Cyber Fighter Associate:

A Decision Support System for Cyber Agility

Charles Huber, Dr. Patrick McDaniel
The Pennsylvania State University
State College, PA
cjh5466@cse.psu.edu

Scott E. Brown
Secure Mission Solutions
2457 Aviation Ave, Suite 200
North Charleston, SC 29406
scott.e.brown96.ctr@mail.mil

Dr. Lisa Marvel
U.S. Army Research Laboratory
APG, MD 21005
lisa.m.marvel.civ@mail.mil

Abstract— **In the event of a cyber attack it is important for network defenders to make quick, informed decisions to secure assets. However, the human decision making process is slow and inefficient compared to the speed at which cyber operations can occur. The use of a decision support system (DSS) would help aid agility decisions to shorten the amount of time a network is insecure. In tactical military networks, such a DSS would need to consider constrained resources such as battery life and bandwidth as well as mission goals. In this paper we describe a DSS (Cyber Fighter Associate (CyFiA)) to help select agility maneuvers to contain and eliminate a malicious infection in a mobile ad hoc network (MANET). A variety of scenarios prioritizing factors such as node criticality, health, security and capability are employed. Our results show that CyFiA selects the best sequence of maneuvers for the scenarios and can reduce energy costs when securing a MANET.**

Keywords—Decision Support; Security; Cyber Agility

I. Introduction

In the event of a cyber attack it is important for network defenders to make quick, informed decisions to secure assets. A defender often must analyze massive amounts of data to determine the type of attack, where the network is compromised, as well as the course of action to restore the network back to the desired security level. The process of accurately analyzing large amounts of data and taking action based on data analysis takes a long time for humans. In addition, the process can be stressful and human biases can cause poor decisions [1]. In a military environment, agility decisions become even more important as information and services can be vital to an operation and impact soldiers' lives. Therefore, it is necessary to find a way to expedite the decision making process while still making informed choices in the event of a cyber attack.

Decision Support Systems (DSS) are computer applications that help make important, informed decisions based on large amounts of data [1]. As a precedent, the Pilot's Associate [2] was a decision support application that helped fighter pilots make decisions while flying. The Pilot's Associate consisted of four subsystems. Two subsystems analyzed the outside environment and two subsystems planned actions based on information gathered from the environment. The planning systems reacted to threats in a way that supported a predefined mission plan. Additionally, there is the Warfighter Associate [3]. The Warfighter Associate is DSS that aids troops and mission command in obtaining and analyzing intelligence, surveillance, and reconnaissance (ISR) assets. Many other systems are currently under development. Here we propose the use of a DSS in the cyber domain to help network defenders make informed cyber agility decisions to secure a network.

To show the plausibility of a DSS in the cyber domain, we devised an experiment in which a DSS (CyFiA) makes agility decisions to secure a small tactical network of ten nodes. Our experiment is designed to follow the framework outlined in [4].

In this agility evaluation framework, several metrics were established to guide the selection of agility maneuvers to accomplish a primary mission (security a critical path) as well as a secondary mission of securing the entire network. An NS-3 network simulation was leveraged to simulate a mobile ad hoc network. The framework uses an exhaustive search to evaluate all possible maneuver sequences, then selects the best in terms of mission priorities (e.g., maintain capability over security). However, cyber defenders cannot wait until the mission is over to decide the best sequence of maneuvers in these dynamic networks and a more proactive decision aid is needed.

CyFiA provides this decision support and takes into account network and node information such as resource constraints, health, capability, and security when making decisions. At this time CyFiA's agility maneuver set uses those described in [4]. Our results show that CyFiA chooses maneuvers to secure and critical path and then the entire network in a variety of scenarios while keeping agility maneuver cost to a minimum.

The remainder of the paper is organized as follows. We introduce the concepts CyFiA and the software used to implement it followed by the simulation procedure. We then describe the potential maneuvers for two scenarios, securing a critical path and then security the remaining network nodes using different priorities. Experimental results from the simulations are then described. Finally, we conclude with a summary of our findings and propose future work.

