments (Figure 5.22). In fact, across the board defenders show improvements versus the attacker types they are not specifically designed to combat. Even when it appears there is a breakdown in total utility obtained (Figure 5.23), when the utility ratio is checked, the evidence indicates that this is not really the case. The game changes and there is actually improvement according to the utility ratio (Figure 5.24).

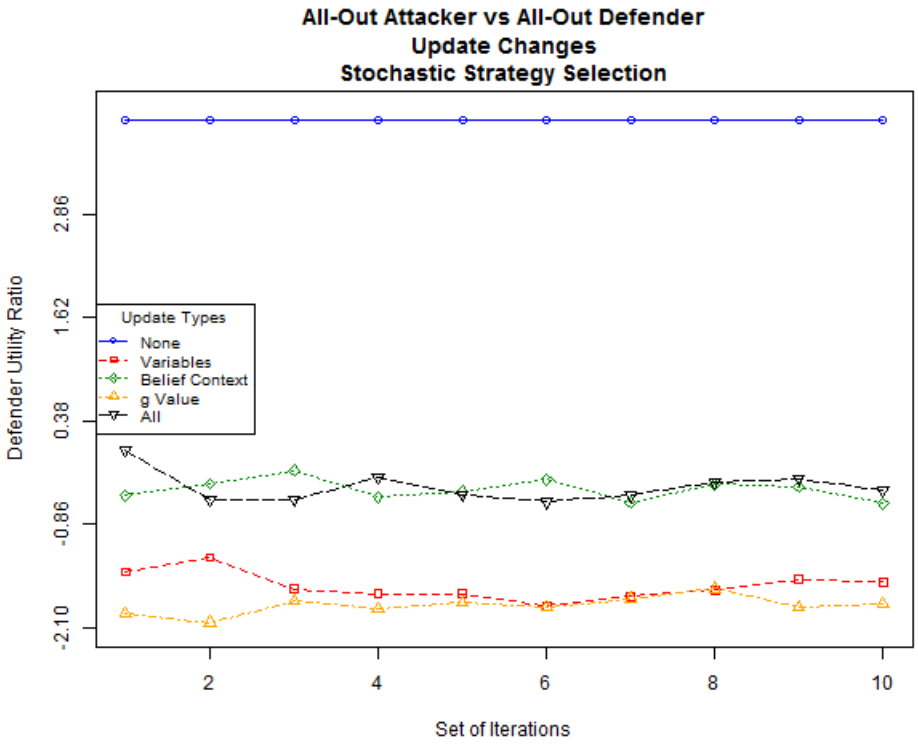


Figure 5.21: All-Out defender utility per game versus All-Out attacker for all update types

Therefore, total utility is a good indication of success, but utility ratio is better because it shows how the adaptations are able to keep utilities high during game fluctuations.

The results indicate that applying updates does improve the defender total utility, utility ratio, and utility per game. The functional utility values serve the purpose of being the catalyst for these improvements. Updating belief contexts also allows for improvements to those already gained with the variable updates. The *g* value updates are not as often effective at increasing the statistics examined and sometimes can be detrimental to the benefits gained by using the variable updates alone. The combination of both the belief context and *g* value updates is typically in line with just applying the belief context updates only; so there appears no statistical gain by adding the *g* value update. This indication

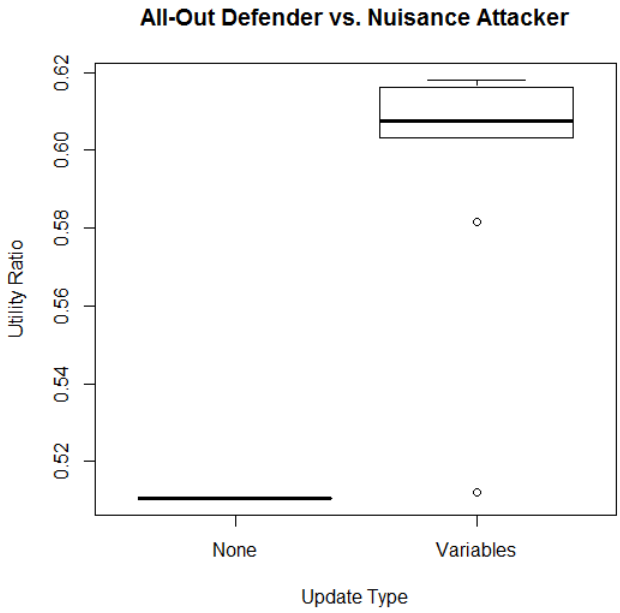


Figure 5.22: All-Out defender utility ratio improvement by adding variable updates

