U.S. DEPARTMENT OF
ENERGY

PNNL-21854

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Final Report for Bio-Inspired Approaches to Moving-Target Defense Strategies

Glenn A. Fink, PH.D.
Christopher S. Oehmen, PH.D.

September 30 2012

Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062;
ph: (865) 576-8401
fax: (865) 576-5728
email: reports@adonis.osti.gov

Available to the public from the National Technical Information Service,
U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161
ph: (800) 553-6847
fax: (703) 605-6900
email: orders@ntis.fedworld.gov
online ordering: http://www.ntis.gov/ordering.htm

This document was printed on recycled paper.
(9/2003)

Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

Final Report for Bio-Inspired Approaches to Moving-Target Defense Strategies

PNNL-21854

Glenn A. Fink, Ph.D.
Christopher S. Oehmen, Ph.D.

Pacific Northwest National Laboratory

30 September 2012

Executive Summary

This report records the work and contributions of the Bio-Inspired Approaches to Moving-Target Defense Strategies project funded by the Networking and Information Technology Research and Development (NITRD) Program and performed by Pacific Northwest National Laboratory under the technical guidance of the client's Systems Behavior Research group. The project has incorporated a number of bio-inspired cyber defensive technologies within an elastic framework provided by the Digital Ants. This project has created (in collaboration with another DOE-funded project[1]) the first scalable, real-world prototype of the Digital Ants Framework (DAF)[11] and ntegrated five technologies into this flexible, decentralized framework: (1) Ant-Based Cyber Defense (ABCD), (2) Behavioral Indicators, (3) Bioinformatic Classification, (4) Moving-Target Reconfiguration, and (5) Ambient Collaboration. The DAF can be used operationally to decentralize many such data intensive applications that normally rely on collection of large amounts of data in a central repository.

ABCD is a hierarchical organization of human and software agents that relies on swarm intelligence and decentralized communication to defend large, complex cyber infrastructures from attack. *Behavioral Indicators* enable defenders to identify previously unknown problems before there are signatures for them by examining the behaviors of systems in a decentralized approach that enables machine-to-machine comparison automatically and in a scalable fashion. *Bioinformatic Classification* provides a means of inexact matching borrowed from proteomics that makes it harder for attackers to reuse old attacks. The *Moving-Target Reconfiguration* technology provides the basis for defenders to reconfigure systems automatically, increasing diversity between systems and simultaneously reducing vulnerabilities. Finally, *Ambient Collaboration* was a minor focus of this work that enables collaboration among human analysts without conscious collaboration effort. Together, these technologies, distributed via the DAF, will help secure large cyber infrastructures that support our society.

This report presents an operational scenario involving a corporate espionage situation and applies the proposed defense as a scalable and efficient solution for electric smart grid defense, communications security, industrial control security, and information technology cyber security. We present a brief summary of the most applicable literature and list the unique contributions of our work including: (i) a novel movement and pheromone model for randomly moving agents, (ii) a set of behavioral indicator data sources and sensors, (iii) a decentralized method of bioinformatics classification of malicious binaries, (iv) a way to distributively change configurations of machines in a moving target environment, (v) a method of ambient collaboration, and (vi) a number of publications.

We conclude the report with a description of our demonstrations that show the efficacy of the DAF in dynamic moving-target reconfiguration of systems and the scalability of the DAF to realistic enterprise scales. In the appendices, we recapitulate the project milestones and deliverables, expand the operational scenario, and discuss in detail the DAF architecture. The accompanying CDROM contains reports, deliverables, papers, videos, software, and other information on the Digital Ants and the related technologies developed by this work.

In this work, we have shown how these component applications may be decentralized and may perform analysis at the edge. Operationally, this will enable analytics to scale far beyond current limitations while not suffering from the bandwidth or computational limitations of centralized analysis. This effort has advanced the Systems Behavior group's Cyber Security research program to secure digital infrastructures by developing a dynamic means to adaptively defend complex cyber systems. We hope that this work will benefit both our client's efforts in system behavior modeling and cyber security to the overall benefit of the nation.

[1]DOE Office of Electricity's Cybersecurity for Energy Distribution Systems funded PNNL to develop and integrate Digital Ants under the Bio-Inspired Technologies for Enhancing Cyber Security in the Energy Sector project, project RC-CEDS-2010

Contents

List of Figures

List of Tables

Chapter 1

Overview and Background

1.1 Introduction

This research concentrates on achieving a Moving-Target cyber defense against complex-adaptive adversaries through the application of bio-inspired research technologies. The Moving-Target problems will focus on two research tasks: (i) Digital Ants as Moving-Target research: using mobile agents (based on digital social ants metaphor developed at PNNL) in combination with distributed genetic algorithms to provide a robust moving target environment, and (ii) Cyber economics: shifting the burden of complexity/costs back to the attacker via a biology-based classification of cyber entities approach that shifts the burden of complexity onto the attacker, making it harder to hide malicious artifacts. The digital ants technique will make our network a moving target in cyberspace, thus requiring increased sophistication (complexity/costs) on the part of would-be attackers. This work concentrates on designing and developing a trustworthy cyberspace–a system of defenses that operate in an environment that is presumed to be compromised.

High-level description The Bio-Inspired Approaches to Moving-Target Defense Strategies project has incorporated a number of bio-inspired cyber defensive technologies within the elastic framework of the Digital Ants. Digital Ants is a hierarchical organization of human and software agents that relies on swarm intelligence and decentralized communication to defend large and complex cyber infrastructures from attack. This project has invented or adapted five technologies and decentralized them via the Digital Ants Framework (DAF). The technologies and the corresponding prior work are listed in Table 1.1.

Table 1.1: Technology areas of the project, related PNNL prior work, and what this project accomplished in each area.

Technology	PNNL Prior Art	Work Accomplished
Digital Ants Framework (DAF)	Tactical Deployment and Management of Autonomous Agents (TDMAA)	Created a scalable core
Behavioral Indicators	Ant-Based Cyber Defense (ABCD)	Created new cyber sensors
Bioinformatic Classification	Machine Learning String Tools for Operational and NEtwork Security (MLSTONES)	Decentralized via DAF
Moving-Target Reconfiguration	none	Invented a genetic-algorithm-based dynamic reconfiguration system
Ambient Collaboration	Vulcan: Unexpressed Communication (cyber-security collaboration techniques)	Designed a new architecture for decentralization via DAF

The Moving-Target Reconfiguration technology was invented by PNNL's partner, Wake Forest University for this project. Moving-Target Reconfiguration introduces a genetic algorithm approach to automatic machine reconfiguration for reduced vulnerabilities and increased diversity. The DAF was developed in collaboration with another

DOE-funded project, Bio-Inspired Technologies for Enhancing Cyber Security in the Energy Sector[1]) Together, the technologies listed above, distributed via the DAF, will help secure large cyber infrastructures that support our society.

1.1.1 Operational Application

This effort has advanced the Systems Behavior Research group's Cyber Security research program to secure digital infrastructures by developing a dynamic means to adaptively defend complex cyber systems. The eventual result will be trustworthy enclaves in cyberspace that will have observable security metrics and be modeled so that abnormalities are readily identified and acted on. In this section, we briefly discuss potential operational outcomes for the digital ants framework and for each of the component applications of this work.

Operational Scenario The operational scenario revolves around a corporate espionage situation where the target corporation is defending its enterprise-computing infrastructure from cyber attack through a variety of attack vectors. We pay particular attention to the need for detecting malicious capacity of executable code and compromised computing elements. We assume that humans will always have a role in the decision making process, but that we can enable them to make rapid, well-informed decisions through delocalized sensing coupled with transparent (i.e. no penalty for false positives) secondary responses. The scenario describes both defender and threat capabilities showing how the technologies advanced by this work will enable a resilient cyber defense. The full scenario was a separate deliverable[2] and is reproduced in Appendix B.

Digital Ants Framework The framework can be used operationally to decentralize many data intensive applications that normally rely on collection of large amounts of data in a central repository. In this work, we have shown how four such component applications may be decentralized and may perform analysis at the edge. Operationally, this will enable analytics to scale far beyond current limitations while not suffering from the bandwidth or computational limitations of centralized analysis. What must be traded in is a central overview of the entire problem space, with humans being able to participate in the analysis at the lowest levels. Arguably, this cannot be done even now since centralized decision-making, even when supported by visual or automated analytics, cannot keep pace with the volume and velocity of relevant cyber information. Instead, the DAF provides a means for accomplishing effective analytics at the edge, making it possible once again to keep up with the growth of networks and processing. Operationally, this means that humans will be in the *right* loop, at a level where they can influence the system appropriately, not "down in the weeds" looking at individual data items. Decentralized analysis will enable analysis to scale without requiring humans to be hired to scale with the size of the data and networks.

Behavioral Indicators While malicious software can vary greatly in form, it nearly always produces unintended side effects on the systems it colonizes. By examining the behaviors of systems rather than the files on the systems, behavioral indicators enable defenders to identify previously unknown problems before there are signatures for them. By decentralizing this behavioral analysis, our work shows how system behaviors can be compared with one another automatically and in a scalable fashion.

Bioinformatic Classification By providing a means of inexact matching, the bioinformatic classification methods demonstrated in this work will shift the workload from the defenders to the attackers by making it harder for attackers to make an old attack work again simply by changing a few bits. Rather than relying on checksums, this method uses bioinformatic sequencing approaches that borrow from proteomics to find highly conserved regions of code (motifs) that are indicative of individual exploits. From these, we can determine, not just whether malware matches exactly, but to what degree it matches each member of a large set of known threats. Thus, we can characterize instances we have not yet seen by the degree they match ones that we have not yet seen. A further, important, side effect of decentralizing this technology is that individual ants carry only fragments of motifs. When one ant's fragment matches, we have strong suspicion that a malware-related file resides on the monitored system. Swarming enables us to match other fragments until we have an arbitrarily high degree of confidence in the match. It becomes very difficult for attackers to change their code fundamentally enough to completely fool this system. From an operational perspective, what we obtain is the ability to stay ahead of attackers by fully utilizing the large amount of data they have already provided us. Conversely, we have made it much harder for attackers to devise new exploits that will not be detected by bioinformatic classification.

[1]funded by DOE Office of Electricity's Cybersecurity for Energy Distribution Systems, project RC-CEDS-2010
[2]This was Deliverable #3, completed 06 July 2011. On the accompanying CDROM this is file D03-Scenario-Final.pdf

Moving-Target Reconfiguration Operationally, software monocultures are much easier to maintain because similarity between systems enables greater levels of automated maintenance. However, monocultures also increase the degree of saturation possible for a given exploit. Our Moving-Target Reconfiguration provides the basis for defenders to reconfigure systems automatically, increasing diversity between systems and simultaneously reducing vulnerabilities. While this approach does complicate things for defenders, it makes systems much more difficult for attackers to comprehend and removes their ability to rely on their reconnaissance. Further, rather than centrally controlling this reconfiguration, our work has shown how digital ants can share good configurations and knowledge about vulnerabilities resulting in systems that automatically configure themselves to be secure and diverse. Operationally, this will enable systems to defend themselves, freeing defenders for other work.