II. Cyber Fighter Associate

The CyFiA is a DSS that uses the Solomon Engine by Veloxiti Inc [5, 6]. Decision Support Systems are comprised of three components: the database management system (DBMS), the model-base management system (MBMS), and the dialog generation and management system (DGMS). The DBMS holds large amounts of information gathered from the environment. The MBMS transforms the data held in the DBMS into something useful for decision-making. The MBMS may hold multiple models to solve multiple classes of problems. The DGMS provides insight to the user in the form of a user interface. The Solomon Engine supplies a solid MBMS and DGMS, but leaves the DBMS design and implementation up to the creator. Since there are so many possible database designs, we will not touch on CyFiA's DBMS implementation.

A. *CyFiA Model Base Management System*

The Solomon Engine is a cognitive engine that makes decisions based on Boyd's observe-orient-decide-act (OODA) loop model. The OODA loop has been accepted as a way to model decision-making in military environments. It is comprised of four steps.

Observe – A user interacts with an environment, usually by using sensors or querying for information.

Orient – The data observed is organized and filtered by the user to attain some situational awareness and understand the current state of the environment

Decide – The user uses information derived from the orientation to prioritize plans and goals.

Act – The user selects and carries out a plan to satisfy some goal

To implement the OODA loop model, the Solomon Engine divides the decision making process into two graphs: the Observe-Orient graph (OOG) and the Decide-Act graph (DAG). The graphs are both coded using a domain specific language (DSL) that has strong ties to the Java programming language. The nodes of the graphs are represented by "*.kb" files that are compiled into Java classes. The Solomon Engine (herein referred to as "the engine") and all its parts are implemented in the Eclipse Kepler IDE.

The OOG and DAG use different data structures, but it is important to note that the structures that make up both graphs contain similar functionality. The DSL that defines the syntax and semantics of these structures accepts many basic variable types including integer, float, double, string, and character. Values are copied between nodes in the graphs, so structures such as lists and queues need to be carefully handled, but Veloxiti provides a data type of "immutable list" that can be passed from node to node or graph to graph without warnings. On top of normal type declarations, values within nodes require another identifier of either "key" or "attribute". Keys are vital to node creation. The engine will not create an instance of a node until all keys defined in the node definition have values. Attributes, however, are optional and do not need to be assigned values at creation time.

B. *Observe-Orient Graph*

The OOG is used to filter data gathered from sensors. It uses data structures called beliefs to hold and pass information. Belief nodes are the connections from the sensors in the environment to the engine. Normally, beliefs have at least one key, but it is possible to have a belief with no keys.

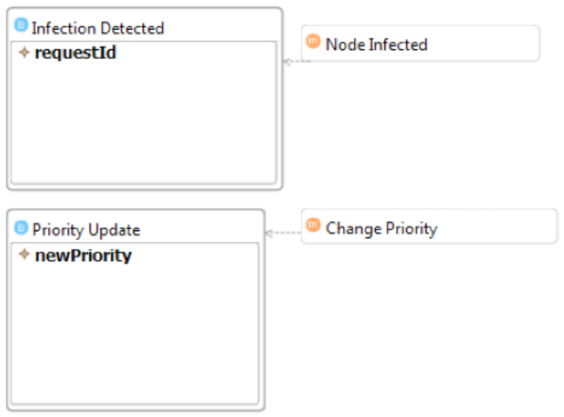

Fig. 1. CyFiA OOG

Otherwise, information can be passed from belief to belief via links.

Links are structures for sharing data between node instances in the same graph. All links have a parent and a child node and have provided methods for updating child node attributes and keys. Links also have a constraint method that can restrict the update of a child if a predefined condition is not met.

C. *Decide-Act Graph*

The DAG is a decision tree composed of data structures called plans and goals. Plans and goals follow two basic rules:

> In order to be satisfied, all of a plan's child goals must be satisfied. Any of a goal's child plans should be sufficient for satisfying the goal. [6]

In addition, goals can only be children to plans and plans children to goals.

