MIRA: A Support Infrastructure for Cyber Command and Control Operations

Marco Carvalho, Thomas C. Eskridge
Harris Institute for Assured Information
Florida Institute of Technology
Melbourne, Florida
mcarvalho,teskridge@fit.edu

Kimberly Ferguson-Walter
U.S. Department of Defense
Baltimore, MD
kjfergu@tycho.ncsc.mil

Capt. Nicholas Paltzer,
David Myers, and David Last
U.S. Air Force Research Laboratory
Rome, N.Y.
nicholas.paltzer@us.af.mil

***Abstract*—In this paper we introduce MIRA, an extensible and modular infrastructure designed for the support of coordinated cyber operations. MIRA was originally designed to provide the infrastructure for a cyber-command and control framework for cyber operations. MIRA provides a core set of services that can be extended to allow for operations in specialized or hybrid environments involving different service requirements or operational constraints. The framework was designed to support the specific needs required for command and control of network sensors, defenses and actuators, and to support the easy integration of third party network components, support services, or communication protocols. After a brief motivation about the needs and requirements for command and control in cyber operations, we introduce and discuss the design and implementation of the MIRA framework. The paper follows the discussion with the description of a simple scenario used to illustrate MIRA.**

I. Introduction

The concept of command and control (C2) is generally associated with the exercise of authority, direction and coordination of assets and capabilities. The concept of C2 encompasses important operational functions such as the establishment of intent, allocation of roles and responsibilities, definition of rules and constraints, and the monitoring and estimation of system state, situation, and progress. More recently, the notion of C2 has been extended beyond military applications to include cyber operation environments and assets.

Unfortunately this evolution has enjoyed faster progress and adoption on the offensive, rather than defensive side of cyber operations. One example is the adoption of advanced peer-to-peer C2 infrastructures for the control of malicious botnets and coordinated attacks, which have successfully yielded very effective and resilient control infrastructures in many instances.

Defensive C2 is normally associated with a system's ability to monitor, interpret, reason, and respond to cyber events, often through advanced human-machine interfaces, or through automated actions. Recent research activities in defensive C2 operations are now showing great potential to enable truly resilient cyber defense infrastructures.

In this paper we introduce and describe an agent-based framework called MIRA, designed to support resilient command and control infrastructures for cyber operations. Our focus is on the coordination of dynamic and moving target defenses, enabling C2 infrastructure that must be continually changing and evolving to better respond to an environment, requirements, and adversaries that are also continually changing and evolving [5], [7].

II. Requirements for Cyber C2

The goal of Cyber C2 is to increase the resilience of the overall computer network. Resilience is found not in the capabilities of individual defenses, but in the coordinated and contextual use of multiple defenses [3], [8]. We have identified several requirements necessary in order to create resilience in Cyber C2 systems.

Support Infrastructure for Distributed Communication and Computation A key requirement for Cyber C2 is a robust, distributed support infrastructure that enables a separation between the physical structure of resources on the network, and the logical organization of the network and network resources (see Figure 1.) This can be done in many ways, using custom programming, using Service-Oriented Architectures (SOAs), or as we have done, using distributed multi-agent systems [7]. This abstraction is necessary in order to identify the control loops needed to coordinate multiple, dynamic defenses.

Semantic Reasoning Reasoning in abstract control loops such as those shown in Figure 1, requires an abstract representation of the network resources, sensors, actuators and defenses, as well as attack vectors, and mission requirements. In MIRA, we represent these resources as OWL concepts [13], which enables abstract reasoning over network configurations, and specification and enforcement of semantically-aware policies that guide the behavior of the C2 system. Both of these capabilities are needed in order to reason about the effects of adding or removing a defense (c.f. [2]) or selecting between equally applicable defenses.

Fully automated control of highly complex systems tend to be brittle against intelligent adversaries. A solution must achieve the right balance between human and automated control. In fact, the best approach is to combine the control methods so that components of both human and automated control systems are operating continually and simultaneously. To achieve fast decision rates in critical areas, human input into control decisions is often made using semantic policies.

