A Cooperative Cyber Defense for Securing Critical Infrastructures

Glenn A. Fink* glenn.fink@pnl.gov
Jereme N. Haack* jereme.haack@pnl.gov
Wendy M. Maiden* wendy.maiden@pnl.gov
Errin W. Fulp† fulp@wfu.edu

Pacific Northwest National Laboratory* Richland, WA
Wake Forest University† Winston-Salem, NC

ABSTRACT

With the growth of Internet connectivity critical national infrastructures have grown intertwined in complex networked relationships. At electrical substations, it is common to find equipment from several companies together administered remotely by several contractors via the Internet. The the threat of nation-state and terrorist cyber attacks further complicates the *ad hoc* arrangement until the complexity of the situation becomes difficult even to describe [11]. Defensive actions and policy changes by one company may have far-reaching negative consequences on the partner organizations in the infrastructure. Currently, no cyber defense is designed to protect such interdependent multi-enterprise infrastructures.

Human-only or machine-only approaches are now insufficient. The former are slow but adaptable, while the latter are limited by their specialization. In either case, humans must accept ultimate responsibility for the actions of automated systems. We believe the solution lies in mixed-initiative [13] defense unifying the complementary qualities of both human- and machine-based approaches.

We describe the Cooperative Infrastructure Defense (CID), a new cyber-defense paradigm whose unique features are:

- CID makes humans an intrinsic part of the solution without requiring them to have direct control.
- CID enables diverse organizations within an infrastructure to cooperate in an adaptive cyber defense.
- CID unifies complex-adaptive swarm intelligence, logical rational agents, and human insight.

The resulting system actually turns false positives into beneficial forms of positive feedback for improved performance.

Keywords

Autonomic computing; security; rational agents; swarming agents; ant colony algorithms; mixed-initiative systems

1. BACKGROUND

Today's cyber adversaries can rapidly and concertedly disrupt multi-organizational critical infrastructures without any central coordination of their own. These critical infrastructures consist of multiple interdependent organizations (*e.g.*, the numerous private companies that make up North America's electrical power grids) that share computational resources in an *ad hoc* arrangement. The interdependence of organizations within the national power grid is so stringent that failure of a single company could produce cascading failures with serious international losses.

Several organizations may collocate their computer and network equipment at electrical substations to reduce costs and improve cooperation. Regulatory agencies, equipment manufacturers, and business partners may also have varying degrees of access privilege. Virtualization, distributed computing, and outsourced administration make it extremely difficult to determine who is responsible for maintaining which parts of which machines. Mergers are nightmares. Worst of all, organizations connect their critical control (*e.g.*, SCADA) networks to the Internet for convenience, making them vulnerable to outsider attack.

Regardless of the degree of their interdependence, infrastructure members remain economically independent organizations. They have separately managed systems, few shared policies, differing business drivers, and proprietary information that cannot be shared without special legal instruments. Accomplishing concerted defensive action across organizational boundaries is difficult, especially at Internet speeds.

Currently, most inter-organizational agreements are made between humans. Human managers interpret organizational objectives for human system administrators who, in turn, generate policies to run the enterprise. All involvement with external entities is mediated by humans. Most repairs require human intervention, and even given future autonomic systems [16], some configuration tasks will require human involvement.

Infrastructures have unique cyber security needs that cannot be adequately addressed by individual enterprise solutions. The complicated relationships in infrastructures imply that defensive actions by one may affect others adversely [11]. The legal issues alone [7, 23] are staggering. A successful infrastructure cyber defense requires managing the consequences of changes and coordination across organizations.

We believe that it would be useful for the constituent organizations to cooperate in the defense of their cyber infrastructure, avoiding actions that would harm partners or stakeholders. To manage the complexity, we factor infras-

tructures into *enclaves*, a set of computer and network hardware and software that is owned by a single organization and administered under a unified policy by a single (possibly separate) organization. An enclave may be very small, consisting of one or two machines, or it may be very large, with numerous networks. Enclaves are the building blocks of infrastructures and are the largest unit over which an automated system can be fielded without violating the rights or policies of a system owner.

System defenders need intelligent, autonomous defenses that can adapt in real-time to threats that originate internally or penetrate perimeter defenses. However, autonomous defenses cannot relieve defenders of their ultimate responsibility for the actions of their autonomous agents and the consequences of those actions. This implies that a mixed-initiative system [13] is needed where humans and rational software agents together regulate the system.

Current cyber defense systems do involve humans at multiple levels, but people are often too 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 who ought to cooperate may be unable to do so because of culture, language, legal, proprietary, availability, or other obstacles. Current systems cannot safely adapt at Internet-speeds to cyber threats. We must bridge the gap between human cognition and the autonomous activity of complex-adaptive systems of agents if we are to solve this problem.

2. INTRODUCTION

This paper presents a new mixed-initiative hierarchical framework of humans and agents that is well suited to protecting computational infrastructures. The framework, Cooperative Infrastructure Defense (CID), is designed to rapidly adapt to new cyber attacks via swarming software agents while enabling humans to supervise the system at an appropriate level. A hierarchy of rational software agents are employed between the swarm and the human supervisors to provide a channel for system guidance and feedback.

CID is designed to be a scalable, dynamic, and robust framework for securing the next generation of computational infrastructures. The key distinguishing features of the CID framework are:

- *CID makes humans an intrinsic part of the solution*, engaging humans without requiring them to directly control. Other approaches concentrate on machine intelligence, ignoring the need for human supervision. Still others include humans in the *wrong* loop, either too close to the problem so that the system speed is bounded by human reaction time, or too high above where the human is completely separated from the source of the problem or the means to solve it.