Ambient Collaboration A minor focus of this work is using the DAF to enable collaboration among human analysts without conscious collaboration effort. Our ambient collaboration work uses the Digital Ants as a means of distributing standing queries throughout an enterprise and sharing queries and query terms among cyber defenders who may or may not be actively collaborating. Queries and terms are collected from participating analysts' queries of local data and their accesses to external information. These terms are sanitized and redistributed throughout the enterprise as new types of ants. Queries and terms that match are rewarded and those that fail to match eventually die out. In this way, participating analysts can follow global trends in the interests of analysts without unintentionally sharing sensitive information.

1.1.2 Background

PNNL has been researching digital ants since 2006 with initial models and prototypes appearing in 2007 and 2008. Digital ants was conceived as a way to achieve concerted, infrastructure-wide cyber-defensive action that spans organizational boundaries. The desire was to enable enclave defenders to coordinate the activities of their defenses without violating the sensitivities of cooperating organizations. Humans must retain ultimate authority and responsibility while avoiding becoming a bottleneck. Meanwhile, automated defenses must strictly observe access policies that differ across cooperating organizations. The mixed-initiative approach provides a basis for shared control among humans at different sites and for humans and software agents at different sites to collaborate.

Current cyber-defense systems involve humans at multiple levels, but people are often far down in the control structure, requiring them to make too many time-critical decisions. Information flow between humans is slow and frequently asynchronous. In a crisis, humans may be unable to cooperate because of cultural, language, legal, proprietary, availability, or other obstacles. Such systems cannot adapt to rapid cyber threats. Effective cyber defense requires a framework that simultaneously capitalizes on the adaptability of humans and the speed of machines. Humans must have a correct balance of decision making and delegation to maximize their effectiveness and to acknowledge their legal responsibility for the actions of their automated systems.

This work builds on the Digital Ants framework more completely presented in other work[11][3]. We have taken the flexible structure afforded by the digital ants and extended it with other application technologies (listed in Table 1.1). This work (in collaboration with another DOE-funded project[4]) has funded the creation of the first scalable, real-world prototype of the digital ants framework.

There are several potential application domains for the proposed bio-inspired security defense. Each of these domains consists of a potentially diverse infrastructure of computing devices that dynamically change over time. In these cases, the proposed defense offers a scalable and efficient solution when compared to traditional static approaches.

Electric Smart Grid Defense A locality may experience power shortages or limited grid connectivity due to natural hazards or malicious tampering. Digital ants could be employed to dynamically identify impacted systems. If the impact is severe enough to partition the grid into electrical islands, then digital ants would adapt to the new electrical topology and seek to match power generation with electrical loads. This could be done in part through the appropriate use of pheromone. Pheromone could also be used to help direct agents to nodes in need of cyber repairs or reconfiguration.

Communications Security Malicious smart phone apps steal information from phones, eavesdrop on conversations, track users, or infect the computers or devices that connect with the phones. Digital ants roaming across subscriber phones would notice anomalous behavior. Analyst social networks could provide cues about external

[3]Please refer to the files in the Extras/Papers folder of the accompanying CD-ROM.

[4]DOE Office of Electricity's Cybersecurity for Energy Distribution Systems funded PNNL to develop and integrate Digital Ants under the Bio-Inspired Technologies for Enhancing Cyber Security in the Energy Sector project, project RC-CEDS-2010

symptoms of the problem, and bioinformatic matching could reveal whether the current attack vector is related to ones seen previously or not.

Industrial Control Security Supervisory Control and Data Acquisition (SCADA) systems regulate critical infrastructure elements like hydroelectric dams, wastewater reclamation, factory processes, and oil pipelines. These systems are a promising target for those who may threaten the populace to attain political or economic objectives. Malicious actors often repurpose old exploits that we have existing signatures for. Where exact matching would fail, our bioinformatic characterization approach is not fooled by syntactic differences. Digital ants carrying attack motifs will find even obfuscated attack tools.

Information Technology Cyber Security The digital ants compare the behaviors of host machines to one another over the entire enclave. Host-based agents that the digital ants report to catalogue changes to a single machine over time. Multiple host information can be collected to give a view over the whole enclave over both space and time. These changes might indicate misconfiguration, broken elements, or even malicious activities. Coupling this with ambient human collaboration and bioinformatic classification provides powerful cyber diagnostics.

The proposed bio-inspired framework can be used to defend enterprise-level infrastructures more efficiently and robustly than current methods. The proposed system potentially offers greater flexibility with regards to managing dynamic environments. It is also well suited for detecting, diagnosing, remediating, and recovering from security issues.

1.1.3 Prior Work and Applicable Literature

Autograph [16] is a distributed system for automatically generating signatures of Internet worms via coordinated sensors that examine byte sequences within TCP packets. Autograph depends on the ability to heuristically identify worm-like patterns in network traffic and employs a “tattler” to share these signatures with other sensors. Unlike CID, Autograph does not incorporate other kinds of rationality, provides no basis for trust among signature-sharing sensors, and does not use feedback to benefit from the false positives it generates.

CRIM [6] is a cooperative module designed for the MIRADOR distributed intrusion detection system (IDS) that clusters, merges, and correlates IDS alerts. CRIM’s correlation attempts to reduce administrator burden by composing sets of related “elementary” attacks into compact attack scenarios that mirror the plans and intentions of the attacker. While CRIM would save human workload, unlike CID, it requires the human to work closely with the software to correlate alerts in to an attack plan. CRIM shares data that could make it useful in an infrastructure, but does not use swarm intelligence, nor does it make efficient use of false positives.

Cossack [19] employs a distributed set of “watchdog” systems deployed at the boundaries of large networks or Autonomous System to detect and control DDoS attacks. While this is a decentralized system that would enable multiple organizations to cooperate in a cyber defense at the large network level, it does not use mobile agents at the individual host level nor does it allow for emergent cooperative sensor behavior. Cossack does not involve humans at all, thus it is useful primarily for high-speed malware such as worms.

Similarly, Nojiri, *et al.* [18] describes a system that is designed to thwart worm attacks. This system uses decentralized mobile agents for detection and reaction and is designed for emergent features. Essentially Nojiri’s system is a “white worm” designed to propagate among “friends” to defeat malicious worm spread. Nojiri presents no mechanism for controlling spread of the worm, and no indication of human involvement.

The hierarchical arrangement of humans and various types of agents has been proposed in a variety of forms. Many of these did not have security applications in mind and none of them considered the special needs of infrastructures.

Qin, *et al.*[21] discuss how IDS may be merged with network management systems (NMS) using a hierarchy of multiple heterogeneous lightweight agents. Similarly, Śmieja[23] presents a heterogeneous hierarchy of agents based on neural nets and Selfridge’s Pandemonium system[22]. Ibrahim[15] proposes a network management system that utilizes a hierarchy of mobile agents to manage a distributed system while reducing bandwidth consumption. Parunak [20] proposes a heterogeneous hierarchy to solve highly constrained military movement problems.

Carvahlo [2] presents the Luna Agent Framework, an agent-based distributed survivability framework that uses reinforcement learning for mission resilience. Luna implements mission decomposition and distribution with replication of critical components and differential task allocation based on estimated level of threat. Level of threat is estimated from a locally perceived attack, or the possibility of an attack, based on threat information that is shared between similar nodes. While Luna is agent-oriented, provides resilience, and may be applied to cyber security, it is more of a framework that digital ants could be implemented in rather than a cyber security solution itself. Luna prescribes no particular degree of human involvement.

While Ibrahim and Parunak do mention human involvement all of the cited works basically relegate the human to the status of an observer with very limited interaction within the system. None of them contains a concept of multiple cooperating organizations and none combines complex-adaptivity with human knowledge and insight. Digital ants is an agent-based framework for cyber security that provides a place for human agents as well as software agents. It directly addresses cyber security and can itself be a framework for implementing multiple types of cyber security technologies in a distributed framework. In the next chapter, we will discuss the contributions of the work to date and show the value digital ants provides as a framework for hosting other cyber security technologies.

Chapter 2

Contributions of the work

2.1 Component Technologies

This chapter discusses each part of the Bio-Inspired Approaches to Moving-Target Cyber Defense project: the digital ants framework (DAF) and the four embedded applications of it. We describe each technology briefly and highlight the contributions of each.

2.1.1 Digital Ants Framework

We re-engineered the DAF as a lightweight, scalable, extensible, mobile-agent framework in Python. Previously, the DAF was written in Java and depended heavily upon the Java Agent Development Environment (JADE) framework that was both cumbersome and required centralized communications. The reengineered DAF now runs on micro-controllers, individual machines, DETER networks, and the PNNL Institutional Computing (PIC) cluster. The PIC has 450 nodes[1] with 32 cores each for over 14,000 cores. By running multiple partitioned file systems on each core, we were able to simulate large networks on a fraction of the PIC nodes. We have achieved runs with up to 20K Sentinels.

Movement and Pheromone Model The new Python Digital Ants Framework has a re-engineered movement and pheromone model that represents a true contribution of this work. The new model enables decentralized geography management and provides greater stability for real-world deployment. We use pheromone vectors to implement a decentralized direction model that enables a directed random walk.

We calculate the effective pheromone vector T as the sum of the vectors: $\vec{D}$: the deposited pheromone, $\vec{B}$: the background pheromone, $\vec{H}$: the heading bias pheromone equivalent, and $\vec{T} = \vec{D} + \vec{B} + \vec{H}$. The discrete probability function vector that determines the ant's next direction, P, is T normalized as follows:

$$\vec{P} = \frac{\vec{T}}{\|\vec{T}\|} \tag{2.1}$$

The i^{th} element of $\vec{P}$, p_i is the probability of an ant going in direction i. We the define the cumulative probability vector, C, as:

$$c_j = \sum_{i=0}^{j} p_i, j \in \{1, \ldots, n+1\} \tag{2.2}$$

The cumulative probability vector gives the probability range for each cardinal direction as an ordered set. Random variable $X \in (0, 1]$ is compared to each c_j in order to find $c_j < X \leq c_{j+1}$, and the resulting j is the index of the new cardinal direction the agent will take. By applying the heading bias vector $\vec{H}$ we ensure that the agent is more likely to continue in its current direction and unlikely to turn in a retrograde direction.

[1] PIC-Compute has 5 head nodes, 450 nodes, and 4 fat nodes. A node equals 32 cores, 64GB memory, 1TB disk. A fat node is a node with 256GB memory.