Plans are especially important in the DAG. Plan nodes are where proposals and actions live. Proposals and actions are the way the engine communicates its decisions to the outside world. A proposal is simply a suggestion of what a user should do given the current state of the environment. The engine may give multiple proposals at once with the expectation that the user will act on one of them. Actions are automated decisions the DSS will carry out to change the environment without user interaction.

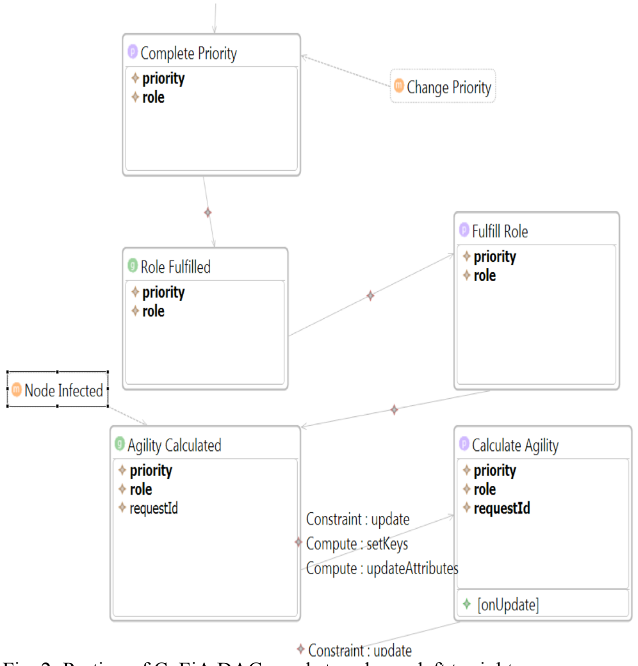

Fig. 2. Portion of CyFiA DAG, reads top-down, left to right

In order for the OOG and DAG to communicate, special links are needed called monitors. Monitors have the same basic functions as the link class, however it is possible to output alerts from a monitor to give information to a user. Fig. 1 shows the OOG and the monitor connections to the DAG in Fig. 2.

III. CyFiA Implementation

The Cyber Fighter Associate uses a simple implementation of the Solomon Engine knowledge base as well as a database containing information about the simulation network to make network agility decisions. The goal of these agility decisions is to prevent malicious software from propagating across the network and eliminate it. Since our experiment was specific to a single attack, CyFiA's OOG is rather basic, specifically looking for infected (malicious) nodes or newly immune (patched) nodes. The OOG consists of two singleton beliefs and two monitors as shown in Fig. 2. The first belief is used to filter information in the database and begin agility decision-making. The second belief is to change agility priorities during the simulation. CyFiA's DAG would determine which node to perform agility on and which agility maneuver to perform. The determination of which node to perform an agility maneuver upon is calculated based on node security, energy cost, and capability. Each of these calculations finds nodes with maximum security, capability or minimum energy cost in the network.

Clearly the order in which agility calculations are carried out has an effect on the node that is subjected to them. Therefore CyFiA allows a priority to be set that determines the order that the calculations will be done. This priority is initially set when the engine is created and can be changed real-time to alter agility calculation order to reflect priorities. Additionally, CyFiA allows for an overall mission priority where nodes that are critical to the current mission (critical nodes) are considered for agility first.