Flexible Interface for the Integration of Third Party Sensors and Defenses In order to support the requirements for abstract representation and semantic reasoning, it is necessary for a

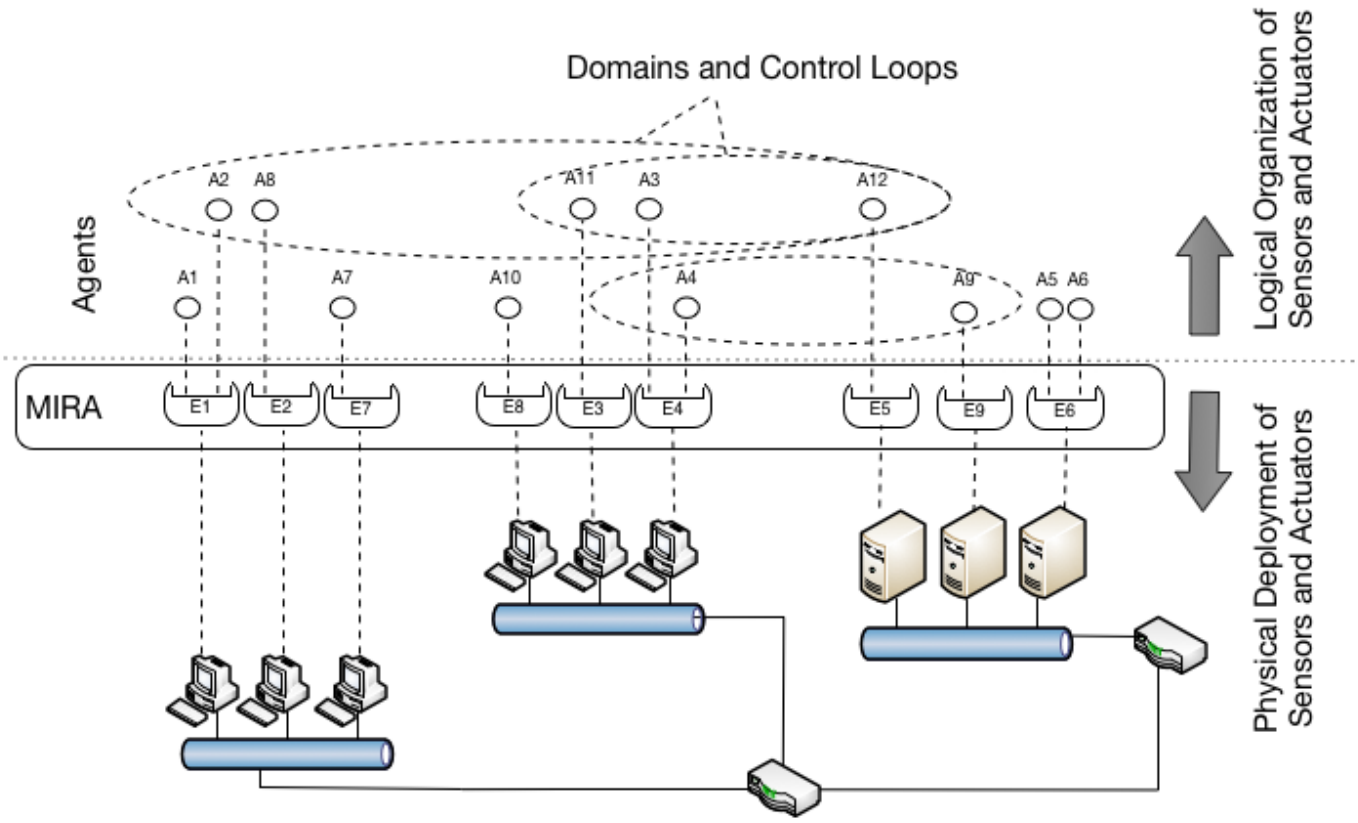

Fig. 1. Logical view of an agent-based command and control framework

Cyber C2 to be able to integrate new, third party sensors and defenses into the C2 framework. This involves two aspects of integration. First is the integration that allows the C2 system to control the sensor or defense by executing control functions or modifying the parameters of its operation. This requires use of an API, socket, remote procedure call, or file-based interactions. Second is the integration that enables the C2 system to determine when to deploy, enable, and/or modify the operation of the sensor or defense. This involves representing the resource requirements, controls, and parameter ranges and settings for the new sensor or defense in a way that the Cyber C2 can understand, manipulate, and control. In MIRA, this is accomplished by automatically creating an ontological representation when specifying the controls and their ranges. Previous efforts have demonstrated that new sensors and defenses can be integrated with an existing MIRA Cyber C2 system in very little time using this approach.

Distributed Learning and Coordination Support Resilience as defined in [3] requires not only robustness to withstand attacks, and the ability to recover back to a pre-attack level of performance, but to reorganize so that the same or similar attacks will not have the same effect on the network. This requires learning and coordination support so that an appropriate combination of defenses can be selected and parameterized to ensure maximal performance. MIRA can make use of several algorithms to enable this, including reinforcement learning [4], bayesian model building [9], or hierarchical model building [9].

Advanced Visualization and Human-Computer Interfaces Computer network defense is a complicated domain, requiring monitoring and understanding the relationships between hundreds of variables coming from numerous sensors and performance monitors placed throughout the network. It is critical that this information is presented in a way that is not only clear and understandable, but in a way that actually amplifies operator performance [11]. MIRA has several interfaces that show the organization and operation of the MIRA infrastructure [15] and of the traffic flowing through that infrastructure [10].