Experimental Parameters The work done for this component has included simulations that have helped refine the population dynamics, pheromone, and movement models used in the DAF. Particularly, the discrete event simulator used in this work showed that digital pheromone enables effective coverage and hit times at an initial agent density of < 0.1. For the large-scale demonstration of the DAF, we experimented with ant densities from 0.03 to 1.0 (population of ants to Sentinels). For this demonstration we used the bio-informatics application with a static number of signatures, and we did not enable population dynamics for simplicity's sake. The pheromone decay factor commonly used was 1.1 (unitless), and at every timestep (second) we reduce the deposited pheromone as follows: $\vec{D}_{t+1} = \vec{D}_t/decay$. We experimented with a variety of pheromone path lengths (the number of hops an activated ant will go while continuing to drop pheromone) and found that 15 hops produced sufficiently long trails. The deposited pheromone strength used was 20 units per hop. These settings consistently produced swarms.

Contributions Our re-engineered DAF proved that the digital ants concept will work on real hardware at realistic scales. It also enabled us to experiment with the many settings that model allows. By adjusting various parameters such as pheromone path length and decay rates, we found that the DAF can be tuned to a wide variety of real hardware environments.

2.1.2 Behavioral Indicators

The behavioral indicators component seeks to use non-specific activity signatures to characterize the behavior of individual systems as compared to their peers. The intent was not to identify signatures of malicious or suspicious activity, but rather to identify and quantify general indications of behavior that could be used as a basis of comparison between machines. The underlying assumption is that if a system differs greatly from its neighbors in many ways, there may be something wrong. Additionally, behavioral indicators may help to characterize what is normal behavior for each system.

Our approach was to identify sources of data that could be mined for indications of differences between machines. Then we would use these indicators as the basis for creating digital ant sensor specifications. By mixing data sources and indicators, we can derive a large number of widely varying sensor types in a fashion similar to how the mammalian immune system works. Figure 2.1 shows a list of data sources, indicators, and sensor definitions and how they may be notionally connected.

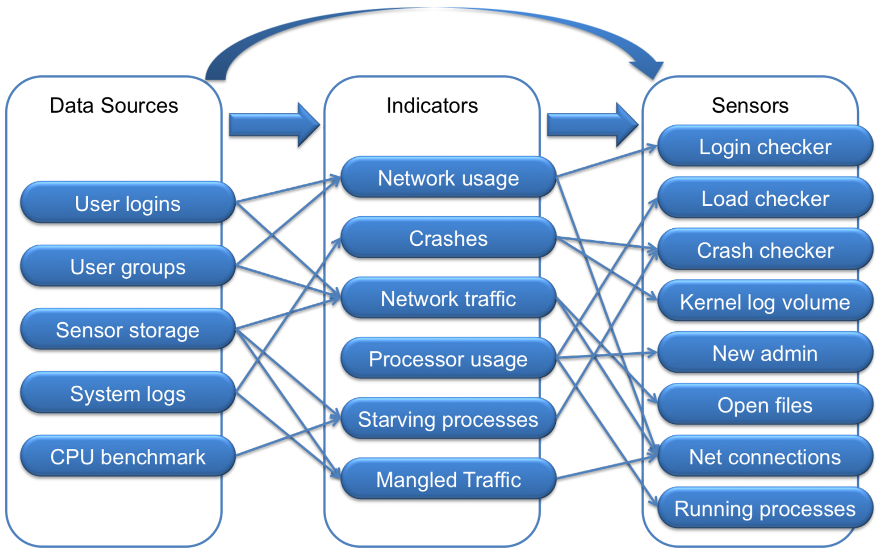


Figure 2.1: The Behavioral Indicators work uses a hierarchy of data sources, indicators, and sensors. Data sources may contribute to either indicators or sensors. Connectivity shown here is notional only.

In the DAF, sensors are just packets of data passed between Sentinels at each monitored machine in the enclave. Sensors carry only a brief history of their observations and the kind of sensor they are from machine to machine. Thus, we are not passing executable code from machine to machine. Instead, the definition of each sensor type must already be defined as part of the DAF at each managed node. Sensors of a type that does not exist on the given Sentinel cannot execute there.

Each arriving Sensor needs a data source to read to compare its historical observations against. These data sources are implemented as application programming interfaces (APIs) at each Sentinel. Over the course of this project, we implemented five data sources (Table 2.1), nine indicators that use these data sources (Table 2.2), and eight Sensor types (Table 2.3). Each sensor type recombines data from the data sources and the definitions of symptoms from the indicator types.

Table 2.1: Data sources defined and their implementation status.

Name	Description	Status
UserLogins	Runs the last' command and parses the output and returns it with data grouped by user.	Implemented
UserGroups	Given a user name returns what groups they belong to.	Implemented
SensorStorage	Sensors can write to and later retrieve data from this data source.	Implemented
Syslog	Returns parsed log data from kern.log, syslog, or auth.log. User can specify log file, how far back to go, search words to filter by etc.	Implemented
BenchMarkCPU	Carries a math expression as a benchmark string and returns a list of results.	Implemented

Table 2.2: Indicators defined and their implementation status.

Name	Description	Status
Processor Usage	Uses standard system tools to report processor usage.	Implemented
Program Crashes	Kernel logs provide information and frequency on program crashes and other recoverable faults	Implemented
Network traffic absent user	Monitors kernel logs for login events and compares traffic activity	Implemented
Processor usage absent user	Monitors logins and compares processes	Partial
Starving processes	Identifies starving or deadlocked processes using basic operating system tools plus analytics	Implemented
Mangled network traffic	Looks for unusually formatted network traffic	Partial
Memory Hash	Takes a snapshot of memory	Future
Browser homepage change	Detects recent homepage changes	Abandoned
New startup programs	Detects new autoloading programs	Abandoned

Contributions This research implemented the first set of behavioral indicator ants tuned to perform detection in real systems. To implement them, we also produced a generic data source, indicator, and sensor architecture that will work for multiple kinds of ants in the future. We implemented a variety of ant types and demonstrated that they are able to move around and compare the characteristics of real machines.

2.1.3 Bioinformatic Classification

Because our objective is to detect new types of malware we needed a novel method of detecting previously unseen malware binaries. We adapted PNNL's bio-inspired classification method, Machine Learning String Tools for Operational and Network Security (MLSTONES). MLSTONES creates a vectorization strategy for converting binaries to "amino acid sequences." These strings can be matched using the same partial matching technology used to perform protein sequencing.

MLSTONES was originally formulated as a centralized matching technique where all the vectorizations of each binary to be analyzed must be brought to the database of motifs for comparison. This project formulated a new method for decentralized analysis via digital ants. To prove that the digital ants concept enhanced with bioinformatic classification can discover polymorphic malware in real time at realistic scales we ran experiments on PNNL's institutional computing (PIC) cluster.

Because the PIC cluster is not approved for processing of malware, we substituted Windows binaries as a malware stand-in since the PIC is a Linux cluster. By perturbing the motifs used, we simulated what would happen as the

Table 2.3: Sensors defined and their implementation status.

Name	Description	Status
LoginChecker	Detects recent admin and root logins.	Implemented
LoadChecker	Checks CPU load via benchmarking and querying OS facilities.	Implemented
SegFaultChecker	Detects number of segfaults in last 20 days via log files.	Implemented
KernelLogVolumeChecker	Detects kernel log volume.	Implemented
NewAdminChecker	Detects accounts that were recently given admin via the auth.log file.	Implemented
OpenFilesChecker	Counts the number of currently open files.	Implemented
CurrentNetConnectionsChecker	Counts the number of currently open network connections.	Implemented
RunningProcessesChecker	Counts the number of currently running processes.	Implemented

"malware" changed form and function over time. We ran multiple experiments where a single malware file had to be located on virtual clusters as large 20,000 machines.

Contributions An example of detecting polymorphic malware is shown in Figure 2.2. In the initial blue (dashed line) run, all 97 fragments of the malware were identified within 35 seconds. For the red (solid line) run, we encoded the signatures with a completely different vectorization strategy and we still matched nearly half of the possible fragments. This is a testament to the robustness of our detection method even when new malware varies greatly from the exemplars in the motif database. Swarms will still form for malware that is distantly related to the known varieties although the swarms may be less intense than those forming for known entities.

2.1.4 Moving-Target Reconfiguration

The Moving-Target Reconfiguration component seeks to develop a new moving-target environment that manages the security and diversity of computer system configurations using genetic algorithms distributed by Digital Ants. A computer system configuration consists of a complex set of values (file permissions, port assignments, account settings, etc.) that individually, or in combination, can yield a system that is functionally appropriate, stable, and secure. However determining a good configuration is difficult given the large number of possible system settings and their potential interactions. In an environment that consists of similar computers, these configurations should still be different from one another, yielding a diverse landscape that hinders the progress of potential attackers.

This component uses Genetic Algorithms (GAs) to find configurations that provide the desired functionality while maintaining a degree of diversity. When applied to system management, the system configuration is modeled as a chromosome where each setting (e.g. file permissions) is a trait or allele. While each computer can perform a GA in isolation, the Digital Ants framework is instead used to share genetic information between systems to improve the chromosome pool. Even though configurations are shared across computers, the mutation and crossover components of a GA will encourage configuration diversity as done in nature. As a result the proposed approach should migrate the computer ecosystem towards a stronger security posture. The resulting improvements are illustrated in Figure 2.3.

Contributions Simulations of Moving Target Reconfiguration showed how the system can: (i) increase spatial diversity of configurations (making machines differ from one another), (ii) increase temporal diversity of configurations (making a single machine change configuration over time), and (iii) reduce vulnerabilities. Experiments to compare effectiveness of various pheromone models have been performed and are covered in detail in the final report from Wake Forest University provided as deliverable 13[2]. The work also resulted in several papers ([4, 5]) [3].

[2]On the CDROM the document name is D13-GA-Final.pdf.