- *CID enables diverse organizations within an infrastructure to cooperate in an adaptive cyber defense.* All other known approaches are targeted at single organizations or enterprises.

- *CID unifies the complex-adaptivity of swarm intelligence, the logical certainty of rational agents, and the insight of humans.* Using various kinds of rationality actually turns false positives into beneficial forms of positive feedback and improve system performance.

The remainder of this paper is structured as follows. CID is described at a high level in section 3. Section 4 describes system design considerations for the CID framework, while threats to CID are described in section 5. Section 6 discusses related work, while possible directions for future research are described in section 7. Finally, section 8 summarizes the proposed CID framework and reviews the expected benefits.

3. SYSTEM OVERVIEW

In CID humans and various types of software agents share the responsibilities of securing the system. Figure 1 shows how one human can supervise a multi-enclave system with a few high-level agents and a large swarm of simple, autonomous mobile agents, called Sensors. Our terminology is as follows:

- Humans are called *Supervisors*. They provide guidance to and receive feedback from one or more enclaves but they are required to take action only in extreme circumstances.

- Enclave-level agents are called *Sergeants* each of which is responsible for the security state of an entire enclave. Sergeants may make service agreements with Sergeants of other enclaves.

- Host-level agents are called *Sentinels* and they are responsible for protecting and configuring a single host or a collection of identical hosts such as a cluster or storage network.

- Swarming agents are called *Sensors* that roam from machine to machine within their enclaves searching for problems and reporting to the appropriate Sentinel.

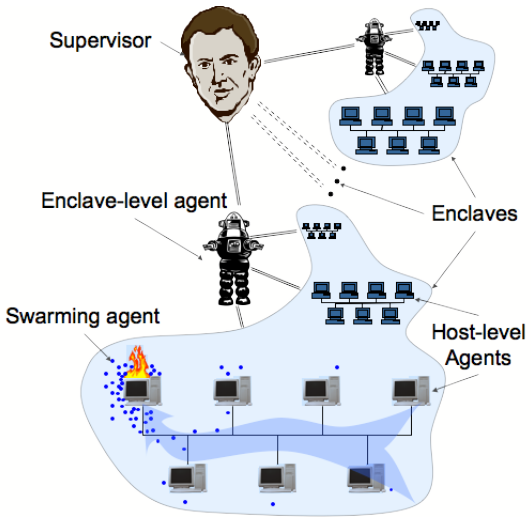


Figure 1: CID is a hierarchical framework of human supervisors, enclave-level rational agents, and swarming agents. A single human may supervise multiple enclaves via the agent hierarchy.

The concept of supervisors and agents of the CID framework operating within a hierarchical structure is supported by the research of Parunak, Śmieja and Selfridge [20, 26, 25],

each of whom suggested hierarchical arrangements of heterogeneous agents. Interposing logic-based, rational agents between the humans and the swarm provides a basis for communication. The hierarchical arrangement gives humans a single point of influence that provides multiple points of effect. The following sections describe the roles of each actor in the CID and the relationships between actors are depicted in Figure 2.

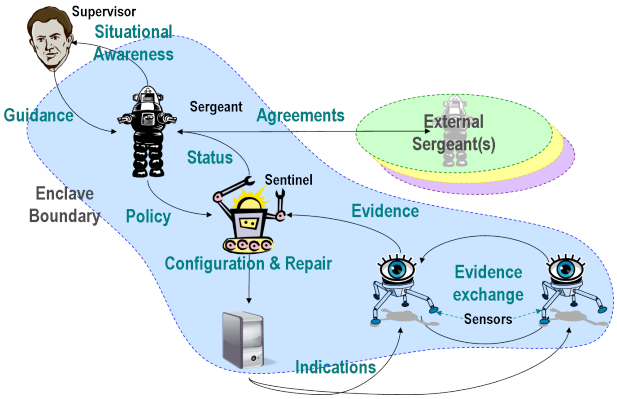


Figure 2: The relationships and interactions among humans and agents in the CID hierarchy.

3.1 Supervisors

At the very top layer of Figure 2 are human supervisors who may direct one or more enclaves. Supervisors may belong to one or more interdependent organizations within the infrastructure, while the cyber assets in every enclave all belong to a single organization by definition. A Supervisor might also be a member of a regulatory organization or law enforcement agency and only monitor the equipment in the enclaves. Human supervisors translate business logic into guidance via natural-language and graphical controls for the top-level agents. In turn, Sergeants provide situational awareness to their Supervisor, perhaps via an information visualization interface.

3.2 Sergeants

Each Supervisor is responsible for one or more enclaves each of which has a top-level agent called a Sergeant. Sergeants translate the guidance from the human Supervisor into actionable policy across all the machines within an enclave. Sergeants are "heavy-weight" rational agents that make decisions based on logic. They may be implemented as Belief-Desire-Intent (BDI) agents [22] or the like. The Sergeant explains the activities of lower-level agents to the human Supervisor and functions as an interface to influence system operation. Supervisors use Sergeants to enact environmental settings and policies that govern the general operation of the lower-level agents without controlling the lower-level agents directly.

Sergeants broker agreements between CID enclaves on behalf of the Supervisors. To ensure that their actions are properly attributable, Sergeants must have a separate digital identification from the Supervisor they report to. Since they negotiate on behalf of humans, they may incur liability for their owning organization. Thus, there must be a mechanism to describe the types and degrees of trust the Supervisor has placed in them. This trust mechanism is described in section 4.4.