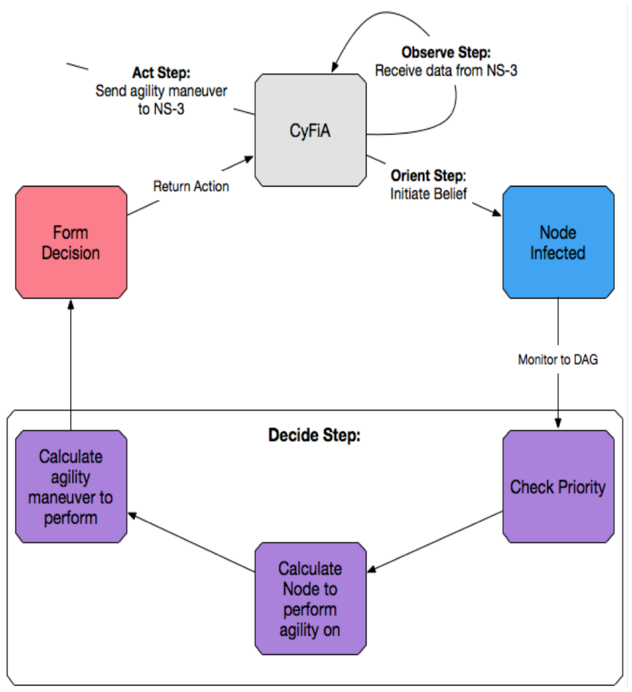

Fig. 3. CyFiA OODA-loop Implementation

After a node has been selected for agility, CyFiA chooses one of four maneuvers (same as those used in [4]):

Patch – patches the selected node with a patch from another node, making it immune to the malicious software.

Healing – The selected node rewrites its code to fix the vulnerability the malicious software will exploit.

Quarantine – The selected node will stop using the function or functions the malicious software will exploit.

Blocking – The selected node will sever a connection or connections in the network cutting off communication with other nodes and preventing the malicious software from spreading.

Currently, we lack an appropriate way to distinguish how these agility maneuvers should be selected. CyFiA selects a maneuver based on the battery life of the selected node, but the decision to implement CyFiA this way was arbitrary and can be modified. We later will see that due to simulation battery values, the patching agility maneuver is always selected. Also, only one agility maneuver is selected at a time, which ensures that CyFiA will not continuously send the same agility maneuver thus allowing the simulator to update node information. Fig. 3 shows a high level look of CyFiA's decision-making process.

IV. Simulation Procedure

To test the basic concepts of a DSS in performing cyber agility, CyFiA was used as part of a network simulation. The (NS-3) network simulator propagates malicious software through a small tactical network of ten nodes. All nodes begin in identical state (health, capability, battery budget, etc.).

A. Initialization

The simulation is composed of three separate entities: NS-3, a publish-and-subscribe server (PASS), and CyFiA. The PASS acts as an intermediary between the simulator and CyFiA. While a PASS is not necessarily needed in this scenario, it is useful to use with a DSS as it helps group information from many different sources (often called producers) and update clients (consumers) with new information. In our experiment, NS-3 is the only producer and CyFiA is the only consumer. We decided to send data directly from CyFiA to NS-3 instead of back through the PASS to keep information flow one way.

B. Assumptions

We make some fundamental assumptions to ensure the simulation runs smoothly. Firstly, we assume that CyFiA omniscient: it has all information about every node and the network graph prior to receiving its first infected node. To satisfy this assumption, NS-3 sends this information to CyFiA prior to the simulation starting. Secondly, we assume that a patch for the vulnerability being exploited exists and that some node within our network has that patch. This is not a requirement for CyFiA to operate correctly (if a patch does not exist, the patching agility maneuver will not be selected).

C. Procedure

Our experiment runs in a turn-based style system. Information is created and sent to the PASS by NS-3. The PASS then forwards this information to CyFiA. CyFiA makes an agility decision and sends that decision back to the NS-3 so new information can be created. The NS-3 uses a queue structure for simulation updates, so the simulator may queue its own propagation events before an agility event, even though the agility event was sent before the propagation event was created. It is necessary to pause NS-3 while CyFiA makes its decision and then un-pause the simulator once an agility maneuver has been queued.

As stated in our assumptions, the first set of information sent is all the attribute information for every node and the network graph. Once the simulation begins, i.e. the first infected node update is received, CyFiA will make an agility decision, send that decision back to NS-3. NS-3 will then update the simulation, including malware propagation, and then send new information to CyFiA. There is an 80% chance of malware propagation on every turn. It is also possible that NS-3 will attempt to infect a node that already has the patch, wasting a turn. The new information sent to CyFiA is only information about nodes that has changed since the previous turn. This cycle continues until all the nodes in the network have been patched.

V. Experimental Measures

We tested CyFiA by comparing results of a patching mission to that of an exhaustive list of possible patch orders. The best-case results are determined in [4] using the previously established notional metrics of health, capability and security for the agility maneuvers.