[3]See MWSCAS-2011-swarming-agents-for-scalable-security.pdf and SAFECON-2011-MT-via-genetic-algorithms.pdf on the CDROM in the Extras/Papers/ directory

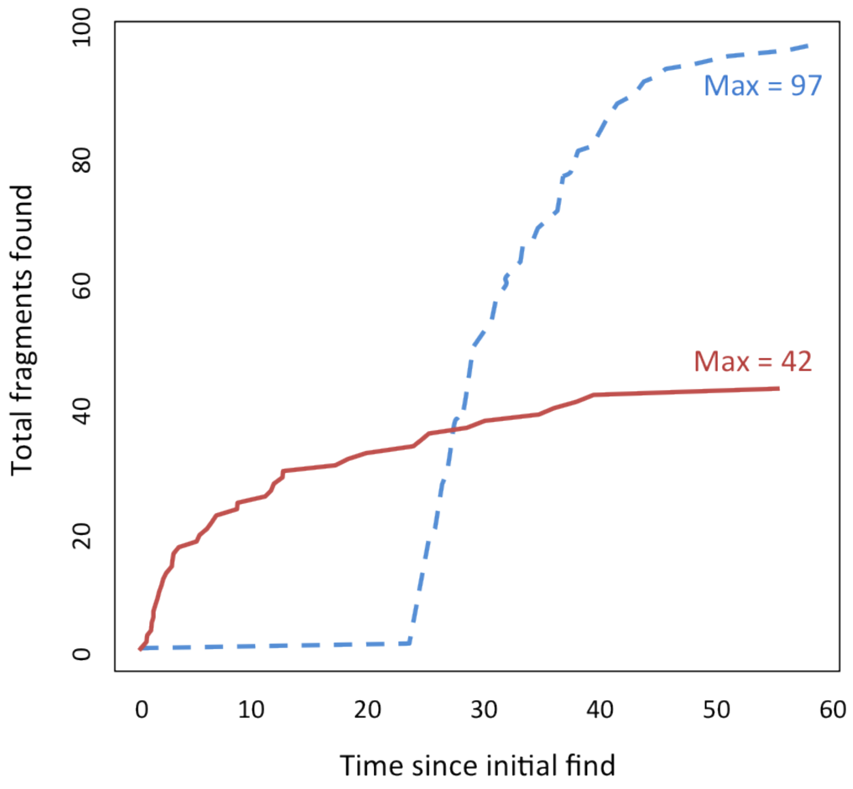


Figure 2.2: By changing vectorizers, we showed how that even if malware changes to a large degree we can still detect changed binaries.

2.1.5 Ambient Collaboration

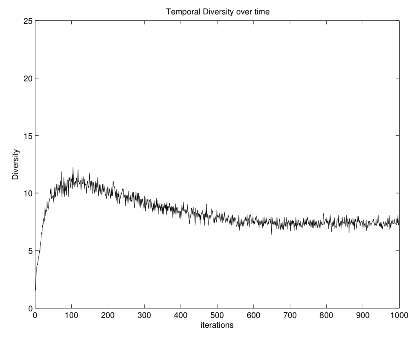
Collaboration among cyber security analysts is a cumbersome process because of the overwhelming volume of security data available and the data owners' hesitance to share data for fear of exposing sensitive information. Ambient collaboration is an application built on the digital ants that enables collaboration without sharing private data. Ambient collaboration was adopted from another PNNL project, Vulcan, which provides a framework where analysts can collaborate securely with their peers without risking exposure of private information, without introducing new tools, and without interrupting analysts' workflows.

Ambient collaboration enables analysts to understand what information others are attending to by sharing the questions analysts ask of their data and the information sources they use to stay informed. The questions they ask include queries against internal databases, joining and exclusion operations performed on data, and command line activities that are used to narrow data. An analyst may also consult blogs, news sites, and other online sources for information on security vulnerabilities and methods to protect their enclaves. The Vulcan project was developed as a way to leverage that communal knowledge within the cyber community.

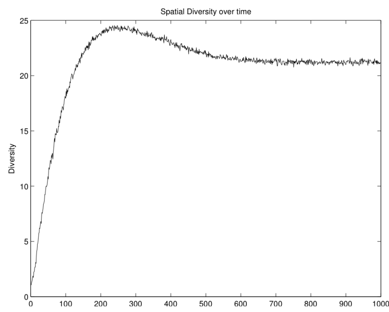
We wish to convert the questions analysts ask of their data and the information sources they use into standing queries that will guide future analysis without requiring any effort from analysts beyond that required to do their normal jobs. Digital ants can then carry these constructed queries throughout an enclave making them standing queries that return information whenever they match.

It is relatively simple to convert data query terms of various kinds to structured data for standing queries [1], but it is less straightforward to do so with unstructured data from web sites, etc. [14]. For data queries, the Vulcan project uses a simple parser to pull out meaningful chunks of data from textual query streams, normalize them according to a common ontology, publish them to subscribers, and finally to translate the normalized queries back into a form that is compatible with the local data ontology. For online sources, Vulcan uses a lexical scanner to pull tokens (e.g., names, places, terms, etc.) from web pages and other sources. Then it uses a natural-language processing pipeline to determine the contextual meaning of the tokens and how they could be used as queries.

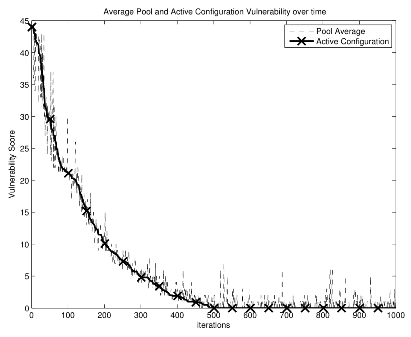
The piece that was missing prior to this research was how to convert these query terms and token streams into meaningful standing queries against internal data sources. This research does not actually implement the proposed



(a) Temporal Diversity



(b) Spatial Diversity



(c) Vulnerability

Figure 2.3: Temporal and spatial diversity are enhanced by moving-target reconfiguration while vulnerability is reduced.

solution, it merely identifies how the solution might be achieved. We propose applying a semantic reasoning engine, like PNNL's CHAMPION reasoner to convert these streams into digital ants that may be used to find matches to queries of concern to the cyber analyst. A block diagram for such a system is shown as Figure 2.4.

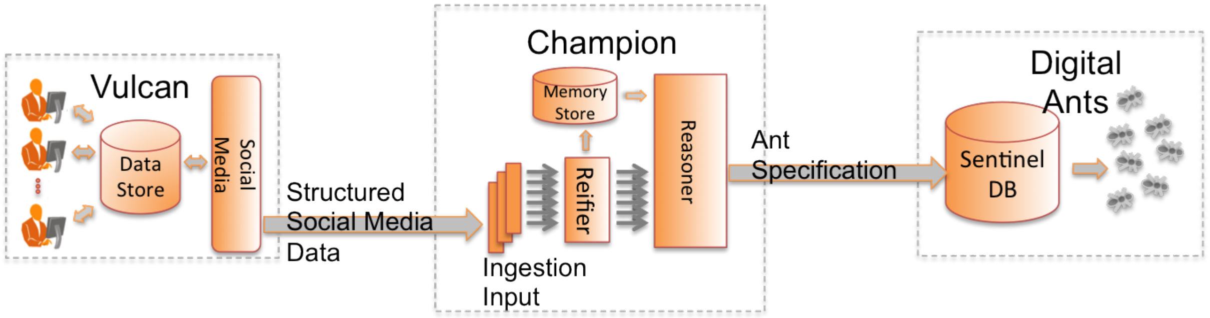


Figure 2.4: Block diagram of how ambient collaboration might be achieved. Vulcan enables collection of data and a collaboration framework to analysts, the CHAMPION reasoner digests the structured data coming from Vulcan and creates digital ant specifications that function as standing queries in the digital ants system.

CHAMPION uses auto-associative memory columns (AMCs) to encode domain-specific knowledge into an ontology and reason over structured input data. In this particular instance, we have a structured data input stream provided by Vulcan that needs to have some domain-specific knowledge encoded in it before it can be used to create digital ant specifications. By leveraging the domain-specific knowledge encoded in the CHAMPION ontologies, an ant specification can be generated as it reasons over the input from Vulcan.

Given a set of these ants, we can release them randomly in an enclave and see how they perform. If they trigger swarm formation that leads to identification of other problems, they may be considered successful standing queries. Those that never trigger useful swarms will slowly be forgotten by the system and will fade from use.

Contributions This work has provided a roadmap for future extension of the Digital Ants Framework to include Ambient Collaboration by integrating Vulcan and CHAMPION into the DAF and providing a design and requirements basis for this integration. It has also produced a publication[1][4]. Combining the strengths of each of the three tools described above, Digital Ants, Vulcan, and CHAMPION, we can provide ambient, unobtrusive, and powerful analysis aids to cyber security analysts.

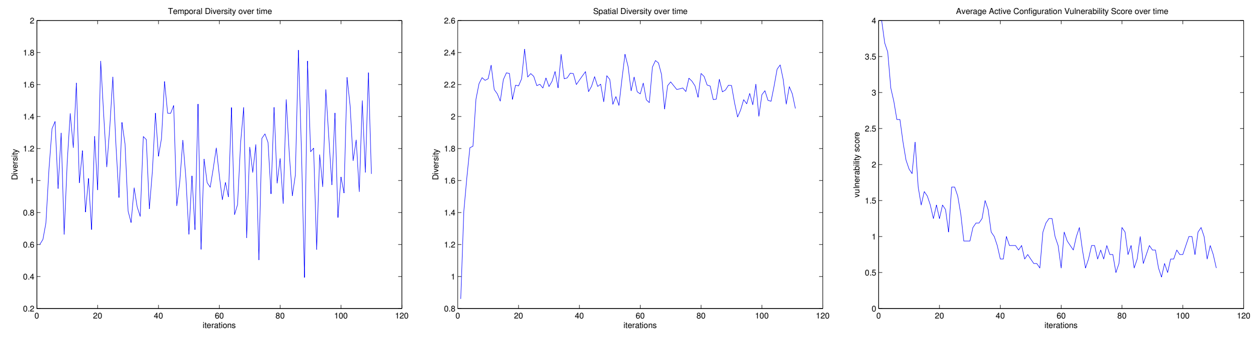
[4]See file VisualCol-2012-Shopping-for-Danger.pdf in the Extras/Papers directory of the accompanying CDROM.

2.2 Demonstrations

2.2.1 Wake Forest Automatic Defense and Reconfiguration Demo

Our partners at Wake Forest University integrated their Moving-Target Reconfiguration genetic algorithm into the Digital Ants Framework for a demonstration of how the system adaptively creates new configurations on real machines. Their setup included 16 Ubuntu 11.10 (desktop version) computers on the DHS DETER cyber range. Hooks were added to enable the algorithm to change OS configuration parameters on the fly. The fitness of each configuration was determined using a remote scoring server.

The team derived eleven OS configuration parameters based on results of the annual Wake Forest Hack-Event. Five of these impact security by creating or fixing a known vulnerability: file permissions, file ownership, net cat backdoor, multiple root users, password strength. Six parameters provide only diversity with no impact on security or performance: login banner, TCP buffer, max file, max open files, process limits, port range. Initially all the machines had identical poor configuration, but by the end of the simulation the population showed a diverse yet uniformly secure set of configurations (illustrated in Figure **??**). During the Fall of 2012, the GA will participate as a defender in the annual Hack Event against human opponents. A more complete account of this work is contained in deliverable report[5].



(a) Temporal Diversity (b) Spatial Diversity (c) Vulnerability

Figure 2.5: Average temporal (2.5a) and spatial (2.5b) diversity of the demonstration machines increased while vulnerability (2.5c) was reduced.