Specialized Test and Experimentation Infrastructures It is very difficult to experiment with Cyber C2 because of the scale needed to construct realistic testbeds and the need to isolate these testbeds in order to run malicious attacks. FIT has developed an infrastructure for Cyber C2 experimentation and testing called the Virtual Infrastructure for Network Emulation (VINE) [17], [16]. VINE simplifies the construction and management of large networks and isolates the networks from others, while maintaining a control backplane for data collection and experiment control.

III. The MIRA Agent System

MIRA is a software agent toolkit and runtime system for the development of distributed intelligent agent-based applications. From an architectural point of view, MIRA is a lightweight framework containing interfaces to a number of common services. Services are constructed to a well-defined

interface specification, and a default implementation, or service provider, of each service provided with MIRA. Additional services and service providers can be programmed and used in addition to, or instead of, the default implementations so long as they satisfy the interface specification.

A. Design Goals

There are a number of goals that MIRA agent system is designed to achieve.

Agent System Service Modularity. The goal of agent system service modularity is to insulate the agent execution environment, called the MIRA Environment, and its services from the agents. Agents are designed to interact with the service interface rather than interacting directly with the implementation of the service (see Figure 2.) The purpose of the MIRA Environment is to provide agents with policy-regulated access to key infrastructure services. By constraining the interaction to the service interface, the agents in the system remain unaware of how that service is implemented and, therefore, can make use of any service implementation that fulfills the service interface. For example, the MIRA Environment has by default two messaging service implementations, one UDP and the other TCP. Agents sending messages in a MIRA environment have no knowledge of which implementation is being used.

There are two benefits to this approach. First, there is a software engineering benefit to developing service-provider implementations to meet a prescribed service interface. This simplifies implementations and ensures that service-provider implementations that are specific to a particular network installation or that make use of specialized equipment will be interchangable with standard implementations. Second, there is a security benefit in not having all implementations for services in a network being the same. The idea of increasing the diversity of the network while maintaining functionality is central to the concept of moving-target defenses [6].