does not mean that the g value is not worthwhile to update, just that automated updates are unreliable. This value is a good candidate for human interaction at this level. The true benefit of the modeling technique is the addition of the variable updates, but the addition of the belief context updates adds to the adaptability of the defensive player. These results indicate that cyber defense systems, built upon hypergame theory principles, could apply these adaptive capabilities to increase their ability to thwart attacks.

5.4 Summary

The pre-experiment tests are performed to ensure that the developed customized software operates as required by the designed game model. Tests that XML game files are input correctly by the software are performed via text output. A check of PSNE discovery of these games also ensures that the data is stored correctly and is also manipulated accurately by the HAT. The portion of the software that produces the MSNEs is checked

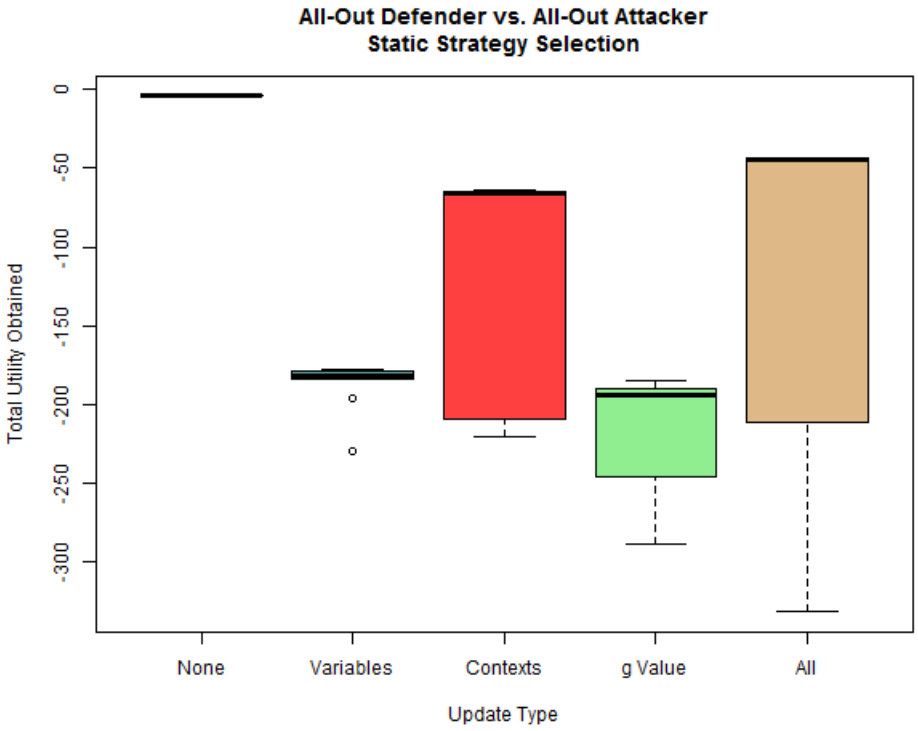


Figure 5.23: All-Out defender total utility obtained in static executions

via output; which is validated by both the Gambit game theory software and the original coded applet. Further testing is done to make sure that the functional utility portion of the code is able to produce the correct answers to given functions. This part of the software is also checked by providing XML game files with string based functions for utility values, showing that the software replaces variables in these functions with values provided. In addition to these tests, the display of the software's version of the HNF is compared with one of Dr. Vane's examples for validation that the components of the hypergame form are being calculated correctly. The completion of these tests allows the confidence that the experiments are performed within the expectations of the game model.