2.2.2 PNNL Scalability Demo

The purpose of the scalability experiment was to verify that the DAF would work on real systems at realistically large scales as the simulation models had indicated. The independent variables for the scalability tests were (i) the number of nodes, (ii) the number of sentinels, (iii) the number of rows and columns of the geography, (iv) the sensor to sentinel density ratio, (v) pheromone bolus amount, (vi) pheromone decay rate, and (vii) the number of steps an ant will drop pheromone after receiving a bolus.

Although our goal was realism, it is important to note that our experiment was quite different from an actual deployment. In a real deployment, we would expect to subdivide the set of Sentinels into enclaves of size $O(10^2)$ rather than placing them all in a single enclave. Our layout relies on *a priori* knowledge of how the Sentinels will be laid out so that they can address each other, while in a real deployment, we will likely use a layout method like Vivaldi[12]. The result of these two departures from operational reality actually provided a more pessimistic presentation of the performance of the DAF than would be expected in a deployment. Thus, we were willing to make these simplifications to provide a lower bound on performance.

Experiment Description and Results The dependent variable was the aggregate CPU time required to run the DAF. We desired to see very low average CPU utilization on each node (after the initial start-up period). Because we were partitioning each real machine on the PIC cluster into as many as 400 lightweight virtual machines to emulate a very large infrastructure, we expected CPU utilization to be very high for the DAF. However, we noticed that each DAF instance was taking only an insignificant fraction of the CPU time (typically less than 1%). To keep the

[5]See file D09-GA-Perf-Analysis.pdf on the included CDROM

activity level of the ants even, we ran tests using only the Bioinformatic Classification ants, with a single target dropped shortly after the system was initialized.

We ran tests with 10K, 20K, 30K, and 40K Sentinels. During the course of these runs we encountered several challenges including competition for usage of the PIC, large-scale outage of the cluster, and even the loss of a technician who was principally responsible for the CPU utilization metrics. Despite these obstacles, we were able to produce results that indicated that the digital ants are indeed scalable to moderate to large enterprises.

Lessons Learned In our original rendition of the DAF, we had included a one-second delay for each sensor at each node to reduce the amount of traffic and to prevent ants from creating a denial-of-service condition in the protected network. However, we discovered that, coupled with our queuing strategy, this delay resulted in a *cumulative* delay as ants queued up to leave each Sentinel. The cumulative delay caused some ants to spend nearly a minute transitioning between adjacent nodes which, in turn, caused pheromone to evaporate before other ants reached it. To overcome this, we increased ant densities to between 10% and 30%, as much as ten times higher than simulations had indicated would be necessary. This allowed swarming to occur, but also resulted in longer queuing delays, especially near the target Sentinel.

However, examination of the amount of CPU time expended by each sensor agent for actual processing did not indicate that the artificial delay was saving much time, so we removed the delay to see what the effect would be. We expected more uniform coverage from much lower ant densities without significant increase in CPU utilization per node, and this is what did result.

Additionally, we discovered several artifacts of running the DAF on a cluster computer that interfered with our experiments. First, the transparent, LUSTRE, file system underlying the cluster had limitations when all 40K sentinels were simultaneously writing log files to it. This volume of file writing is not an operational requirement of the digital ant system, but is necessary for performing experiments where detailed knowledge of ant behavior is needed. So this is not a bottleneck for operational use. LUSTRE also complained when each of the 40K virtual machines tired to run its own separate copy of Python locally. However, we were able to mitigate this by placing separate copies of Python on each node. Artifacts such as these will not be a problem when running one Sentinel per machine in a real enterprise, but they illustrated the dangers of assuming that a particular job control strategy will work on a given high-population emulation. These lessons will be very valuable when running the DAF on the client's testbed system.

2.3 Publications

This work resulted in a number of papers that were written, some of which were accepted for publication. Those accepted for publication included:

- Michael Crouse and Errin W. Fulp, "A Moving-Target Environment for Computer Configurations Using Genetic Algorithms," In *Proceedings of the 4th Symposium on Configuration Analytics and Automation, 31 Oct 2011.*
- Fink, Glenn A.; Oehmen, Chris; Haack, Jereme; McKinnon, A. David; Fulp, Errin W.; Crouse, Michael B. , "Bio-Inspired Enterprise Security," Self-Adaptive and Self-Organizing Systems (SASO), 2011 Fifth IEEE International Conference on, pp.212-213, 3-7 Oct. 2011
- Crouse M, GA Fink, JL White, EW Fulp, KS Berenhaut, and JN Haack. 2011. "Using Swarming Agents for Scalable Security in Large Network Environments." Invited paper in *Proceedings of the 54th IEEE International Midwest Symposium on Circuits and Systems.*
- Bruce J, GA Fink, "Shopping for Danger: E-commerce techniques applied to collaboration in cyber security," in *Proceedings of VisualCol 2012 workshop*, IEEE Press, Denver, CO.
- Fink GA, KS Berenhaut, and CS Oehmen. 2012. "Directional Bias and Pheromone for Detection and Verification on Networks." In Sixth IEEE International Conference on Self-Adaptive and Self-Organizing Systems. PNNL-SA-87555, Pacific Northwest National Laboratory, Richland, WA.

Additionally, we have a number of planned that stem from this work or follow-on work related to this project:

- A short article in Transmission and Distribution World professional journal
- An article on scalability of the DAF for Supercomputing or TPDPS
- A RAID or Security and Privacy paper comparing digital ants to IDS

- Two book chapters on digital ants in a planned book titled, Movement on Networks, edited by faculty from Wake Forest University

This work has thus provided many contributions to the state of the art in agent-based systems and cyber security. We plan to build on this legacy in the future.

Chapter 3

Path Forward and Conclusion

The digital ants framework that made all this work possible is far from complete. Many enhancements remain to be done, and much work will be required to prepare it for operational use. It is our hope that the research investment of NITRD will be built upon by other agencies and companies who see promise in this new approach to cyber security of large enterprises.

Swarm Detection Digital ants is intended to be a "lights-out" system that does not need constant monitoring by human analysts. Without this ability, it will never be able to scale to large enclaves and enterprises. Thus, we desire our system to be able to detect when swarms are forming and notify the human operator or take appropriate action automatically. To do this, we must develop reliable metrics for detecting swarm formation. Swarms are not simply large numbers of agents appearing on a machine. Instead, they are a function of the number of recent detections on adjoining nodes, the amount of pheromone pointing in the direction of the swarm, and the gradient of pheromone in the enclave. Finding the proper mix of detectable conditions and finding the right, decentralized way to detect them is a subject for ongoing work.

Ambient Collaboration While Ambient Collaboration was investigated in this work, no implementation was derived. We would like to implement and test Ambient Collaboration in future versions of the DAF. We believe that Ambient Collaboration would be very valuable to our clients who must investigate cyber events that are distributed throughout the world in environments where effective sharing is difficult. We propose applying a semantic reasoning engine, like PNNL's CHAMPION reasoner to convert streaming tokens from the Ambient Collaboration capability into sensor ants that may be used to find matches to queries of concern to the cyber analyst. Enabling Ambient Collaboration would be a game-changing technology that would provide an important advantage to cyber defenders.

Population Dynamics One of the most adaptive features of the digital ants is their ability to dynamically manage the population of agents in response to need and available resources. For this project, we concentrated on simply creating an implementation of the DAF, and we saved population dynamics for future work. This feature of the digital ants must be enabled to make the ants work operationally. Particularly, population dynamics would prevent attacks where an adversary might attempt to partition the network and seal off all the ants from the area where he would like to work unobserved. Additionally, population dynamics prevents the digital ants themselves from producing a denial of service effect on the defended network.

Detecting polymorphic malware Our bioinformatic classification has shown itself very flexible in discovering target binaries despite large changes to their structure. However, we believe that concrete demonstrations of this capability against real polymorphic malware are necessary to show measure the actual effectiveness of the system against realistic threats. As malware develops more and more complex anti-forensic capabilities, an actual trial of the system against these adversaries is the only way we can determine the efficacy of our approach.

Large-scale demonstrations While we have prototypically shown that DAF can scale to large enterprises, we must increase the scale of these by one or more orders of magnitude, and concomitantly, we must increase the fidelity of these demonstrations to include the entire DAF and its full set of capabilities. This will enable us to make claims of efficacy with quantified confidence in the features of the DAF.

Higher-level reasoning In this project, we have necessarily limited our scope of implementation to the Sensor (ant agent) and Sentinel level. Even our implementation of the Sentinel has been necessarily simplistic. The DAF can accommodate much higher level reasoning at the Sentinel and Sergeant level than we have yet implemented. These higher level functions could be accomplished by fusing digital ants with other reasoning software such as the CHAMPION reasoner or the KAOS policy engine. Such high-level reasoning will enable the system to be much more usable by humans and will make the system a better collaborator with human analysts.

Smart grid integration A follow-on project that uses the DAF is investigating using digital ants to protect the smart electric grid from cyber and physical threats. After the conclusion of this project, much will still remain to make the DAF ready for implementation in end-user devices, etc. We hope to find clients in the commercial and government sectors who will partner with us to use digital ants to protect these critical infrastructures from malice.

Conclusion While this project has amply demonstrated the promise that solutions like digital ants holds for cyber security and other applications, much work remains to make the approach tenable. We have shown that this approach can scale dramatically and that it can enable monitoring of large systems with little reliance on human interaction. This effort has advanced efforts to secure digital infrastructures by developing a dynamic means to adaptively defend complex cyber systems. We hope that this work will benefit both our client's efforts in system behavior modeling and cyber security to the overall benefit of the nation. To that end, we intend to continue work on the digital ants and seek partners and clients to continue pursuing these worthy goals.

Appendix A

Milestones and Deliverables

This appendix summarizes the milestones and deliverables of the Bioinspired Approaches to Moving Target Defense Strategies project. The milestones and deliverables in the statement of Work (SOW) were divided into research tasks as follows:

Task 1: Providing a Robust Moving Target Environment using Mobile Agents and Distributed Genetic Algorithms

Task 1A: Concept Modeling and Initial Explorations of Mobile Agents and Distributed Genetic Algorithms

Task 1B: Combine the Social Insect Genetic Algorithm Elements

Task 1C: Advanced Prototype Implementation/evaluation

Task 2: Biology-based classification, recognition, and sequestration of cyber entities: shifting the burden of complexity onto the attacker

Task 2A: Selection and Preliminary implementation in Applications Area.

Task 28: Formalize the Approach for Selected Area

Task 2C: Test the Prototype Implementation

In concert with the technical advisor, PNNL translated the SOW into a schedule of milestones and deliverables that traced back to the original tasks and deliverables and reported progress in the monthly DI-MGMT-80368 status reports. The final DI-MGMT-80368 report contains the mapping of each project milestone and deliverable to the project schedule, a final tallying of these results, and the completion dates of each milestone. The dependencies of these milestones and deliverables numbered as in the management report is recapitulated in Figure A.1.