Security limitations on agents. MIRA agents have limited visibility into the lifecycle and service management functionality maintained by the MIRA Environment, so that agents cannot intentionally or unintentionally access agent system functionality that could be used to disrupt agent services. For example, agents have access to the agent discovery and lookup functionality of the Registration service, but they do not have access to register and deregister either themselves or other agents. By maintaining this separation between infrastructure functions and agent functions, we reduce the likelihood of deliberate or accidental errors caused by agents. Additional security measures include class loading, encryption, authentication, and code mobility. MIRA agents use a different class loader from the MIRA Environment class loader, so that malicious agents cannot load or access service classes by reflection. All infrastructure communications are encrypted by default, and PKI-based authentication is used to ensure that all participants in communications can be identified. A decision has been made to explicitly prevent agent code mobility, so that all MIRA Environments must contain all of the agent code necessary to run any agent that moves into the Environment.

Operating with Default Services. When a MIRA Environment starts up, it consults a local configuration to determine which services and agents should be started immediately. If this configuration file does not specify a specific implementation for a service, a default service is used. The default service generally has a minimal set of configuration dependencies, and are typically entirely self-contained. As is the desired behavior with all services, the default services either locate or create the service at startup, and negotiate with other existing components as necessary. Default services allow agent developers to focus their development efforts on the agent itself, with very little setup and configuration of the MIRA Environment required.

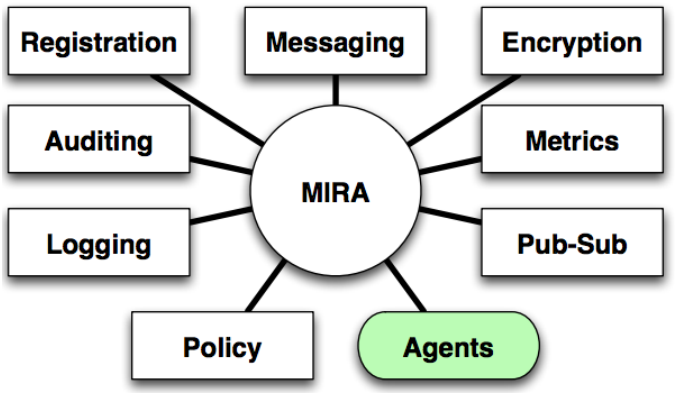

Fig. 2. Agent communicate with the MIRA Environment to access Core Services.

B. Core Services

The services that are currently specified for MIRA are shown in Figure 2 and include:

Registration Service This service provides a naming and location service for MIRA environments and agents in the network. Environments and agents have their address and particular properties registered with the registration service when they start up, are modified, and when they terminate. The registration service also provides methods for environments and agents to lookup the location of other environments and agents based on their registration properties. There are currently three registration service implementations provided with MIRA, two that require external servers, and one that is self-contained.

Message Service The message service is used by MIRA to send messages between agents and environments. Different implementations of the message service typically instantiate different transport mechanisms, such as the TCP- and UDP-based implementations that are currently provided.

Message Encryption Service The message encryption service provides the means for encrypting and decrypting data and messages. By having the encryption service as a separate service from the messaging service, any message encryption can be applied to any message transport service transparently to the user. There is one public-key encryption service provided with MIRA.

Audit Service The auditing service captures and publishes a timeline of all of the events issued by the MIRA infrastructure for use either internally for supervisory reasons or externally for debugging and visualization capabilities. There are three auditing services provided with MIRA, two supporting output of events to external databases, and one self-contained.

Logging Service The logging service is a utility service for developers and operators that specifies naming and location information for logs that may be generated on many different network hosts. There is one logging service provided with MIRA that enables configurable output of MIRA logs to the console, disk, or syslog.

Publish and Subscribe Service The publish and subscribe service provides the means for specifying the types of output agents and environments create, and the types of input that they need for processing. There is one simple peer-to-peer publish and subscribe service provided with MIRA.

Policy Service The policy service provides the means for controlling the operation of the MIRA services on an interactive, detailed level. Policy control of the services ensures that each action taken by the MIRA infrastructure can be checked in the context of the current operating conditions, as specified in a user-defined policy. There are two policy services provided with MIRA, one semantic-based and the other based on XACML [12].