3.3 Sentinel

Sentinels are mid-level rational software agents that correspond to today's notion of autonomic computing agents [16]. Each Sentinel is responsible for a single machine or group of machines that are identically configured. For example, a Sentinel might be responsible for a single server, a router, a storage area network, or a group of load-balanced web servers, etc. Sentinels implement the policy they receive from the Sergeant and apply it to the configurations of the machine(s) they manage.

Sentinels also interface with the lowest-level agents, the swarming Sensor agents. The Sensors gather information on potential problems, and the Sentinels fuse this information (along with contextual host information) into a potential solution. Although we do not specify the means of deriving the solution, we expect to use mechanisms similar to the Willow survivability architecture [15].

Sentinels give feedback on the utility of Sensor findings to the Sensors that visit their nodes. We use the analogy of foraging ants to describe the effect this feedback has on the system. By "feeding" visiting Sensors, the Sentinel will attract more of them. A variety of visiting Sensors will provide more information on the potential problems experienced by the Sentinel and enable it to make more informed decisions about how to fix the problems.

False positives from the Sensors are not a problem; the Sentinel should feed them unless what the Sensor reports is *known* not to be a problem. This strategy will attract more Sensors until a clearer picture can be constructed. When the Sentinel has a clear picture of the problem, it will make a report to its Sergeant. The report to the Sergeant will contain the classifiers used by the Sensors that helped diagnose the problem and any other evidence it has that may be useful in building new, more specific classifiers. In this way, *CID uses false positives to generate useful positive feedback and increase its problem-solving ability.*

3.4 Sensor

Sensors are lightweight, swarming, mobile software agents that roam and detect problems. They are modeled after behaviors of social insects and they employ "swarm intelligence." The Sensors' logic is as simple as possible; their power is in their numbers and their diversity. Sensors wander across the "geography" of the enclave by randomly adjusting their current heading similar to the movement of real ants. Each Sensor is programmed to match a particular set of conditions in the hosts they visit. There are two broad categories of Sensors: Markovian (memoryless) and differential. Markovian Sensors look for static conditions that may either define signatures of known problems or well- known anomalous conditions. Differential Sensors look for differences in conditions between hosts in recent memory and their current host. For example, there may be an unusual rate of network connections, a large number of open files, strange file names in system directories, or unusually high CPU utilization.

When Sensors find the conditions they are programmed to look for, they will report their findings to the Sentinel in charge of the machine and ask to be *fed*. The Sentinel may or may not feed the Sensor depending on how useful it

believes the information to be. When fed, Sensors become excited and leave behind a trail of digital pheromone [5] that may lead other Sensors to the Sentinel that fed it. Excited Sensors are more likely to spawn a new Sensor.

There are several ways Sensors may spawn: they could recombine classifiers with other successful Sensors similar to genetic programming, new Sensors could be generated by the Sentinels or Sergeant to solve specific problems, or the human Supervisor could design them. Determining the exact means of spawning is a matter for future work. Further details on Sensor implementation are located in section 4.

3.5 Example Operation

In the absence of an attack on a CID protected system, there should be very little pheromone and a relatively small number of Sensors. Sensors would be moving around the enclave in a directed stagger designed to get them to visit many systems in random order. When a virus or worm attacks a network protected by CID, changes would appear in the configuration or behavior of the infected machines. Perhaps there would be an unusual rate of network connections, an unusually large number of open files, files with strange names in the system directories, or an abnormally busy CPU. A Sensor programmed to notice one of these symptoms will report the anomaly to the Sentinel in charge of the affected machine. The Sentinel will assemble this new information with what it already knows, compare it to what it believes should be normal for that host, and try to diagnose the problem. If the Sentinel concludes that this is either a symptom of a problem or a significant anomaly, the Sentinel will reward the Sensor with some *food* in proportion to how certain it is that the finding is evidence of a problem. This will excite the Sensor and it will deposit pheromone on each host it subsequently visits. This pheromone gives the direction and distance to the host where the Sensor was fed, and the strength of the food it received.

As unexcited Sensors cross the pheromone path leading away from (but pointing towards) the affected host they are likely to follow it to find a host where food is available. These Sensors will contain different classifiers, so they will be searching for other kinds of evidence of problems in the host. For this example, the Sentinel collects more and more information along with its own observations and develops a diagnosis of the problem and a means to mitigate it. Sensors who visit and report a non-problem will not be fed to prevent spreading excessive pheromone and attracting more Sensors. The Sentinel reports to the Sergeant when it has identified what appears to be a virus on its host system and includes a report of the Sensors classifiers that led it to this determination. The Sentinel will also tell the Sergeant what steps it took to fix the problem.

The Sergeant will then build new Sensors with classifiers tuned to detect symptoms similar to those reported by the Sentinel. The Sergeant will log its actions and report to the Supervisor. Then it will send copies of the new Sensor out into the rest of the enclave. If the new Sensor type is reported effective, the local Sergeant may send the classifier to the Sergeants of other enclaves notifying them of the problem, symptoms, and reported resolution. This is essentially the process for creating a new "antibody" Sensor that functions as an infection signature.

Once the Sentinels repair the infected systems, Sensors passing through will not find any more problems, will not be fed, and will generate no new pheromone. Over time the existing pheromone evaporates away, and the system returns to its normal state with the Sensors foraging for the next "food" source.