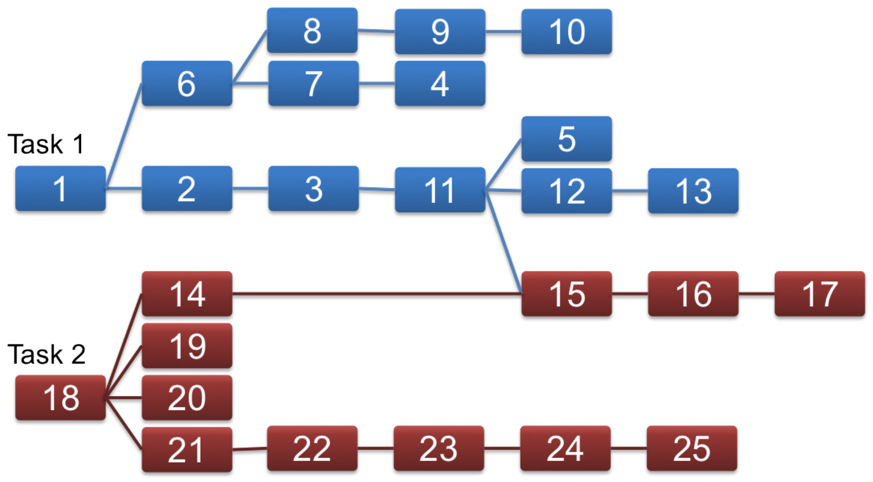


Figure A.1: Milestone and deliverable dependencies.

Further, for the ease of access to deliverables, PNNL has agreed to provide a CDROM recapitulating all publications, graphics, movies, and software[1], indexed by the task numbers listed above. All these items, with the exception of the final report have been previously sent with the monthly progress reports, but the CDROM will provide a convenient index of all the products of the project. Some of the deliverables have been recapitulated in this report, others are in files on the CD. The file names and locations of all the deliverables are summarized in Table A.1 below.

Table A.1: Bio-inspired Approaches to Moving Target Defense Strategies deliverables by file and location. Note, only deliverables, not milestones are listed in this table.

#	Description	Location
D01	Draft publication of approach methodology.	D01-SASO-Poster.pdf
D03	Complete initial measures, develop benchmark platform and threat model.	D03-Scenario.pdf and Appendix B
D04	Produce a performance analysis and simulation results publication.	D04-mwscas2011.pdf
D05	Deliver prototype of social insect concept.	(part of D17)
D08	Combine methodology of approaches and deliver prototype.	D08-DES.zip
D09	Perform a preliminary assessment of combined approach, and choice of testing/evaluation testbed.	D09-GA-Perf-Analysis.pdf
D10	Draft a publication and deliver DRAFT publication describing results and preliminary tests	D10-SafeCon-2011.pdf
D13	Draft publication implementation approach and results.	D13-SpatialComp-2012.pdf
D14	Deliver report that details software sensor specifications, system architecture for fragmenting sequence data and interface between social insect modeling platform and string matching archive system.	D14-MLSTONES_ant_specs.pdf
D16	Deliver test suite and validation platform.	(part of D17)
D17	Deliver a prototype of the integrated system (social insect modeling and string matching archive).	D17-LeapDaf.sparseimage
D19	Deliver a brief paper describing the selection process, reasons for final selection, and methodology to be used going forward.	D19-Process-Justification.pdf
D20	Demonstration of simulation of digital ants application in this area.	D20-Demo-Results.pdf
D21	Deliver paper describing the vectorization strategy and the prototype ants application vectorization strategy with figure of merit.	D21-Vectorization-Strategy.pdf
D22	Deliver a prototype of the ant application using an appropriate implementation software language.	D22-mlstones_ant_sim.tar.gz
D25	Deliver a report describing the reference implementation, test process.	D25-SASO2012.pdf

[1]All software files included are research prototypes. They are provided as-is and not guaranteed to be suitable for any particular purpose.

Appendix B

Bio-inspired Approaches to Moving Target Defense Strategies Operational Scenario

This scenario describes use cases that provide a visionary endpoint for which the Bio-Inspired Approaches to Moving Target Defense Strategies project deliverables are a significant step. The scenario focuses on situations where the objective is defending an enterprise- computing infrastructure from cyber attack through a variety of attack vectors. We pay particular attention to the need for detecting malicious capacity of executable code and compromised computing elements. We assume that humans will always have a role in the decision making process, but that we can enable them to make rapid, well-informed decisions through delocalized sensing coupled with transparent (i.e. no penalty for false positives) secondary responses.

B.1 Defender-side scenario description

Goals The primary goals of the Digital Ants framework are to (1) enable digital ant applications of various kinds to operate simultaneously over a common topological and logical domain; and (2) demonstrate this capability on two different but overlapping applications in enterprise network security (malware detection and optimizing system configuration).

Domain For this project, our domain of interest is enterprise IT systems where each node represents a single host (server, desktop system, laptop, portable device, or other networked device). At scale, this will model a large (100,000+ systems), globally distributed networked enterprise with complex, hierarchical structure (via enclaves, etc.). Over this real network structure we will construct a logical network topology for which each node has a predefined number of neighbors. This will allow the digital ants to traverse the enterprise in a regular 2-D mapping that obeys regular Cartesian geometry, and over which pheromone signals can be deposited.

Framework The digital ants framework at the core of the project will support logical mapping of a modeled globally distributed enterprise IT infrastructure onto a regular 2-D domain across which digital ants of different types will traverse. The framework will be demonstrated using digital ants of two basic types: (1) ants that have sensors for detecting malicious software in the file systems resident at any node; and (2) ants that transport candidate configurations of high fitness from one host to another to support genetic-algorithm implementation of configuration optimization. Each type of ant will have its own sensor, fitness function, pheromone rules and interactions, and attributes. Both ant types will traverse the same logical topology simultaneously and alert sentinel and human users regarding their behavior and results.

B.1.1 Application scenarios

(i) **Malicious code detection** The primary unit of analysis that malicious code detection ants will perform is a calculation of similarity between translated representation of files on a host (node) with respect to a list or sub-list of motifs that represent malware families. Ants may be created that "specialize," meaning that they look only for one or a few types of malware signatures. Alternatively, ants may search against all malware signatures at a high level, and then produce more specific ants that look for subfamilies within that family. Computing capacity and memory available to ants will be the primary determinant in which strategy is used. We want to maximize the chances of identifying "bad" files without impacting users on a system. Ants finding malicious code will be rewarded (and hence "live" longer), and deposit pheromone trails pointing back to the infected node attracting other ants. In the case that ants are specialized, swarming will be a means to suggest to a sergeant or sentinel that multiple infected files are present on a node. In the case that ants are not specialized, swarming will indicate that multiple families of a malicious code instance are present on a node.

(ii) **Configuration optimization** The primary unit of analysis for configuration optimization will be to use genetic algorithms to discover a configuration that will most likely result in survival of a node. Each node will have many candidate configurations locally generate from ongoing genetic algorithm calculations. However, to prevent local minima from obscuring solutions with high degree of fitness, digital ants will be used to transport candidate configurations between systems. The expectation is that this will ensure highly optimal configurations will be seeded into the candidate list of other nodes once they are found, thus increasing the likelihood that more globally optimal solutions will be reached at each node..

B.2 Threat-Side Scenario Description

Key Features:

- Industrial espionage is the motivation.
- Attacker has no practical limits on means and expense of attack.
- Information available to attacker is limited to publicly available reconnaissance data and what can be obtained via social engineering.
- Adversary is patient.
- To win, the attacker's identity must not be revealed.

The attacker may exploit the following system vulnerabilities:

- Patching is difficult to automate in large, diverse systems, and humans may be unreliable.
- Using COTS security products makes it easy for attackers to analyze possible security system flaws.
- Legacy compatibility limits security system capabilities.
- Security systems may share the same host with the asset that is the target of attack (TOA).
- Overloading the security mechanism by a simple multithreaded attack may interrupt the operation of the asset.
- Security systems are usually software based, with no autonomic recovery mechanisms.
- Crashing the security system may expose the TOA.
- It is very difficult to deeply analyze network behavior at runtime and correlate changes of behavior together on a multi-enclave system.
- It is too hard to contain attack diffusion in a large complex internetwork.
- Most competing companies are reluctant to share information due to fear their proprietary information may be violated.
- Attacking the system is a low-risk activity for a clever, resourceful attacker.
- A single attack may work on multiple similar systems.

Attacker tools

- Hardware fabrication capabilities.

- Zero-day system exploits.
- Social engineering methods to recruit insider agents via social networks.
- Well-trained and funded attackers.
- Stolen certificates and digital keys.
- Experimentation lab to mimic the TOA and the defense systems.

Attacker goals

- Steal industrial secrets.
- Operation disruption to cause losses at will.
- Persistent access to defender's systems.
- Avoid revealing the attacker's identity/affiliation.

B.3 Scenario Narrative

Players The attacker is a sophisticated hacker group hired by XYZ, Ltd. to conduct industrial espionage and stealthy denial of service against ABC, Inc. and related firms. XYZ has two goals: foremost it wants to steal proprietary information from ABC. Second, XYZ wishes to disrupt ABC's operations, giving XYZ a competitive advantage. The victim is ABC, Inc. and its competitors and business partners. ABC is a multinational organization. ABC has 100,000 employees distributed over a well-connected set of 200 branches that are distributed over 20 countries. These branches contain interconnected cyber and physical components. Some 120,000 PCs and laptops, 35,000 embedded, networked computing systems (process-control systems, robotic appliances, printers, etc.), and 1,000 servers organized in clusters comprise ABC's computing end devices. ABC also assigns smart phone PDAs to selected employees based on their job. There are 25,000 PDA/smart phones.

Ingress Vectors Social engineering is used to install Trojan horse downloader programs on systems within the ABC infrastructure. The malware modifies the machine configuration under the cover of an installation program [9, 3]. The malware may also surreptitiously use the microphones and video cameras of victim machines, and may establish unauthorized WiFi access points [17].

Establishing a Beachhead Malicious programs downloaded by the Trojan use passive network monitoring to identify key machines and map the network for further exploitation. The malware uses advanced packers to obfuscate the true nature of the executable. The malware creates the fake WiFi access points to perform man-in-the-middle attacks to gather information destined for the Internet. It waits for sounds and images with sufficient entropy to occur in its sensors and funnels the video and audio feeds from its sensors out the high-bandwidth corporate network after hours. The malware sends lists of available machines and network monitoring results to the attacker across the network periodically. When a machine is positively identified as a potential target, the malware conducts an active reconnaissance and attempts to subvert it using the trust relationships within the corporation to its advantage. It may also use credentials gathered from the access point or other reconnaissance activities to masquerade to targeted systems as a legitimate user. The malware slowly returns directory listings to the attackers who are looking for exfiltration-worthy content. The attackers use the information gathered to identify executives or employees who have access to and responsibilities for industrial process control systems. Once promising employees are identified, the adversary targets them for spear-phishing attacks meant to gain their confidence and further their infiltration of the target organization. On subverted machines, the attackers attempt to change the configuration and patch the operating systems so that other malware will be disabled or protected against. They do this to protect their own command and control network from disruption by other malware.