Metrics The metric service provides the methods and functions for monitoring and reporting performance and resource usage measurements in the MIRA environment and for specific agent actions.

IV. An Illustrative Scenario

As an example of using MIRA for Cyber C2, we developed a scenario involving an enterprise network containing subnetworks for several different departments, including Human Resources, IT Security, Research, etc (see Figure 3.) Each of these subnetworks run a behavior system that emulates the behavior of typical operators in each of the subnetworks producing background traffic for the experiment [14]. An attacker machine was created behind its own router on a different network. The scenario runs under VINE in a private OpenStack cloud environment.

The VINE experimentation controller was used to deploy services, reset the scenario after each experiment, and to collect data results. The experiments were started and stopped using an Ansible [1] script on the experimentation controller. The script started the FTP traffic generators on the client nodes and the attacker scripts on the attacker node.

A C2Agent was responsible for sensing the state of the FTP services and controlling the moving target defense. To accomplish this, the enterprises used the MIRA agent system as a control interface for all sensing, messaging and coordination between enterprises. Both moving target defense and service sensing was started automatically by the MIRA system, which was bootstrapped to the virtual machine. This allowed the agent system to be started when the virtual machine started, which in turn started the moving target defense as well monitoring of the FTP service.

The attacker machine used a known vulnerability in one FTP application (proFTP) to terminate the FTP process and sleep for a predetermined time before performing the attack again. This behavior is meant to simulate a persistent attack from a malicious actor. The FTP servers in turn, had a watchdog task running that would restart the service when it was detected that the service was no longer running. The result of this attack and response is an FTP server that is intermittent and unreliable. The C2Agent will pick up on this diminishing service and adjust the moving target defense configuration to favor the non-vulnerable FTP server in an effort to mitigate the persistent attack.

A. Agent Configuration

In the experiment, MIRA environments were running on several computers in the IT Security department (middle left of Figure 3.) The key agents were

- *C2Agent.* This agent coordinates and controls the behavior of the defense-specific agents by tracking overall system state and suggesting parameter and configuration changes for defenses to agents that interface to specific defenses. The C2Agent uses the Publish-Subscribe core service to subscribe to updates from application and defense monitoring and control agents that indicate the performance and status of the applications and defenses. It publishes control messages to application and defense agents to reset or modify the control parameters associated with the defense or application.

- *C2 Interface.* This agent is a simple interface to the agent system that allows user interaction with the C2 Agent and defense-specific agents. Users can issue commands to the C2Agent or to the agents controlling the defenses directly. This agent runs in the same MIRA Environment as the C2Agent.

- *AppOSDiversityC2Agent.* This agent is the proxy to the agent-based C2 system for the AppOSDiveristy defense. The AppOSDiveristy defense is a general purpose Moving Target defense that switches between several different applications that provide the same service. In our scenario, the defense switched between two applications that provide the FTP service, ProFTP and vsFtp. The agent exposes a number of controls to the defense, including adding to or removing from the list of machines currently involved in the defense, setting the switching frequency

- *FTP Agent.* This agent monitors the status of the FTP servers and reports the status of the FTP service to the C2Agent. It does this by attempting to log into the FTP service and retrieve a small file. The agent counts and reports how many times the retrieval is successful and how many times it is unsuccessful during a specified time window. The C2Agent can control this time window to enable faster, but noisier, updates or slower, and more accurate, updates.

Figure 4 shows the flow of messages between the agents used in this experiment. The principal function of the Cyber C2 is to perceive the status of the environment (FTP Agent sending FTP status to C2 Agent), present that information to the operator (C2 Agent sending C2 Status to C2 Interface), and then to make a decision and enact it (C2 Agent sending AppOS Command to AppOS Diversity Agent).