At every point in this process where the Sergeant is involved, the human Supervisor is made aware of the situation. The Supervisor may step in and adjust the constructed classifiers, resolution instructions, etc. Supervisors may also instruct the Sergeant not to share its solutions or may adjust the Sergeant's trust rating so that other systems will require the Supervisor's signature before accepting classifiers from that Sergeant.

4. SYSTEM DESIGN CONSIDERATIONS

The CID framework consists of a cooperative, hierarchical system of humans and agents, where each entity in the system has a specific role for ensuring a secure environment. This section discusses important design considerations needed for the system to properly operate, such as the basis for the design of the swarming sensor agents, how they are managed, and the role of trust management in maintaining system integrity.

4.1 Alternative Sensor Interaction Models

Two major classes of social insects, ants and bees, serve as two very different models for the behavior of software agents. Ants use an indirect communication mechanism outside the nest while bees use direct communication in their hive. Ants explore their environment until they find food. After picking up food, they return to their nest while depositing pheromone on their return trip. Other ants will then follow that pheromone back to the food and deposit more pheromone, strengthening the path as they return.

As long as food can be found at the same location, the path remains strong. However the pheromone evaporates over time, allowing new paths to be discovered and adaptation to the changing environment to occur. Dorigo used a pheromone-based approach to solve the traveling salesman problem with greater efficiency and robustness than other methods.[9].

In contrast, scout bees explore their environment, find food, then return to the hive. At the hive, they inform forager bees of the location of food, giving a direction and distance. The forager bees then follow the directions to the food.

Lemmens has shown that while the bee model can be more efficient (since bees are directed towards food instead of relying on pheromone) it can be less adaptive. While a bee-based system can rapidly find the shortest path to a target, if that path becomes blocked they will still attempt to take it [17]. Meanwhile, if a shortest path becomes blocked for the ants, they will resume searching and lay down pheromone marking the new shortest path.

In the course of our research, we have built simulations of both the ant and bee models to determine which was most appropriate. We found that the bee model was more natural to the discontinuous geography of cyberspace since bees fly directly to the area where nectar has been found. However, a faithful bee model still requires the concepts of direction and distance that imply continuous geography. We settled on the ant model after developing a workable geography model (section 4.5) and when we saw how stable and intuitive pheromone-based solutions were.

4.2 Sensor Agent Management

Decentralized, pheromone-based systems have been demonstrated to simply and effectively solve highly constrained problems where logic-based, optimizing approaches prove intractable [9]. Figure 3 shows the simulation model of an pheromone based system we built to investigate tradeoffs among pheromone implementations.

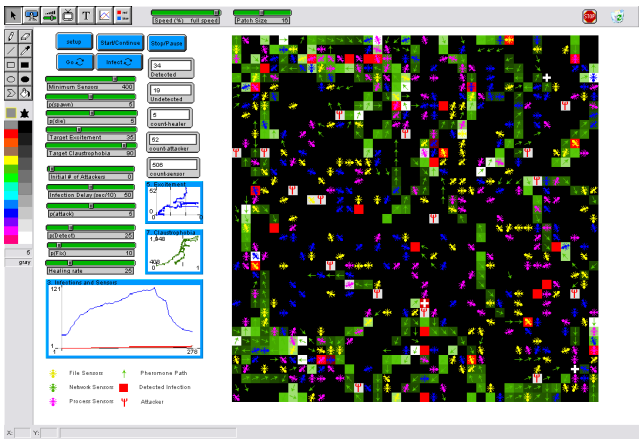


Figure 3: Our StarLogo simulator of a simplified CID implementation showing Sensors (digital ants), Sentinels (patches), and Sergeant (display).

Central to any system that relies on swarming sensors is the ability to communicate information and influence action. The CID framework uses a pheromone-based system for inter-Sensor communications that has been demonstrated to be simple and efficient [9]. This decentralized communication approach allows Sensors to not only achieve the local goal of finding food, but also collectively achieve the global objective of discovering inflected computers in a dynamic environment.

Pheromone-based techniques have been shown to be robust and therefore appropriate for dynamic applications, such as network routing [3, 6]. This is an important feature for CID since the number of computers and their security status (e.g. infected or clean) will change over time. As previously discussed, a Sensor that discovers pheromone has a probability of following it that is related to the pheromone strength. The rate of pheromone decay and how pheromone is potentially overwritten will impact the system adaptiveness, as discussed in [3, 6].

4.3 Sensor Agent Implementation

Sensor populations vary naturally depending on the level of threat and the degree of crowding they perceive. Each Sensor calculates these quantities independently, and they are used to determine whether the Sensor should continue, terminate, or spawn. Each Sensor has an excitement level ($0 \leq e \leq 1$), a claustrophobia level ($c \geq 0$), target levels for both excitement ($0 \leq e_{goal} \leq 1$) and claustrophobia ($c_{goal} \geq 0$), and cutoff probabilities for both spawning ($0 < s < 1$) and death ($0 < d < 1$). While excitement measures the amount of "food" they recently received, claustrophobia measures the perceived crowding (relative number of other Sensors encountered recently). The target levels are the goals each Sensor is trying to achieve. Having a target level for claustrophobia also causes Sensors to spawn when they have not encountered other Sensors for some time. This ensures that the population will not decrease below an effective level in the absence of infection, despite the lack of excitement in their lives.

$$\text{Sensor} = \{e_t, c_t, e_{goal}, c_{goal}, s, d\}$$
$$\text{Goal \#1: } min(|e_{goal} - e_t|)$$
$$\text{Goal \#2: } min(|c_{goal} - c_t|)$$

At every host a Sensor visits, the excitement and claustrophobia values are updated and the Sensor uses these values to determine whether it will continue to live, spawn or die. A host's Sentinel may feed the Sensor an amount, $0 \leq f \leq 1$ that is used to update the excitement and claustrophobia levels at that time.

$$e_{t+1} = \frac{(e_t * n)+f}{n-1}; \quad c_{t+1} = \frac{(c_t * n)+k}{n-1}; \quad n, k \in \mathbb{I}$$
where $n > 1$ weights the result toward past e,
and $k \geq 0$ is the number of other Sensors present.