The experiments are executed with the HAT software via execution file that gives the software direction on which game file to run, how many iterations to perform, and

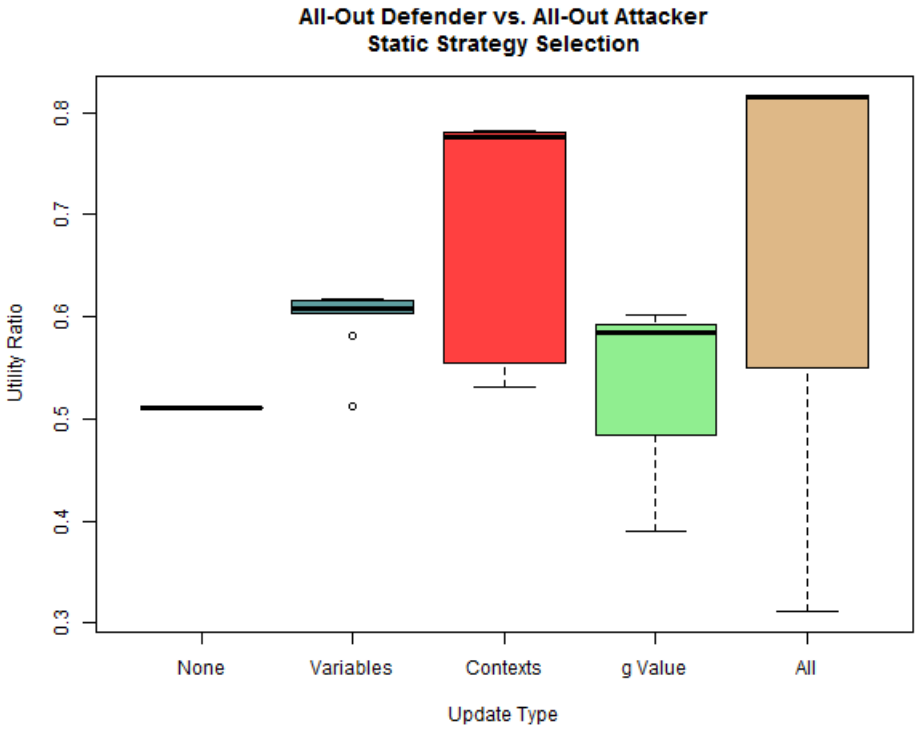


Figure 5.24: All-Out defender utility ratio indicating improvement in strategy selection

what kind of updates are required. 2,500 game runs, totaling 250,000, iterations are completed in this manner. The speed of the software and computer allow this to be done in a reasonable six minutes. The games are setup to pit all different styles of defenders against all the different styles of attackers. Each defender type views their starting position a little different from the others and this creates five special formulations of the initial hypergame. Games are run first without any updates; the results are similar to what is expected with standard game theory. Variable updates are next and provide a more realistic view of what occurs in an ongoing struggle for the network; strategies costs can change, attackers can be considered more or less dangerous. Finally updates to the defender's belief contexts and g values are included in order to help the defender learn more how the opponent is playing the game. This updates are applied to take advantage of the knowledge gained over the

course of previous iterations of the game and influence the strategy selection choices made by the software for the defender.

The analysis of the experiments confirms that the updates are effective at increasing utility ratio, utility per iteration, and utility per game. The results are an indication of the benefit of using functional utilities with variable updates and belief context revisions. Although the *g* value updates is not usually as effective as the belief context revisions, it can be considered a candidate for human decision making. Overall, the updates improve the defender's ability to select competitive strategies when faced with the variety of opponents. The adaptive ability of the hypergame model suggests that network defense systems can benefit from the strategy selection techniques used in these experiments. These findings provide the foundation from which improved and adaptable network defense systems can be built.

VI. Conclusions on Hypergame Theory Analysis

As stated most eloquently by Chris Inglis, National Security Agency deputy director, "It's almost impossible to achieve a static advantage in cyberspace whether that's a competitive advantage or a security advantage when things change every minute of every hour of every day" [17]. Therefore, cyber defense systems must have ways to adapt to the changing landscape of the conflict they face. Hypergame modeling and analysis is an avenue to this goal. The HAT software shows that hypergame theory when applied is able to achieve the adaptive nature required for today's cyber defense needs.