Exploitation The goal of social engineering attacks is to get malicious software on the victims' machines that will take over their machines and grant access to sensitive information and offline industrial control systems. The attackers take advantage of the users' privileges and access to change the configuration of these systems so that they will be easier to access from the outside or vulnerable to certain future attacks. When a system that talks to an industrial control system is identified and compromised, the adversary downloads malicious software to the controller that will cause small performance degradations and reduction in machine usable lifetimes.

Extension and consolidation Attackers seek to adjust security controls and configurations of systems that contain promising repositories of documents so as to retain access. They periodically use the Trojan program to download utility programs for accomplishing specific missions. Because degrading or destroying industrial process control system may not always be to the attackers' advantage, the attacker only activates this mission at a strategic time when it will be of greatest market advantage to XYZ, Ltd for their competitors to be disabled. The attackers are constantly monitoring the victim network and their exploited machines to ensure stability. Attackers try to remove evidence of their presence from machines that are compromised but found to be not useful toward further exploitation.

Command and control The attacker uses a botnet fielded from within the victim's network to accomplish his work. The botnet uses strong symmetric encryption with preshared keys (bot-to-bot) and the industry standard TLS (HTTPS to external machines) to protect communications in the botnet from being detected [8]. Only bots known to be on executives' systems are allowed to talk outside the company, and then only by hooking an executive's browser while he is actively surfing the web. Other bots relay their messages to the attackers through the executive machines to reduce the profile of the command and control network.

B.4 How Digital Ants Deliverables Combat this Threat Scenario

Digital Ants Genetic Algorithms will compare configurations across all monitored systems, noting anomalies and seeking a set of highly intrusion-safe configurations. Scalablast Ants will defeat the obfuscation induced by binary code packers and will search for motifs showing relationships of code artifacts to known malicious code. Behavioral indicators ants will note deviations from the normal among systems within the same enclave, highlighting anomalous behavior. Ambient Collaboration will enable analysts within the target organization(s) to share information anonymously, automatically searching each organization's computers for indicators of potential compromise by using search terms and contents gleaned from members' analysis activity.

Appendix C

Digital Ants Framework Architecture

C.1 Overview

The framework is written in Python, which was chosen for its flexibility, fast development cycle, strong library support, and its ability to be run of many different architectures and operating systems. The framework is meant to be just that: a mechanism to build domain specific DigitalAnts applications. As such it is engineered in such a way as to provide the majority of the boilerplate code, thus allowing the user to focus on the behavioral aspects of the agents.

The framework implements Sergeants, Sentinels and Sensors. Communication between the Sergeants and Sentinels is done with TCP/IP sockets, using plain-text messages. This can be changed in the future, but is appropriate for development where it is useful to telnet to a port and send commands.

Each Sentinel is identified by an IP address and a port number. A configuration file exists that the Sergeant uses to contact each Sentinel to join it to the Sergeant's enclave. In addition the Sergeant informs each Sentinel of its neighbors, which are also specified in the configuration file.

C.2 Getting Started

A 'developer' domain example is in subversion, called: 'dev'. This is a Python package at the same level an the digitalants package, which contains the framework. Inside the 'dev' application are implementations of all of the agent types as well as some data sources. These are not very useful beyond providing a mechanism to test while developing the framework. Hopefully they will also prove instructive.

To see how the all the pieces work together run the following from two different command prompts (known to work on Mac OS X and Linux):

Program 1 Starting the agent framework.

```
./start_agent.py dev sentinel Sentinel sentinel-config-1.xml
./start_agent.py dev sergeant Sergeant sergeant-config.xml
```

The first line starts a Sentinel and the second starts a Sergeant. *start_agent.py* is the name of a script that loads the agent definitions, creates a new instance with the specified config file (the last parameter) and runs the agent.

The second parameter indicates the package that contains the code. The third is the module that contains the agent, while the fourth is the class name of the agent.

The enclave specified in the sergeant-config.xml file actually contains five Sentinels. The implication being that the first line can be executed five times (each in its own 'terminal' window) with the config file modified appropriately (sentinel-config-1,2,3,4,5.xml).

Typing ctrl-c in the window that is running the Sergeant will shutdown the Sentinels and end the simulation.

C.3 Framework

Below the main aspects of the framework are described in some detail.

C.3.1 Agents

The Sergeant, Sentinel, and Sensor are all agents. There is an *AbstractAgent* class from which all of the agents derive. It contains a single abstract method execute that is where subclasses can customize the behavior of each type of agent.

AbstractAgent::execute is the main 'run loop' of each agent. In the case of the Sentinel and the Sergeant, this is executed as part of the external initialization of those classes. For the Sensor, it is executed by the Sentinel.

There are two scripts used for development to start the Sergeant and Sentinel: *start_sergeant.py* and *start_sentinel.py*, respectively. These each take the XML configuration file as a parameter.

C.3.2 Sergeant

The Sergeant is responsible for creating the network of Sentinels, creating and dispatching Sensors to the network and as a central logging location. An XML configuration file is loaded by the Sergeant that specifies the Sentinels (by IP address and port) and their neighbors. The file also contains the Sensor specifications.

The Sergeant has no default behavior, but this can be specified in a subclass. There is currently a development Sergeant subclass that sends a single Sensor of each type to a random Sentinel.

C.3.3 Sentinel

The Sentinel is responsible for managing an interface to data sources for Sensors to read, as well as receiving and executing Sensors. The Sentinel also packages and transmits (migrates) Sensors to its neighbors.

The Sentinel has an XML configuration file that specifies its ID (synonymous with the IP port) as well as its data sources. Upon startup the XML file is read and the data sources are instantiated and stored in a list by the name of the data source. The Sensor has a method, 'inspect_data' that the Sensors call. The method takes two parameters: the name of the data source and an optional parameter to pass to the data source. The Sentinel invokes the *get_value* method on the data source, passing the optional parameter and returns the value to the Sensor.

When a Sentinel receives a new Sensor it is added to an execution queue. If there are Sensors to execute it will run the execute method of the Sensor and then sleep for a configurable amount of time.The Sentinel has no default behavior besides managing Sensors and data sources. However subclasses of the Sentinel class can do any type of processing they like.

Data Sources Data sources are the means by which a Sentinel can provide data to a Sensor. Each data source is a subclass of *AbstractDatsource* which simply has an abstract method *get_value* which takes a parameter. The implementation can store values, make calls to the operating system, or do anything that the user deems necessary.

C.3.4 Sensor

Sensors inherit from *AbstractSensor* which has an abstract method execute. The base class provides initialization to assign a unique ID to each Sensor (this is one reason that Sensors are created at the Sergeant). In addition the base class provides a memorize method that allows the Sensors to maintain a limited memory (default is 5). [[This should be broken into a subclass, as not all Sensors will need this.]]

Each Sensor contains a reference to the Sentinel instance that it is currently visiting. This reference is managed by the Sentinel when a Sensor is received and unpackaged. The Sensor uses this reference to communicate with the Sentinel, e.g., to move to a new Sentinel or to query a data source.

C.4 Logging

The framework has a mechanism for centralizing all logging at the Sergeant level. *AbstractSensor* has a log method which takes a level and a message. This method calls into the *AbstractSentinel* log method which then writes the message to the Sentinel's logger. This logger uses a *DatagramHandler* which allows it to send its messages to a server. For the Sentinel, the *DatagramHandler* points to the Sergeant who writes messages it receives to its logger. Currently the Sergeant's logger writes to a stream logger, but this will be changed to a file handler.

The logging.conf file is used to setup the framework's logging. This determines the default log level, format of messages, etc. Additionally, the Sergeant's configuration file has a "loggingPort" attribute which is used to setup the server which listens for log messages. This port is handed to the Sentinel when it joins an enclave in order to setup its *DatagramHandler* to point to the Sergeant.

Appendix D

Porting the Digital Ants Framework

At this writing, the Digital Ants Framework is a research prototype for very large infrastructures that is not yet suitable for portable installation and running. However, this appendix outlines some of the requirements for porting the DAF to another machine for testing and demonstration of this sort. The list of dependencies is not complete since porting is a very delicate, manual process, currently. Additionally, it makes little sense to run the DAF on very small networks (less than about 100 machines) because even at low sensor densities they quickly become saturated obviating the need for pheromone. Preferably, the DAF would be run in large systems with 10,000+ machines, but such a test environment is difficult to obtain. However, we have made extensive use of simulations and virtualization on cluster-type supercomputers to test the utility of the DAF at scale.

D.1 Running the DAF in a Cluster Computing Environment

We currently run all our large-scale runs of the DAF on the PNNL Institutional Cluster (PIC) computer. We will eventually migrate it to other clusters for testing that is not appropriate on the PIC, but we have adapted the DAF for this environment just for this purpose. Here are some minimal requirements for a cluster computer to run the DAF:

- System must have Python 3.2.2 and Perl 5.8.8 or later installed
- Must be a multi-node Linux cluster, running Red Hat Enterprise Linux
- Must use Infiniband interconnections with MPI communication library
- Must allow Ethernet sockets to connect nodes of the cluster
- Must enable a manual logical overlay network to be constructed over it so that all the nodes may be addressed by coordinate
- System must use Simple Linux Utility for Resource Management (SLURM) or equivalent cluster job management software

Adapting the DAF for a new cluster environment is a complex matter that involves a great deal of manual tuning. Cluster computation often relies on precise timing and predictable usage of input/output channels, etc. Running the DAF on the PIC has uncovered many difficulties, not related to the DAF itself, but to the cluster environment:

1. Running 10,000 copies of Python simultaneously brought the system down.
2. Writing thousands of files to the central storage simultaneously crashed the LUSTRE file system.
3. Writing multiple hundreds of files to the same node at the same time proved impossible.
4. Running more than 400 instances of the DAF per node failed quietly due to unexpected deadlock when more than two nodes were employed.

These examples illustrate the fragile nature of supercomputing environments that are optimized for high-speed processing, not as simulations of large computational infrastructures. However, through careful tuning of our use of the environment, we were able to overcome each obstacle (these and many more) in turn.