TABLE I. Table Type Styles

Values	**0**	**25**	**50**	**75**	**100**
Health	***H(i)***	***H(s)***	**---**	***H(v)***	***H(p)***
Capability	***C(b)***	***C(i)***	***C(q)***	***C(h)***	***C(p)***
Security	***S(i)***	***S(h)***	***S(q)***	***S(b)***	***S(p)***

A. Notational Measures

Notional measures of health, capability and security are based on a maneuver state of a node and were numerical set based on their interrelationship.

Health is defined as the overall network health, this value ranges from 0-100 depending on the current state of the node: *i*) infected; *s*) susceptible, node shares a connection with an infected node; *v*) vulnerable, if the node if vulnerable to the infection but not infected, and *p*) patched, if the node is patched and therefore immune to the current infection.

Capability and Security are a function of the agility maneuver: *p*: Patch, *h*: Healing *q*: Quarantine and *b*: Blocking and Table 1 lists the numerical values for the node states.

B. Calculating Costs

The second parameter is cost of patching, each time the patch is transmitted between nodes in the MANET there is a cost associated with transmitting and receiving the patch, and therefore it is important that nodes are patched in an efficient way. We repeat the cost of patch calculation [4] here for the reader's ease. We define the data transfer rate in bytes/second for a link between node i and j as $T_{i,j}$. The energy required to receive data at a node is p_{rx} and the energy required to transmit data is p_{tx}, both in units of Joules. As an example, assume that the network nodes are linked between Node *m* and Node *r* with *r* - *m* intermediate hops. Node m holds the patch needed by Node r. Given a patch of size, *b* bytes, the cost to patch can be calculated as:

$$P_P(m,r) = \sum_{i=m}^{r-1} \frac{b}{T_{i,i+1}} (p_{tx} + p_{rx}) + p_a \tag{1}$$

It is reasonable to assume that the energy required for applying a patch, p_a, can be calculated offline and represented as a constant value. The cost of other maneuvers does not depend on network node links are assumed to be computed off line. It is also reasonable to assume that energy estimates to heal vulnerable software, quarantine a function, and update routing tables to block a node can all be calculated offline using nodes similar to that in the network and that they can be represented as a constant as is the case when calculating the energy cost necessary to apply the patch.

VI. Experimental Results

Each of these parameters is tested separately using CyFiA, and the results were compared to that of the exhaustive list of possible results. We can then determine if CyFiA was successful in achieving the best possible solution. In addition, there were two separate scenarios that we consider. The first focuses on securing the entire network, patching each node according to their current calculated health. The second scenario contains nodes that are set as 'mission critical' nodes, these nodes receive higher priority when selecting a node to patch. Patching will continue until all mission critical nodes are patched before moving on to patch the rest of the network. For each of these scenarios we use an exhaustive list of all possible patching scenarios, resulting in a total of possible 538 outcomes (agility maneuver sequences). The goal of utilizing CyFiA is provide the best-case scenario, in a single execution. We will compare our results from CyFiA to determine if we achieve the best-case scenario.

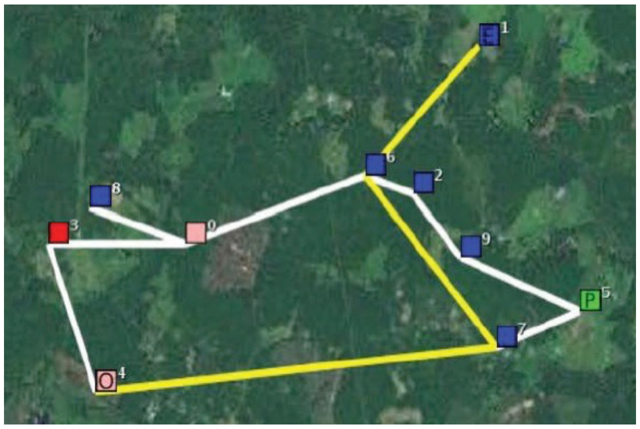

Fig. 4. CyFiA Scenario Network

A. Network First

The first test deals with patching the entire network with each node in the network given the same patching priority (each node is equally important). Fig. 4 shows the scenario is configured with one node representing the source of the infection (red node) and one node that is acting as the patch source (green node). Susceptible nodes are nodes that are subject to infection because they communicate with an infected node (pink nodes). Blue nodes are vulnerable and nodes along the yellow path are critical nodes.