6.1 Findings

The innovative results discovered in this research effort are an indication that there is utility in the application of hypergame theory to the study of network defense techniques. The examination of the meta-level game, a game beyond what an opponent sees, provides an avenue in which one can get past the outcomes promised by the Nash Equilibrium (NE) alone. The hypergame strategy selection also mitigates risk by providing the means to include this fear of being outguessed into the equations that quantify the strategy selection process. The adaptability of network defense systems could be improved by applying some of the techniques learned through experimentation of hypergame models specifically designed around the defended system.

The experiments performed give initial evidence that there is the possibility for hypergame theory to provide a useful backbone to network defense systems. Inclusion of hypergame theory on the agent level would allow some autonomous actions to be performed; while inclusion at a higher level can give insight on an overall strategy to mitigate attacks. However, these findings do not disenfranchise the human element; on the contrary, they show that human administrators can use the evidence gleaned from

hypergame analysis to make more informed decisions about the types of attack they are facing. This gives hope that more powerful defense measures can be built that use the power of computation for analysis and the human mind for intuitive decision making.

Even though these results are for a designed specific cyber defense model, the results indicate that there is value in applying the principles of hypergame theory. A multitude of models can be designed according to specific networks or defense scenarios. The software is designed in a general fashion so that hypergames can be created that encompass a wide variety of situations, and does not necessarily have to be network defense related. Improvements over standard game theory results are possible using hypergame techniques. The functional utility values are able to model real world situations much better than static utility models. This is due to the fact that utility values can change and update according to factors that are provided to the model. The coupling of hypergame theory with the functional utility model creates a strategy selection technique that is more likely to adapt to changes in attack patterns. Incorporating these principles into cyber defense systems can make them more agile and allow less damage to be done to the networks they defend.

6.2 Impact and Action Recommendations

The goal of this research is to give reasons for the use of hypergame theory modeling for future cyber warfare. Game theory has been used to study cyber warfare in many different aspects and its continued study seems assured. However, little has been done the cyber warfare arena in the use of hypergames. This research shows that a given modeling technique can improve performance of defense strategy selection through the use of hyperstrategy selection. The need for greater computer security is only growing and one need only examine the Government Accountability Office *reported issues chart*, see Figure 1.1, or even check the daily news to get a feel for the situation. Attacks against computer systems are on the rise and show no signs of slowing down. The use of hypergame theory provides insight into better strategy selection techniques; these techniques, when

applied to defensive measures, can help to improve security. This study is the basis for several initiatives involving hypergame theory.

Modeling techniques are not limited to being as compact as the one in this study. Indeed the attack and defense strategies themselves are a lumped idea of the kinds that can be used. The model only provides one possible idea of the types of attacks that can be given their own strategy selection. Various selections of attacks are possible: probe attacks, DDOS attacks, etc. Similarly defense techniques can also be expanded to include direct defenses against the type of attacks that are encountered. The expansion of a game grid to include many more strategies does increase complexity and the time necessary to compute results. However, given that the current software runs 250,000 iterations on a 4x3 game grid while employing all update modules takes 6 minutes shows that increasing the complexity can still be accomplished. These results are achieved on a laptop with an Intel i7 M640 quad core 2.80GHz processor with 8GB RAM running Windows 7 Enterprise 64-bit; therefore, even more powerful machines can be used to increase efficiency when large complex games analysis is desired. There is also the option of parallelization between computers to divide the calculations up and perform them at the same time to increase performance for large game models. Increasing the complexity of the hypergame model can move it toward a more true to life scenario which in turn increases its usefulness.

The HNF Analysis Tool (HAT) software can be used as a tool for modeling real-world scenarios and running possible outcomes. Hypergames, based upon actual data of malicious network infiltrations, when entered into the software may provide data to improve cyber defense systems. The manipulation of functional utility values within the game and running iterations on these values can show how improvements can be made in defensive capabilities by using hypergame theory. The alteration of game models presented against the same attack characteristics could prove useful in the design of network security agents that have been upgraded through the lessons learned through experimentation. The

software could also be upgraded to implement an interactive mode for a user to play. In this mode a person could test their strategy selections versus recommendations by the software. This is useful for training of administrators on how a hypergame theory defense system would provide them with strategy choices.