A general description of the batch processing that takes place to prepare the DAF to run on the PIC for ants run with just bioinformatic matching ants is as follows:

1. The SLURM sbatch command kicks off Perl scripts on each machine that is part of the enclave.
2. Perl scripts create the virtual machine infrastructure for the Sentinels to run on.
3. Perl scripts copy all the files and directories required to each virtual machine.
4. The Perl scripts launch all the Sentinels and Sensors on that node using the colonize and subcolonize scripts.
5. The colonize scripts fork “left” and “right” children and then execute themselves until all the required sentinels on the node are created.
6. The Sergeant script tells each Sentinel who its neighbors are according to the cluster’s physical and logical configuration.
7. The subcolonize scripts launch creating the appropriate number of ant agents on each node.
8. The Sensor agents begin to move around the enclave. The first ones begin to move before all the Sentinels are finished being created.
9. One can check to see whether all the Sentinels are running using `squeue | grep <userid>`. R means Running and it will show what node each job is actually running on. Log files are written to `/pic/scratch/<userid>/logs`.
10. After all the Sentinels are running, the user drops an infected file on a node of his choice.
11. After running for several hundred seconds more, the user terminates the job by sending the terminate message to each Sentinel.
12. Once all the Sentinels are terminated, the user must go to the `/pic/scratch/<userid>/logs` directory and type `./logmerge.csh <job_id>`.
13. Finally, the output can be collected and copied into the user’s scratch space by typing `./prep.csh <job_id>`.

D.1.1 The DAF Backplane

Because the very high scale DAF runs use multiple lightweight virtual machines to oversubscribe a cluster and emulate a large enclave, there are multiple Sentinels on each node of the cluster. Thus, all the Sentinels on a single node will report the same values for data sources that relate to the CPU, etc. To solve this problem, we have designed the Backplane, a simulation layer that reports realistic individual results for each virtual partition of each machine. The alternative for large runs would be to obtain tens of thousands of real machines to run on, and that was not tenable.

The DAF Backplane provides a simulated data layer for each sentinel. This backplane has the following properties:

- Ability to run different conditions for each simulated host: normal use, non-malicious errors, infected, etc. Together, the set of conditions produces a scenario.
- Communication channel separate from the platform’s to allow simulated nodes to communicate, e.g., for spreading conditions among backplane hosts.
- A means for backplane hosts to find their neighbors to simulate the spread of malware without reference to the digital ants’ geography.
- An internal store for data which DataSources require (CPU usage, file, network usage, etc.).
- Ability to add behaviors to the Backplane to modify hosts’ internal states to simulate user activity, background activity, and malicious activity.
- Data sources that take a time argument so that they only execute when an agent makes a data request.
- The ability to start a sentinel in either real or virtual mode so that the backplane may be disabled for one-to-one runs against real systems.
- A Shadow Sergeant that can pick a backplane instance and tell it to start the worm behavior program.

Backplane Communication As mentioned above, there is a backplane host for each sentinel, with multiple sentinel/backplane pairs per node. Since we have only one IP address per node, we assign each backplane host an IP address/port pair. However, since every backplane host needs to be able to send and receive messages from all of the other backplane hosts we have selected a particular port for at least one backplane host to listen to on each host. When a backplane host starts up it attempts to bind to this port. If the port is not bound, then this backplane host will become the "primary node backplane host." If the port is already bound, then we can assume that another backplane host is listening on that port. In this case, the new backplane host will bind to another port and send a message to the primary host telling it what port it is listening to. In this way, all simulated backplane hosts can communicate via an independent communication channel separate from the DAF.

Backplane Scenarios The backplane is started by the same script that starts the Sentinel and is passed the same information (ID number, address, etc.). Some decisions on which behavior to run may be based on ID numbers. For instance, assign browser behavior for IDs ¡ 100, and assign workstation behavior to IDs =¿ 100. Once the base behaviors are set up, the worm behaviors like spread rate, effect on backplane data sources, and deactivation conditions may be specified. The Scenario Coordinator monitors Sentinels for startup and sends a signal to some subset of backplane hosts to start the worm behavior. The backplane scenario plays out interdependently with the DAF with the backplane worms affecting what the DAF Sensors see and the Sentinels being able to deactivate worms under certain specified conditions.

By testing on a single core Linux Mint VM we gathered four sets of typical values that may be combined into backplane behaviors:

1. No active behaviors. Reported values:
   - CPU: 10% of single core
   - Network: No connections
   - Memory: 480M - 550M used
2. Web browsing. Reported values:
   - CPU: Up to 10% of single core when not loading pages or doing stuff. Up to 70% when interacting with the browser.
   - Network: Average of 20 connections
   - Memory: 670M used. (about +120M)
3. Compiling Linux kernel. Reported values:
   - CPU: 100% of a single core (split 70/30 between user and system)
   - Network: 0 connections.
   - Memory: 1000M (+450M)
4. Untarring Linux Kernel. Reported values:
   - CPU: 70% of a single core (60/10 split user and system)
   - Network: 0 connections.
   - Memory: 950M (+400M)

By combining these behaviors and selectively perturbing them when the worm arrives, we can simulate a realistic scenario in a large enterprise. While this chapter has not presented a thorough enough description of the DAF that it can be directly ported to any environment, it is our hope that it will assist those who want to port the DAF to a system of their choice to identify requirements needed to perform such a port.

Bibliography

[1] Bruce J, GA Fink, “Shopping for Danger: E-commerce techniques applied to collaboration in cyber security,” in *Proceedings of VisualCol 2012 workshop.*

[2] Carvalho, M. 2009. A distributed reinforcement learning approach to mission survivability in tactical MANETs. In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies (CSIIRW ’09)*, Frederick Sheldon, Greg Peterson, Axel Krings, Robert Abercrombie, and Ali Mili (Eds.). ACM, New York, NY, USA.

[3] Crenshaw, A., Programmable HID USB Keystroke Dongle: Using the Teensy as a pen testing device. Available at: `http://www.irongeek.com/i.php?page=security/programmable-hid-usb-keystroke-dongle`. Last viewed: 5 July 2011.

[4] Crouse M, GA Fink, JL White, EW Fulp, KS Berenhaut, and JN Haack. 2011. “Using Swarming Agents for Scalable Security in Large Network Environments.” Invited paper in *Proceedings of the 54th IEEE International Midwest Symposium on Circuits and Systems.*

[5] Crouse M and EW Fulp, “A Moving-Target Environment for Computer Configurations Using Genetic Algorithms,” In *Proceedings of the 4th Symposium on Configuration Analytics and Automation, 31 Oct 2011.*

[6] Cuppens F and A Miege (2002). Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*

[7] Fink GA, CS Oehmen, JN Haack, AD McKinnon, EW Fulp, and MB Crouse, “Bio-Inspired Enterprise Security,” Self-Adaptive and Self-Organizing Systems (SASO), 2011 Fifth IEEE International Conference on, pp.212-213, 3-7 Oct. 2011

[8] Golovanov, S. and Soumenkov, TDL4 Top Bot I. `http://www.securelist.com/en/analysis/204792180/TDL4_Top_Bot`. Last viewed: 5 July 2011.

[9] Goodin, D., Hackers pierce network with jerry-rigged mouse: Mission Impossible meets Logitech. Posted in *Enterprise Security*, 27 June 2011. Available at: `http://www.theregister.co.uk/2011/06/27/mission_impossible_mouse_attack/`. Last viewed 5 July 2011.

[10] Greitzer FL, and RE Hohimer. 2011. ”Modeling Human Behavior to Anticipate Insider Attacks.” Journal of Strategic Security 4(2):25-48. doi:10.5038/1944-0472.4.2.2

[11] Haack JN, GA Fink, WM Maiden, AD McKinnon, SJ Templeton, and EW Fulp, “Ant-Based Cyber Security,” in Proceedings of the 8th International Conference on Information Technology: New Generations. IEEE Computer Society, 2011.

[12] Dabek, F., R Cox, F Kaashoek, and R Morris, 2004. “Vivaldi: a decentralized network coordinate system,” SIGCOMM Comput. Commun. Rev. **34**:4, pp. 15–26, ACM, New York, NY, USA.

[13] Hohimer RE, FL Greitzer, CF Noonan, and JD Strasburg. 2011. ”CHAMPION: Intelligent Hierarchical Reasoning Agents for Enhanced Decision Support.” In Proceedings of the Sixth International Conference on Semantic Technologies for Intelligence, Defense, and Security (STIDS 2011), November 16-17, 2011, Fairfax, Virginia, vol. 808, ed. PCG Costa and KB Laskey, pp. 36-43. CEUR Workshop Proceedings, Aachen, Germany.

[14] Hui PSY, J Bruce, A Endert, GA Fink, ML Gregory, DM Best, and LR McGrath, “Towards Efficient Collaboration in Cyber Security” in Proceedings of the 2010 International Workshop on Collaboration in Security (COLSEC 2010), PNNL-SA-70532

[15] Ibrahim MAM (2006). Distributed Network Management with Secured Mobile Agent Support. In *Proceedings of the 2006 International Conference on Hybrid Information Technology (ICHIT '06)*, pp. 244-251.

[16] Kim H-A and B Karp, "Autograph: Toward Automated, Distributed Worm Signature Detection" In *Proceedings of the USENIX Security Symposium*, Aug. 2004.

[17] Kitchen D and R Wood, WiFi Pineapple, Mark II. `http://hakshop.com/collections/frontpage/products/wifi-pineapple`. Last viewed: 5 July 2011.

[18] Nojiri D, J Rowe, and K Levitt (2003). Cooperative Response Strategies for Large Scale Attack Mitigation. In *Proceedings of the 3rd DARPA Information Survivability Conference and Exposition (DISCEX)*, pp. 293–302.

[19] Papadopoulos C, R Lindell, J Mehringer, A Hussain, and R Govindan (2003). COSSACK: Coordinated Suppression of Simultaneous Attacks. In *Proceedings of DISCEX III*, pp. 2-13.

[20] Parunak HVD, P Nielsen, S Brueckner, and R Alonso (2007). Hybrid Multi-Agent Systems: Integrating Swarming and BDI Agents. In *Lecture Notes in Computer Science*, Springer, vol. 4335/2007. `http://www.newvectors.net/staff/parunakv/ESOA06Hybrid.pdf`

[21] Qin X, W Lee, L Lewis, and JBD Cabrera (2002). Integrating intrusion detection and network management. In *Proceedings of the Eighth IEEE/IFIP Network Operations and Management Symposium*, Florence, Italy, pp. 329-344, April 2002.

[22] Selfridge OG (1988). Pandemonium: a paradigm for learning. In *Neurocomputing: foundations of research*, 1988 MIT Press, Cambridge, MA, USA, pp. 115–122.

[23] Śmieja F (1996). The Pandemonium System of Reflective Agents. In *IEEE Transactions on Neural Networks* **7**:1, pp. 97–106. `http://ieeexplore.ieee.org/iel4/72/10170/00478395.pdf?arnumber=478395`.