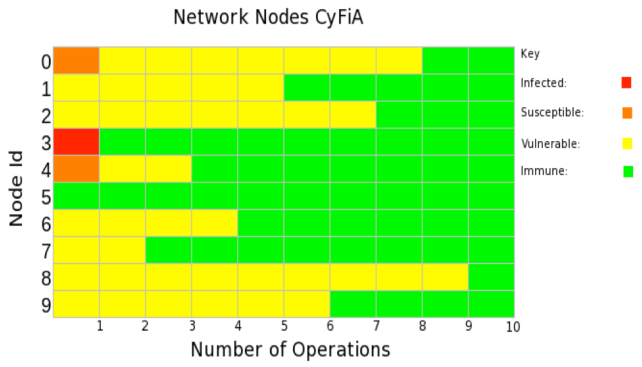

Fig. 5. CyFiA Health Map for Network Scenario

When analyzing the results from CyFiA, we need to look at a few different things, such as, how many nodes are infected before the infection is contained, and the number of operations it takes to fully patch the network. Fig. 5 shows that all nodes were patched in a total of 10 operations, which (because we are patching 10 nodes) is the least number of operations. However it is equally important to check the order that nodes were patched. Looking at Fig. 5, it can be seen that nodes CyFiA patches nodes first based on their health value, with node 3 being patched first because it is the first infected, but then will continue to patch the network, based on a nearest neighbor. This allows CyFiA to ensure that the patch is delivered to each node with the least number of hops possible. Because of CyFiA's ability to have complete knowledge of the network, it can select a node to be patched immediately upon infection, which may work for our current model, but will eventually need to include a delay to represent the time before the infection is detected.

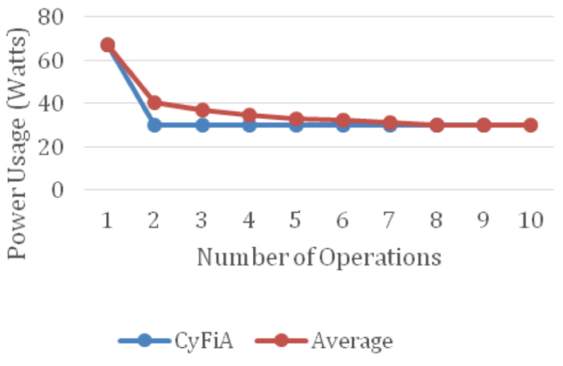

Fig. 6. Cost Compared to Average Cost per Operation

As the patch propagates through the network, there is a cost with each time the patch passes through a node calculated in Watts. Each of these values is taken into account when comparing the results from CyFiA. CyFiA's cumulative cost for patching the network in this scenario is 307.5W. This is considerably lower than the average cumulative cost of our exhaustive list of patching scenarios, which is 363.01W. Fig. 6 illustrates that the first node to patch is nearly always the most costly maneuver. This is because of the distance between the initial infected node and the source of the patch. After the initial infected node is patched, we can see how CyFiA selects to patch each node according to nearest neighbor. This allows CyFiA to always make the best choice minimizing battery/energy cost. Fig. 6 shows how after the initial turn, CyFiA always selects the node to patch based on distance from a node that already has the patch, therefore allowing it to perform under the average cost for the first few operations.

B. *Critical Nodes First*

The second scenario contains nodes that are set as 'mission critical' nodes. These nodes are shown in Fig. 4 the mission critical path highlighted in yellow. CyFiA will prioritize

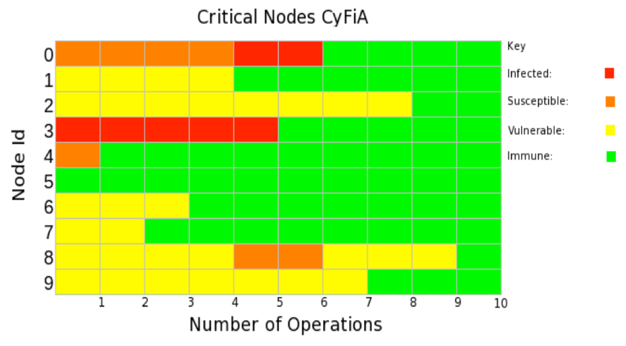

Fig. 7. CyFiA Health Map for Critical Node Scenario

patching these nodes first over other nodes in the scenario.