Network security agents infused with hypergame theory decision making processes can provide valuable insight into what is taking place within their area of influence. Their strategy decisions can be passed along to centralized decision makers which examine the game from a higher perspective. Autonomous agents can have some strategy choices available to make on their own, but the centralized system would be able to choose overall strategies to combat the larger threat because it has a view of the larger hypergame. It can control the number and location of agents, including the type of defense agents since it knows where they are needed most. The use of hypergame theory for this decision making could provide greater return than even that available at a NE view of the game [45]. Hypergame theory can provide the maximization of defensive measures with the minimization of network interference. This is done by taking a standard game theory model and ensuring that the hypergame model at worst gets the utility available at the NE. Couple this with administrative support that can update the risk factors (i.e. the g value) when damage seems to be increasing and therein lies the makings of a next generation cyber defense system.

6.3 Software Expansion and Future Research

The software as written is complete to the point of giving this research the tool it required for analysis of the feasibility of hypergame theory applied to network defense. However, that is not to say that the software cannot be extended to incorporate different features. One of the core design principles used is open for extension and closed for modification [22]. This means that extendibility is built in allowing for new features to be added, but not at the price of having to modify existing code. The concept also includes

modularity so that, if required, old methods can be removed and new ones plugged in their place. In other words, even if the guts of how something is performed changes, the ins and outs can be kept the same. These concepts allow for the possibility of the following expansion ideas to be implemented without recoding the main structure of the software.

An interactive mode is suggested as a teaching tool. The software could allow a student to control the strategy selection with or without a suggestion provided. In this manner one could try to anticipate and discover the type of opponent that is playing against the user. A changeable g value slider would be an excellent way for the software to suggest which hyperstrategy that should be considered for the player's next move. This would be similar to that employed in the Security Policy Assistant (SPA) software [48], allowing the player to see how confidence in what is believed to be true affects the Hypergame Expected Utility (HEU). The interactive mode would also be a useful tool for model design because when moving slowly through game iterations one could see how changes are implemented in the game grid. This allows one to see where tweaks or different values may make the model more like the scenario that is attempting to be realized.

The HAT software currently supports operation for two-player games, but it does have the ability to be expanded to n-player. Cooperative games, in which agents of both defensive and offensive sides are considered as different players, can be considered as possible scenarios. Attackers can also be considered as different players even when keeping the defensive side as a single player. Rarely are all attacks a network is experiencing coming from the same source. Indeed, attacks may not even have the same goal and be from entities that are not cooperating or even have knowledge that the others are involved. This last scenario is an excellent candidate for hypergame analysis because it contains the core concept of misperception and incomplete information. Some changes would be required to incorporate these concepts into the HAT. The HNF form needs updated to show more

than two players, but this can be accomplished like Gambit (Figure 3.1, 35). The concept of multi-player games was considered as an expandability option during software design.

The software currently supports multiple player inputs in the XML, but most objects used within are generic to handle expansion. The naming conventions for opposing player strategies listed in the player objects would require updating. The update here would require each opposing player's strategy to be listed in the name so that cross referencing could be completed when a player's utility values are obtained. Also NEs are being determined with the Lemke-Howson algorithm which is only suitable for two-player bimatrix games [31]. This is due to the fact that a greater number of players removes the problem from linear complementarity because it is no longer bimatrix. Some suggested methods for *n*-player game NE calculation are simplicial subdivision, Govindan-Wilson, and non-globally convergent methods [2]. A method that follows a new process can be inserted in place of the Lemke-Howson algorithm. The modular design method used in the software development allows these changes to be implemented with limited impact on the other portions of the code.

One of the main functions of the HNF is that it examines the game as the row player views it, but not the column player. Column most certainly has a different view of the game which the HNF does take into account, but row may not have all the information about strategies available to column. This does appear counterintuitive because as a network defender the job is to minimize the loss occurring to the system. However, it may be advantageous at some times to see how minimizing column utility could affect strategy selection.