Excitement levels near 1.0 indicate alarming conditions and trigger rapid spawning of new Sensors. Claustrophobia levels do not have an upper bound, but generally, above 1.0 indicates some degree of crowding. Next, the Sensor must determine whether it is excited enough to spawn. If $p(spawn) \in (0,1) < s - \sqrt[3]{e_{goal} - e_t}$, then a new Sensor is spawned. The cube root is used to amplify the distance between e_{goal} and e_t while preserving the sign of the difference.

Only in the case where no spawning occurs is there a test to see if the Sensor will terminate itself due to overcrowding. The rationale for this decision is that overcrowding is tolerated as long as there is plenty of food (*e.g.*, the threat level remains high). Thus, if the Sensor does not spawn, and if $p(die) \in (0,1) < d - \sqrt[3]{c_{goal} - c_t}$, then the Sensor terminates itself. The probabilities $p(spawn)$ and $p(die)$ are calculated at each host visited. Since the claustrophobia goal is only tested when the Sensor does not achieve sufficient excitement, the two goals do not conflict.

Sensor pheromone is deposited by Sensors at the Sentinel and consists of a tuple that contains a strength at the given time ($s_t > 0$) and a direction ($0 \leq direction < 360$). This differs somewhat from Brueckner's model (section 3.2.1 of [5]) in that his work used sensor gradients to tell direction rather than modeling direction explicitly. We have chosen an explicit representation because the values are stored on physically separate hosts, making gradient following much less efficient in our application. Pheromone evaporates (losing strength over time) and may propagate (spreading to adjacent hosts in the virtual geography).

A Sensor that discovers a pheromone trail would have a probability of following the trail ($p(follow)$) related to of the pheromone strength and the excitement:

$$p(follow) = 0 \leq f(s_t, e_t) \leq 1 \quad \text{where } f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$$

The pheromone will lose strength as it evaporates; however as previously described, the path would remain intact as long as other sensors find food at the host. A decentralized communication approach enables Sensors to achieve the local goal of attaining their target excitement level while simultaneously collectively contributing toward the global objective of discovering infected computers in a dynamic environment.

4.3.1 Initial Results

Using our ant model simulation, we were able to show that the excitement and claustrophobia parameters were necessary and sufficient to control Sensor populations and thwart simulated attackers. Figure 4 shows a synthesized version of a typical model run. In the figure, at times a, e, and g excitement is low and claustrophobia is high causing Sensor populations to decrease. At b and f excitement is high causing rapid growth in Sensor populations. At f, the spread rate of attackers was increased by an order of magnitude, but a sufficient quantity of Sensors control the infection before it spreads.

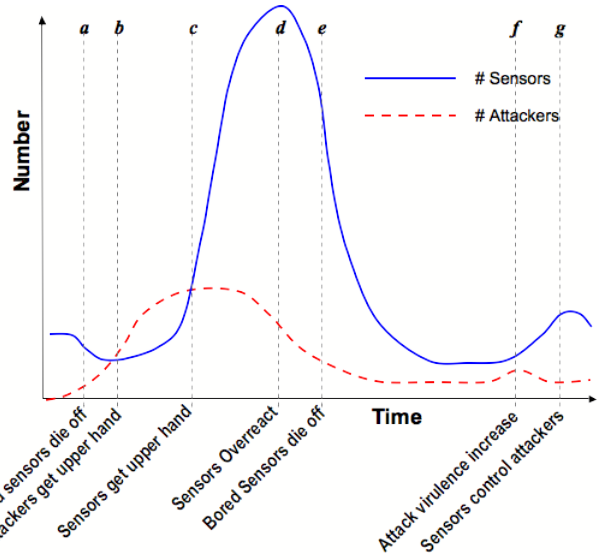


Figure 4: A synthesized plot showing typical interaction of simulated Sensors and attackers.

4.4 Trust Management

Although CID is a security system, the CID agents themselves cannot be implicitly trusted. Sensors depend on the computing resources of their hosts, but unlike most mobile agent systems, Sensors must seek out rather than avoid malicious hosts. This makes them subject to a number of potential threats, from both errors and adversaries, that could render them untrustworthy. Diversity and sufficient population prevent Sensors from being subverted *en masse*. It is difficult to subvert enough of them simultaneously to destabilize the whole system. The more privileged but non-mobile Sentinels and Sergeants may also be subverted or incapacitated, and Sergeants have with the additional responsibility of negotiating with Sergeants of other enclaves on behalf of its human Supervisor.

Trust management systems are generally based on reputation[18], credentials[2, 24], or both[12, 8]. Reputation systems are most common in e-commerce where the two parties do not know each other directly, but they can also be used to detect anomalous behavior in known entities [27]. Credential-based systems may use authentication and authorization via credential chains that anchor trust in a mutually-trusted third party.