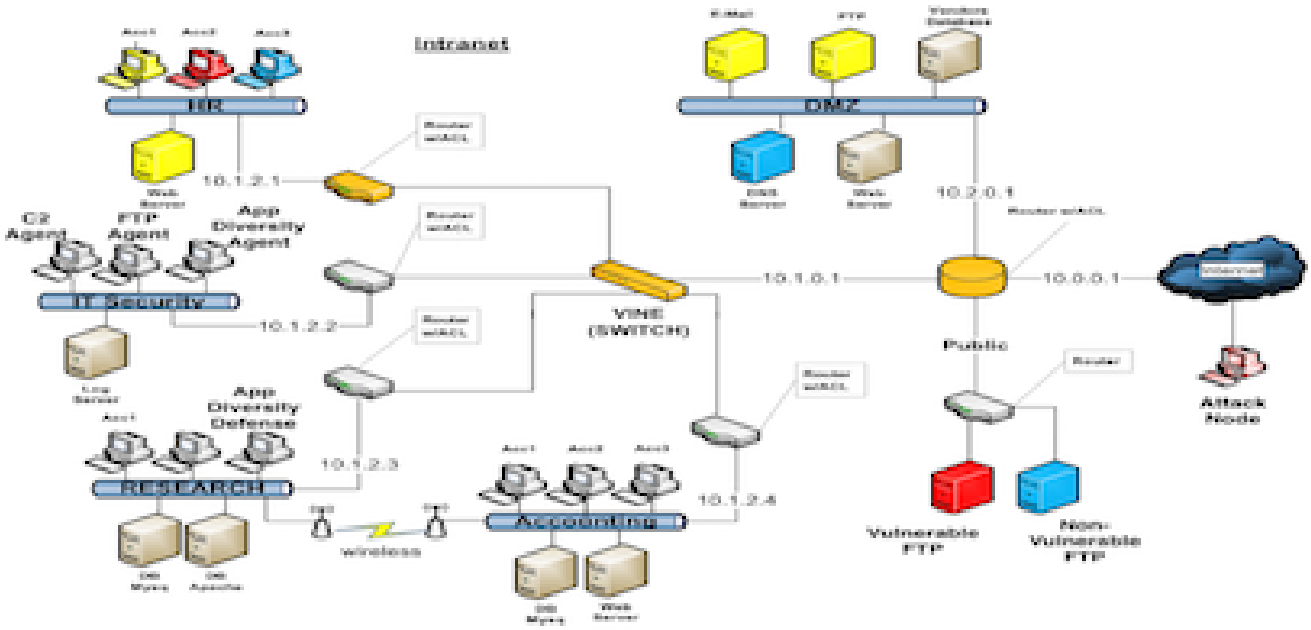

Fig. 3. Network diagram of the enterprise network used in the experiment.

B. Experiment Description

There were four separate experiments conducted and each were run multiple times to measure reliability and repeatability of the data collected. They were:

1) *No Attack (Control)* While the defense was running, the attacker machine was turned off and the two FTP clients in each enterprise produced FTP traffic at a constant rate. The behavior system logged in, uploaded and then deleted a file from the server and measured the successes, failures, iterations and transfer metadata.
2) *Attack* The attacker machine was activated and launched attacks on the public FTP site at regular intervals. Because a moving target defense was in use, there was only a single FTP url and the attacking machine was unaware of which FTP it was attacking. Thus, the expected result would be that only half of the attacks and half of the FTP attempts would be successful. This is the baseline that is used to measure any improved defense has on the system.
3) *Control Change* In this experiment, the C2Agent had the ability to dynamically reconfigure the moving target defense. An agent monitored the FTP application and log files and would trigger when it detected the FTP service success rate start to diminish. It would change the moving target defense settings to favor the FTP that was unaffected by the malicious actor. By configuring the moving target defense to use a 4:1 (20%) ratio instead of a 1:1 (50%), it is expected that the number of successful attacks will go down while the number of successful FTP attempts will go up.
4) *Further Change* The C2Agent reduced the time spent in the vulnerable FTP service to 10% of the available time, where we expect to see further performance gains.