6.4 Summary

Achieving an adaptive strategy to combat cyber threats is what hypergame modeling and analysis is able to provide. The customized software HAT shows that this outcome is possible. It presents a foundation upon which future cyber defense initiatives

should be based. The results of this innovative study provide the initial evidence that hypergame analysis techniques can improve automated defenses as well as strategy selection recommendation to the humans that watch over networks. Computer network defense systems are required to be more agile and adapt the ever changing landscape which they are encountering; establishing hypergame theory based defense systems is one way to accomplish this goal. Future hypergame models should be expanded to more resemble real-world situations as closely as possible. These models can be designed and run with the HAT software for development and simulation. This software is the beginning of what should be a revolution in network defense techniques.

Bibliography

[1] "2007 cyberattacks on Estonia", 2013. URL http://en.wikipedia.org/w/index.php?title=2007_cyberattacks_on_Estonia&dir=prev&action=history.

[2] "Algorithms for Solving Two-Player Normal Form Games", April 2013. URL http://www.cs.cmu.edu/~sandholm/cs15-780S09/slides/Game%20theory%20lecture2-algs%20for%20normal%20form.pdf.

[3] "NetBeans IDE", April 2013. URL https://netbeans.org/.

[4] "Zero-day Attack", 2013. URL http://en.wikipedia.org/wiki/Zero-day_attack.

[5] Aigner, M. and M. Fromme. "A Game of Cops and Robbers". *Discrete Applied Mathematics*, (8):1–12, 1984.

[6] Alazzawe, Anis, Asad Nawaz, and Murad Mehmet Bayraktar. "Game Theory and Intrusion Detection Systems", 2006. ISA 767-Secure E-Commerce.

[7] Alexander, J. McKenzie. "Evolutionary Game Theory", 2009. URL http://plato.stanford.edu/entries/game-evolutionary/.

[8] Alpcan, Tansu and Tamer Başar. "An Intrusion Detection Game with Limited Observations". *In Proceedings of 12th International Symposium on Dynamic Games and Applications*. July 2006.

[9] Bandyopadhyay, Tridib and Reda Sebalia. "Countering Cyber Terrorism: Investment Models Under Decision and Game Theoretic Frameworks". *In Proceedings of the Southern Association for Information Systems Conference*. Richmond, VA, March 2008.

[10] Basset, Trebor. "Extensive Form Game 1", 2005. URL http://en.wikipedia.org/wiki/File:Extensive_form_game_1.JPG.

[11] Bennett, P. G. "Toward a Theory of Hypergames". *OMEGA*, 5:749–751, 1977.

[12] Bennett, P. G. and M. R. Dando. "Complex Strategic Analysis: A Hypergame Study of the Fall of France". *Journal of the Operational Research Society*, 30(1):23–32, 1979.

[13] Brumley, Lachlan. "HYPANT : A Hypergame Analysis Tool", November 2003.

[14] Burke, John. "Implementing Network Threat Protection". *Information Security Essential Guide*, February 2013.

[15] Chen, Lin and Jean Leneutre. "A Game Theoretical Framework on Intrusion Detection in Heterogeneous Networks". *IEEE Transactions on Information Forensics and Security*, 4(2), June 2009.

[16] Collier, Mike. "Estonia: Cyber Superpower", December 2007. URL http://www.dcestonian.com/estonews/articles/07/cyber1217.htm.

[17] Corrin, Amber. "Is government on the wrong road with cybersecurity?" URL http://fcw.com/articles/2013/05/21/csis-cybersecurity.aspx?s=fcwdaily_220513.

[18] Edwards, Alex. "Simple Equation Solver".

[19] Farivar, Cyrus. "Cyberwar I : What the attacks on Estonia have taught us about online combat", May 2007. URL http://www.slate.com/articles/technology/technology/2007/05/cyberwar_i.html.

[20] Fisher, Max and the Washington Post Foreign Staff. "The U.S. weapons systems that experts say were hacked by the Chinese", May 2013. URL http://www.washingtonpost.com/blogs/worldviews/wp/2013/05/28/the-u-s-weapons-systems-that-experts-say-were-hacked-by-the-chinese/.