Both forms of trust management are valuable in the CID system. Credentials may be used to authorize activity at all levels of the hierarchy as well as in the cross-domain interactions of sergeants. Reputation, on the other hand, can be used to detect subverted or non-performing agents throughout the hierarchy.

Because trust management can easily become a major complicating factor, especially for lightweight agents, we must establish acceptable levels of risk to balance trust with performance. Our design incorporates such tradeoffs. First, our trust management design parallels the CID hierarchy in performance and scalability. Sergeants observe Sentinels who in turn observe Sensors, and all reputation data flows up to the Sergeant for an enclave-wide perspective. Reputation indicators for Sensors and Sentinels collected in real time may include heartbeat tuples: $(host, agentID, timestamp)$ that indicate inactivity or unusual activity patterns. Automated post-mortem analysis of heartbeat tuples enhances the performance of the reputation system. The adaptive nature of the system leads us to expect that nonperforming Sensors would quickly die off, so unusual longevity of these may also be an indication of trouble.

Given that the Sergeant is less adaptive than the other sensors and assuming that the Sergeant is placed on a limited access, carefully managed system, we assume that the risk of it being subverted is comparably small and would be more likely to be detected by the human. Therefore, we feel implicitly trusting the Sergeant within its enclave is warranted given nominal safeguards.

Sergeants' cross-enclave negotations, however, cannot be implicitly trusted because of the potential security and economic consequences. Here, trust may require authorization credentials such as attribute certificates [10] that bind authorizations to an identity. For instance, if a Sergeant wishes to negotiate a change in the quality of service it is receiving from a partner enclave and there is a cost associated with increased service, the partner enclave could ask the Sergeant whether it had sufficient trust to negotiate the change autonomously. The Sergeant would present credentials, signed by its Supervisor showing the dollar amount (or some other unit) it is trusted with to make decisions of this sort. If the trust level was sufficient, the partner would go ahead with the change. If the Sergeant lacked sufficient trust, the partner could require the human Supervisor's approval. Supervisors would probably start out trusting the Sergeants very little and as the Sergeants demonstrate reliability in their dealings, the Supervisor may choose to augment their trust level. Symmetric key-based credentials with a trusted third party or public keys without certificates could also be used to minimize the overhead of trust management.

4.5 Geography

In the real world, insects travel along a continuous, three-dimensional geography. While there are discrete landmarks along the way, between any two points in space there is always another physical point. In cyberspace, agents travel from host to host with "nothing but net" in between. Direction, distance, the spread of pheromones, and other means of communication in the real world depend on a continuous geography. In cyberspace, however, the discrete "geography" causes many of the metaphors from the continuous world to break down. Organizations may have network layouts where nodes that are very close to one another physically are many hops separated on the network. While it is not likely that we can make cyberspace continuous, we can cer-

tainly map it to a discrete layout where at least direction and "geographical" distance are preserved. We may eliminate the discontinuities of the physical network by mapping the network diagram onto a grid, as seen in Figure 5, that also induces the idea of direction as if on a map.

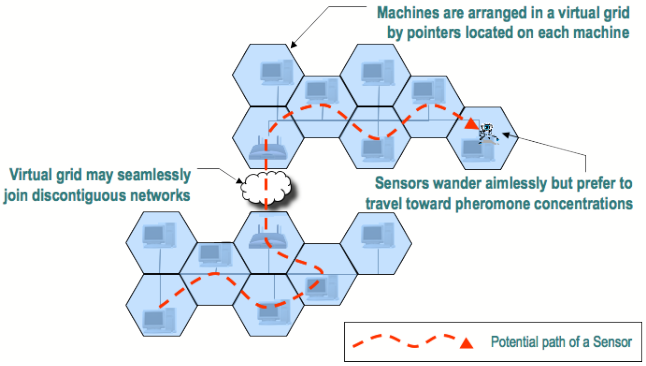


Figure 5: Mapping the network onto an ideal grid that eliminates the discontinuities of the underlying physical network.

Another problem with mobile agents in cyberspace is ensuring that they only visit hosts where they are invited. We do not want Sensors to query or attempt to travel to hosts outside their own enclaves. The virtual geography solves the problem since nodes that do not belong to the organization don't appear on the map. Sensors simply "can't get there from here." [1] The cyber geography, since it is always rectangular, can wrap around top-to bottom and left to right like a torus.

To translate a diagram of a typical network (Figure 6(a)) into a geographic grid, we number the nodes using a the breadth-first search starting at any arbitrary node. Next, we lay out the numbered nodes in the grid using a space- filling curve like Peano-Hilbert that preserves spatial relationships (Figure 6(b)).

The effect of using this layout is to minimize the network distance the sensors move when traveling from square to square and to reduce the traffic associated with Sensor migration. Machines on a common network segment are more likely to have similar characteristics, so keeping the mapping allows them to build up a characterization of the local network segment before comparing between separate segments.

The Sergeant maintains the mapping for its enclave. This geography is based on address data and registration data from Sentinels when they join the organization. The Sergeant periodically tells each Sentinel who its neighbors are. This design allows for a dynamically changing geography as hosts are added to and removed from the enclave. Armed with their portion of the map Sentinels can provide the IP addresses of the eight nearest neighbors to Sensors who inquire.