The results of the experiment are shown in Table I and Figure 5. Table I shows the "dwell percent", which is amount of time spent in each FTP application, the number of FTP attempts, the number of failed attempts, and the overall percentage of successful FTP connections. Interestingly, even with no attacks being run, there were a few failed FTP attempts. We have yet to analyze the experiment data to determine the cause of the failure.

Starting the attacks has a significant, immediate effect on the FTP success rate. With the AppOS Diversity defense running and spending an equal amount of time using each FTP server, the success rate is the lowest seen. This is because 50% of the time, the Cyber C2 chooses a vulnerable application to provide the FTP service. The reason the success rate is not 50% exactly is that the time to recover from an attack is slightly longer than the time to complete an FTP session. Therefore, more successful FTP transfers than failures occur during the

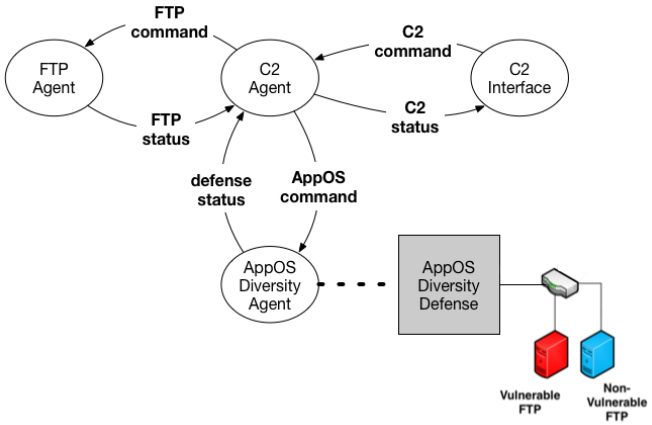

Fig. 4. Communication pattern of agents used in the experiment.

	Dwell Percent	Attempts	Failures	Enterprise
No Attacks	50	306	4	98.69281046
Attacks	50	299	94	68.56187291
Control Change	20	311	76	75.56270096
Further Change	10	319	66	79.31034483

TABLE I. SUMMARIZATION OF EXPERIMENT RESULTS

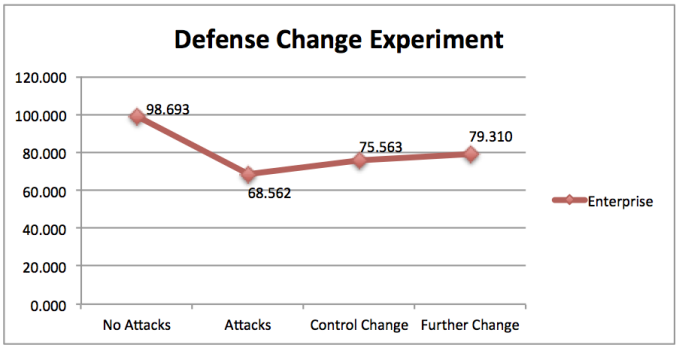

Fig. 5. Control changes to defense results in improved access to protected service.

same time interval.

Changing the control so that only 20% of the time is spent in the vulnerable service improves performance (Control Change), and reducing it to 10% improves performance still (Further Change). Eventually, the C2 will eliminate the vulnerable FTP application from being used, as long as the other FTP application continues to successfully respond to FTP requests.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have motivated the need for proactive, adaptive Cyber C2, and presented several requirements for building one. We discussed our distributed communication and computation infrastructure called MIRA, and showed an example scenario run in our experimentation infrastructure, showing the effects of controlling a moving target defense control parameter.

Our next step for the research into resilient Cyber C2 systems is to quantify the effects of defense deployment and parameter changes in order to make better decisions during subsequent attacks. Additionally, we are expanding the scope of experimentation to include scenarios with incremental responses, such as terminating/redirecting/isolating connections, locking out/logging out users, killing processes, or quarantining messages.