[21] Fowler, Martin. "Refactoring Home Page", 2013. URL http://refactoring.com/.

[22] Freeman, Eric, Elisabeth Freeman, Kathy Sierra, and Bert Bates. *Head First Design Patterns*. O'Reilly Media, 1005 Gravenstein Highway North, Sebastopol, CA 95472, 1st edition, October 2004.

[23] He, Fei, Jun Zhuang, and Nageswara S. V. Rao. "Game-Theoretic Analysis of Attack and Defense in Cyber-Physical Network Infrastructures". G. Lim and J.W. Hermann (editors), *In Proceedings of 2012 Industrial and Systems Engineering Research Conference*. 2012.

[24] Hipel, Keith W., Muhong Wang, and Niall M. Fraser. "Hypergame Analysis of the Falkland/Malvinas Conflict". *International Studies Quarterly*, 32:335–358, 1988.

[25] House, James Thomas and George Cybenko. "Hypergame Theory Applied to Cyber Attack and Defense". Edward M. Carapezza (editor), *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense IX*, volume 7666. 2010.

[26] Hu, Junling. "Systems and Papers", 2013. URL http://junlinghu.com/papers.html.

[27] Huxham, C. S. and P. G. Bennett. "Hypergames and Design Decisions". *Design Studies*, 4(4), October 1983.

[28] Igure, Vinay M., Sean A. Laughter, and Ronald D. Williams. "Security Issues in SCADA Networks". *Computers and Security*, 25:498–506, March 2006.

[29] Jormakka, Jorma and Mölsä. "Modelling Information Warfare as a Game". *Journal of Information Warfare*, 4(2):12–25, 2005.

[30] Khan, Mohammad Emtiyaz. "Game Theory Models for Pursuit Evasion Games", 2007. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.89.7157. Department of Computer Science University of British Columbia.

[31] Lemke, C. E. and J. T. Howson. "Equilibrium Points of Bimatrix Games". *SIAM Journal on Applied Mathematics*, 12(2):413–423, 1964.

[32] Lyman, Robert K., Lieutenant Colonel USAF. "Decisive Use of IO". *IOSphere*, 2007.

[33] Manshaei, Mohammad Hossein, Quanyan Zhu, Tansu Aplcan, Tamer Başar, and Jean-Pierre Hubaux. "Game Theory Meets Network Security and Privacy". *ACM Transactions on Computational Logic*, September 2010.

[34] Mckelvey, Richard D., Andrew M. Mclennan, and Theodore L. Turocy. "Gambit: Software Tools for Game Theory", 2010. URL http://www.gambit-project.org.

[35] McMillan, Robert. "Siemens: Stuxnet Worm Hit Industrial Systems", September 2010. URL http://www.computerworld.com/s/article/print/9185419/Siemens_Stuxnet_worm_hit_industrial_systems?taxonomyName=Network+Security&taxonomyId=142.

[36] Nash, J. F. "Equilibrium Points in *n*-Person Games". *In Proceedings of National Acamy of Sciences*, 48–49. 1950. URL http://www.pnas.org/content/36/1/48.full.pdf+html.

[37] Osborne, Martin J. *An Introduction to Game Theory*. Oxford University Press, 2004.

[38] Otrock, Hadi, Mona Mehrandish, Chadi Assi, Mourad Debbabi, and Prabir Bhattacharya. "Game Theoretic Models for Detecting Network Intrusions". *Computer Communications*, (31):1934–1944, January 2008.

[39] Pritchard, David. "The Lemke-Howson Algorithm", March 2013. URL http://www.public.iastate.edu/~riczw/MEGliter/general/Lemkehowson.pdf.

[40] Quin, Liam. "Extensible Markup Language (XML)", 2013. URL http://www.w3.org/XML/.

[41] Roy, Sankardas, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, Vivek Shandilya, and Qishi Wu. "A Survey of Game Theory as Applied to Network Security". *In Proceedings of the Hawaii International Conference on System Sciences*. 2010.