Although we do not yet implement it, the geographic representation allows for pheromone to spread via diffusion. Pheromone diffusion has made some of our early simulations run very slowly, and we suspect it may also use an inordinate amount of bandwidth to implement on a network. Additionally, we would have to decide what effect diffusion of pheromone with different directions would have on grid squares where other pheromone was previously laid. We will have to study geographical diffusion of pheromone to determine whether it is of sufficient utility to add.

[1]Sensors must use the facilities of a Sentinel to migrate, thus preventing runaway Sensors.

4.6 Non-Optimal Solutions

Typically when we write a program to solve a well-defined problem we prefer to find an optimal solution. This search for optimality has given rise to whole classes of problems for which optimal solutions cannot be computed in polynomial time in terms of the size of the input (i.e., NP-Complete problems). But in natural systems, robustness routinely trumps optimality. Ants individually appear to take nearly random paths around obstacles, but as a few succeed in finding a shorter path, others follow, and soon the colony has found a nearly optimal short path around an obstacle. If the obstacle is changed, the old path may become suboptimal, but the ants will follow their pheromone until some ant "randomly" stumbles upon a shorter path. Pheromone is naturally stronger on shorter paths. Similarly, bees communicate clearly but inexactly the location of the nectar they found. Cells that are about to become infected self-destruct rather than give place to a virus. Real life tolerates and often capitalizes on mistakes, imprecision, forgetfulness, and even death.

But in the world of cyber infrastructures, will one organization find it acceptable that its servers "give their lives" in the interest of the collective? Will we tolerate the decisions of digital ants when they cost us real money and uptime? It seems quite possible that humans may not share the automated system's opinion about what constitutes an acceptable loss. Will business executives tolerate suboptimal solutions while the system figures out how to neutralize a new threat? Will human supervisors be able to keep from interfering when the system seems to be slow to arrive at an "obvious" solution? Interference may seem like the way to quickly achieve the desired goals, but that may actually prevent the system from righting itself and finding a true near-optimal solution in the long run.

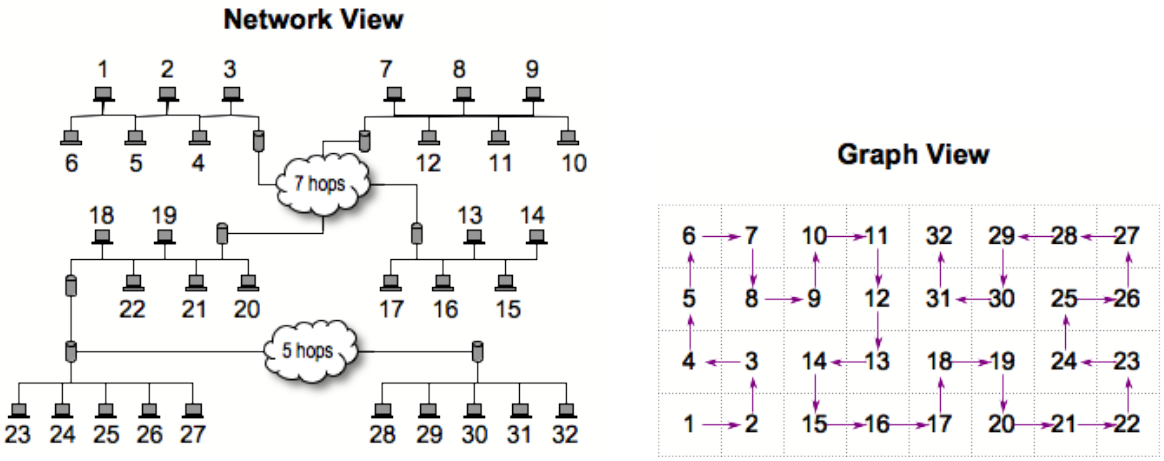
Security problems in the real world are ill-conditioned and highly resistant to the search for "optimal" solutions. So we have already taken on many risks simply because of the complex nature of networks and the undecidable problems involved in defending them. Ultimately, we believe the best solutions imitate the wisdom found in nature's designs even though they resist our abilities to analyze them thoroughly.

5. ATTACKING CID

No security solution can address all possible threats, especially attacks that target the security system itself. For example, current Intrusion Detection Systems (IDS) are often susceptible to denial-of-service attacks. Similarly, it is possible to attack the CID framework itself. This section describes potential attacks against CID and how they might be mitigated.

5.1 Adaptive Attacks

Adaptive attacks aim to make suspicious behavior appear to be normal by repeating it until the adaptive defense system ignores it. Suppose an adversary wished to attack a service that operated on an infrequently used TCP port. An adaptive approach might be to probe that port frequently without attempting to breach the system security. At first,



(a) Network view.

(b) Graph view.

Figure 6: Example network view and graph view of a system consisting with 32 hosts. The graph view of network was created using a breadth-first search and Peano-Hilbert layout.

the Sensors might notice unusual traffic, but when no other problem is found on the system, the Sentinel may choose to ignore this information, not feed the Sensors providing it, and thus extinguish an important clue to an upcoming attack. At some point the adversary would launch his exploit against the system hoping that the Sensors that should have detected his activity would have died out.

One solution to this potential attack is to compare status with other hosts in the enclave using differential sensors The intruder could circumvent this solution by similarly attacking all hosts in the enclave, but this should be more difficult.