In the critical node scenario, the patching order that CyFiA gives us is based on patching critical nodes first based on their health. In Fig. 7, CyFiA chooses to patch node 4 first because of its proximity to the infected node 3. This move will ensure that critical node 4 is protected, but allows node 0 to become infected. After patching node 4, CyFiA will then begin to patch each of the critical nodes, based on health and distance from the patch. When we compare these results to the

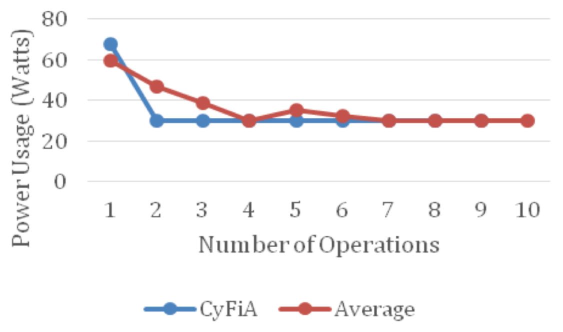

Fig. 8. CyFiA Cost Compared to Average Cost per Operation

exhaustive list, we can see that the average network health is slightly higher in CyFiA's results.

When measuring the cost associated with the CyFiA results we get similar results to the first scenario. The cumulative cost of patching the network is 307.5W. This is due to the fact that CyFiA is still patching nodes in accordance with nearest patched neighbor. Despite the fact that we are patching critical nodes first, the cost for patching the network remains largely the same. The average cumulative cost from our exhaustive list increases, however, to 366.58W. Fig. 8 shows the results of CyFiA when compared to the average cost per operation.

VII. Conclusion and Future Work

We have shown that a DSS can help aid agility decisions to shorten the amount of time a network is insecure. This is particularly advantageous in tactical military networks where constrained resources such as battery power and bandwidth as well as mission goals must be considered. We produced a simulation in which a CyFiA was used to contain and eliminate a malicious infection from a small network considering node criticality, health, security and capability. Our results will show that our DSS selects the best sequence of maneuvers and reduces energy costs.

Our future efforts will consider scenarios for securing nodes when a patch does not exists and when there are multiple vulnerabilities present in a network (with varying propagation rates). Additionally, information about the severity of vulnerable/infected nodes will be incorporated.

Acknowledgment

The effort described in this article was partially sponsored by the U.S. Army Research Laboratory Cyber Security Collaborative Research Alliance under Contract Number W911NF-13-2-0045. In addition, this research was partially accomplished under Contract Number N65235-06-D8847. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation hereon.

References

[1] Marek J. Druzdzel and Roger R. Flynn, "Decision Support Systems," Encyclopedia of Library and Information Science, Second Edition, Marcel Dekker, Inc., New York, 2002.

[2] Sheila B. Banks and Carl S. Lizza, "Pilot's Associate: A Cooperative, Knowledge-Based System Application," IEEE Expert 1991, October 2015.

[3] Norbou Buckler, Laura R. Marusich, Stacey Sokoloff, "The Warfighter Associate: Decision-support software agent for the management of intelligence, surveillance, and reconnaissance (ISR) assets" SPIE Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR V 2014, October 2015.

[4] Lisa M. Marvel, Scott Brown, Iulian Neamtiu, Richard Harang, David Harman, Brian Henz, "A Framework to Evaluate Cyber Agility," IEEE MILCOM 2015, Tampa, FL, October 2015.

[5] Boyd, J., "Organic design for command and control," In Discourses on Winning and Losing, unpublished slide presentation, Marine Corps University Research Archives, charts 16, 25 (1986).

[6] Larry Lafferty, "Velox Technology Overview," unpublished.