[42] Sandler, Todd and Kevin Siqueira. "Games and Terrorism: Recent Developments", 2009. URL http://research.create.usc.edu/nonpublished_reports/84.

[43] Shen, Dan, Genshe Chen, Jose B. Cruz Jr., Leonard Haynes, Martin Kruger, and Erik Blasch. "A Markov Game Theoretic Data Fusion Approach for Cyber Situational Awareness". Belur V. Dasarathy (editor), *Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2007*, volume 6571. SPIE, 2007.

[44] Shiva, Sajjan, Sankardas Roy, and Dipankar Dasgupta. "Game Theory for Cyber Security". *In Proceedings of the Cyber Security and Information Intelligence Research Workshop*. April 2010.

[45] Sickendick, Karl, Captain USAF. "Intrusion Detection System Allocation at the Nash Equilibrium", 2012. Department of Electrical and Computer Engineering, Professor Gary Lamont.

[46] Singh, Anshuman, Arun Lakhotia, and Andrew Walenstein. "Malware Antimalware Games". 2010.

[47] Spaniel, William. "Hawk-Dove Game", March 2013. URL http://gametheory101.com/Hawk-Dove_Game.html.

[48] Vane, Russell, IV. *Veridian Security Policy Assistant Users Manual*, July 2002.

[49] Vane, Russell R. "Hypergame Theory for DTGT Agents". *American Association for Artificial Intelligence*, 2000.

[50] Vane, Russell R., III. "Planning for Terrorist-Caused Emergencies". F. B. Armstrong M. E. Kuhl, N. M. Steiger and J. A. Joines (editors), *In Proceedings of the 2005 Winter Simulation Conference*. 2005.

[51] Vane, Russell R., III. "Advances in Hypergame Theory". *International Journal of Autonomous Agents and Multi-Agent Systems*, 2006.

[52] Vane, Russell Richardson. *Using Hypergames to Select Plans in Competitive Environments*. Ph.D. thesis, George Mason University, 2000.

[53] Walker, Paul. "A Chronology of Game Theory", September 2012. URL http://www.econ.canterbury.ac.nz/personal_pages/paul_walker/gt/hist.htm.

[54] Wang, Muhong, Keith W. Hipel, and Niall M. Fraser. "Modeling Misperceptions in Games". *Behavioral Science*, 33:207–223, 1988.

[55] Wang, Muhong, Keith W. Hipel, and Niall M. Fraser. "Solution Concepts in Hypergames". *Applied Mathematics and Computation*, 34:147–171, 1989.

[56] Wilshusen, Gregory C., Directory Information Security Issues. *CYBERSECURITY: A Better Defined and Implemented National Strategy Is Needed to Address Persistent Challenges*. Technical report, United States Government Accountability Office, March 2013.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
13–06–2013	Master's Thesis	Jan 2012–Jun 2013

4. TITLE AND SUBTITLE

Applied Hypergame Theory for Network Defense

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Gibson, Alan S., USAF

5d. PROJECT NUMBER

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-13-J-02

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Cyber operations are the most important aspect of military conflicts in the 21st century, but unfortunately they are also among the least understood. The continual battle for network dominance between attackers and defenders is considered to be a complex game. Hypergame theory is an extension of game theory that addresses the kind of games where misperception exists, as is often the case in military engagements. Hypergame theory, like game theory, uses a game model to determine strategy selection, but goes beyond game theory by examining subgames that exist within the full game. The inclusion of misperception and misinformation in the hypergame model shows the usefulness of the theory because it provides strategy selection techniques that help mitigate risks. This goal of this research is to provide a foundation for the use of hypergame theory for adaptable cyber defense systems able to adjust to the existence of misperceptions and misinformation. The results discovered in this research effort are an indication that there is utility in applying hypergame theory to the study of network defense techniques. This innovative research indicates that the application of hypergame theory is able to achieve the adaptive nature required for today's cyber defense needs.

15. SUBJECT TERMS

Cyber warfare, game theory, hypergame theory, network defense

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. Gary Lamont (ENG)
U	U	U	UU	122	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x4718 gary.lamont@afit.edu