Our next steps in the development of the MIRA infrastructure are creating additional providers, such as interfacing with Kafka as a PubSubService provider, and using ZeroMQ as a MessagingService provider. We have also started the process of creating an open-source version of MIRA, OpenMIRA, that should be available soon.

ACKNOWLEDGMENT

This research project is sponsored by the U.S. Department of Defense. Any opinions, findings and conclusions or recommendations presented in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense.

REFERENCES

[1] Ansible home page, 2015.

[2] M. Atighetchi, N. Soule, R. Watro, and J. Loyall. The concept of attack surface reasoning. In *The Third International Conference on Intelligent Systems and Applications (Intelli 2014)*, pages 39–42, 2014.

[3] M. Bishop, M. Carvalho, R. Ford, and L. M. Mayron. Resilience is more than availability. In *Proceedings of the 2011 Workshop on New Security Paradigms*, NSPW '11, pages 95–104, New York, NY, USA, 2011. ACM.

[4] M. Carvalho. A distributed reinforcement learning approach to mission survivability in tactical manets. In F. T. Sheldon, G. Peterson, A. W. Krings, R. K. Abercrombie, and A. Mili, editors, *CSIIRW*, page 21. ACM, 2009.

[5] M. Carvalho, T. C. Eskridge, L. Bunch, A. Dalton, R. Hoffman, J. M. Bradshaw, P. J. Feltovich, D. Kidwell, and T. Shanklin. Mtc2: A command and control framework for moving target defense and cyber resilience. In *Resilient Control Systems (ISRCS), 2013 6th International Symposium on*, pages 175–180, 2013.

[6] M. Carvalho and R. Ford. Moving-target defenses for computer networks. *IEEE Security & Privacy*, 12(2):73–76, 2014.

[7] M. M. Carvalho, J. M. Bradshaw, L. Bunch, T. C. Eskridge, P. J. Feltovich, R. R. Hoffman, and D. Kidwell. Command and control requirements for moving-target defense. *IEEE Intelligent Systems*, 27(3):79–85, 2012.

[8] M. M. Carvalho, T. Lamkin, and C. Perez. Organic resilience for tactical environments. In J. Suzuki and T. Nakano, editors, *BIONETICS*, volume 87 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 22–29. Springer, 2010.

[9] M. M. Carvalho and C. Perez. An evolutionary multi-agent approach to anomaly detection and cyber defense. In F. T. Sheldon, R. K. Abercrombie, and A. W. Krings, editors, *CSIIRW*, page 28. ACM, 2011.

[10] T. Eskridge, M. Carvalho, P. Polack, F. Nebhard, and H. Thotempudi. Interactive visualization of netflow traffic. Technical Report HIAI-TR-15-5-2, Harris Institute for Assured Information, 2015.

[11] T. C. Eskridge, D. Still, and R. R. Hoffman. Principles for human-centered interaction design, part 1: Performative systems. *IEEE Intelligent Systems*, 29(4):88–94, 2014.

[12] S. Godik and T. Moses, editors. *eXtensible Access Control Markup Language (XACML) Version 1.0.* February 2003.

[13] W. O. W. Group. Owl 2 web ontology language document overview. Technical report, 10 2009.

[14] S. Mammadov, D. Metha, T. Toggweiler, and T. C. Eskridge. Youshka behavior system. Technical Report HIAI-TR-15-5-1, Harris Institute for Assured Information, 2015.

[15] P. Polack, M. Carvalho, and T. Eskridge. Visualizing multi-agent systems. In *2013 IEEE/WIC/ACM International Conference on Web Intelligence*, 2013.

[16] E. L. Stoner. A foundation for cyber experimentation. Master's thesis, Florida Institute of Technology, 2015.

[17] E. L. Stoner, M. M. Carvalho, and T. C. Eskridge. Vine: A cyber emulation environment for advanced experimentation and training. In N. H. D. Foundation, editor, *2014 National Security Innovation Competition Proceedings*, pages 14–17, 2014.