5.2 Blackhole Attack

Since Sentinels receive the map of the enclave geography from the Sergeant, an attacker could modify the map at a subverted Sentinel so that all outgoing paths lead back to the same host. The integrity of the cyber geography of the CID system is vitally important to its proper operation. If an attacker was able to modify the maps and make some systems unreachable, then they would have free reign there. The ability to modify maps would also make many of the other attacks much easier to implement. The system must ensure that the geography at each node is accurate to mitigate this attack. Another potential mitigation would be to have the Sensors become aware of their path so that they would notice any recurring loops and find another way to escape.

5.3 Pheromone Trap Attack

A pheromone based attack uses the mechanisms of the social insect based system against itself. By manipulating the way ants drop pheromone an attacker could implement several attack strategies against CID.

By manipulating pheromone trails, an attacker could make a trap host very attractive to Sensors and when they are drawn inside they are terminated. This would decrease the number of wandering Sensors available to respond to other attacks. However, if the trap is too deadly it would keep ants from leaving the node and spreading pheromone to lead others to it. Then the attacker would have to subvert hosts around the trap and manipulate the pheromone stored on them. A modified attack would be to kill only half the sensors that visit and lavishly feed the others. The survivors would then lead other ants back to the trap where they cold be killed efficiently.

An attacker could also implement a disinformation attack by subverting two hosts, one of which is a decoy and the other the true target. By examining Sensor classifiers, the attacker could make a node exteremely interesting to sensors that it would then feed. While the ants are busy following pheromone to the decoy, there would be less chance of sensors noticing problems on the attacker's true target.

Pheromone traps are less effective since excited (well-fed) Sensors pay little attention to pheromone trails. This keeps Sensors from continuously returning to a host that just fed them. Another mitigation is that excited sensors always overwrite pheromone trails they cross so that legitimate trails would quickly replace maliciously laid pheromone.

5.4 Masquerading Host Attack

If attackers could insert a malicious host into the CID geography or subvert an existing host, they could inspect and modify sensors which visit it. Inspecting sensor agents would give the attacker an idea of what the sensors are looking for. Using this information, an attacker could implement an adaptive attack by ensuring that their attack is kept within a detection threshold. Inspecting the Sensors could also give the attacker the information needed to create malicious programs that use the CID's mechanisms to gain entry to other protected hosts. At the very least, a malicious Sentinel could confuse legitimate Sensors by rewarding them all the time causing them to lure other Sensors there too.

A mitigation might require Sentinels to send a cipher-text stream to the Sergeant as a credential to show that they were legitimate and to require the Sentinel to destroy the key for this stream if they ever believed that they were subverted. Of course, this solution gives rise to many other kinds of attack as well.

5.5 Sensor Subversion Attack

An attacker with the capability to subvert existing sensors or create a malicious sensor could use the mechanisms of CID against itself. Bad sensors could simply ignore signs of attack they would normally report. Worse, they could lie about conditions on a system causing the sentinel to waste resources attempting to fix fake problems. Subverted sensors could also dropping circular pheromone trails to lead others astray.

To mitigate subversion attacks Sensors might be encrypted in transit and contain a shared identifier string. A Sensor that decrypted without a proper identifier would be considered an intruder. However, mobile code is still vulnerable unless the hosts can be trusted. Protecting the Sentinel is the best way to avoid subversion of Sensors.

5.6 Other Attacks

There are many possible attack vectors, and certainly we have not enumerated them all. But the real problems in a system like CID arise from insufficient diversity of Sensors and discretization of inherently continuous ant colony algorithms. These are difficult to overcome when applying any solution from a continuous nature to a digital "world" created by man. Though we can approximate continuous spaces and infinite diversity to an arbitrary degree, we are limited by our finite ability to represent them in available space. These problems exists to some degree in any human-made system.

6. RELATED WORK

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 intrusion detection systems (IDS) may be merged with network management systems (NMS) using a hierarchy of multiple heterogeneous lightweight agents. Similarly, Śmieja[26] presents a heterogeneous hierarchy of agents based on neural nets and Selfridge's Pandemonium system[25]. Ibrahim[14] 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.

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.

7. FUTURE WORK

Each level of the CID hierarchy plays an important part in securing critical infrastructures. The requirements of the levels and how they interoperate were described, however future research is needed to determine how best to provide the necessary functionality. For example the field of biologically inspired autonomous agents is a rapidly evolving and there may be developments which could be brought into the design. This section will briefly discuss a few areas of future research.

Pheromone-based communication was introduced to manage Sensors within the framework, however microeconomics is another approach that has several benefits. In this case agents have a budget and seek to purchase resources to maximize their utility. Prices are set to encourage action towards a common goal and can adjust over time [4]. While perhaps slightly more complex than using pheromones, microeconomics offers various possible allocation goals and provable stability in dynamic environments [19].

Given the use of classifiers, research is needed to determine the classification methods best suited for each level within the framework. Sensors need classifiers that are efficient and robust, while Sentinels need more complex classifiers that can adapt to changing conditions. It may be advantageous encourage adaptive behavior when creating Sensors. Sensors could combine in some fashion (similar to genetic algorithms), yielding Sensors more adept to finding issues.

The interaction mechanisms between Supervisor and Sergeant also need to be explored further. We need to research the interface that allows the human to make sense of the information coming back from the Sergeant. Likewise, we wish to describe a language that will allow the human to provide guidance that humans can easily understand and Sergeants can unambiguously translate to executable policy.

Research must be done to determine efficient and secure means of transmitting mobile agents across the enclave and between enclaves. We have done some work that uses port knocking as part of a transport authentication protocol. The whole issue of lightweight authentication must be explored as part of the security of the overall framework.

8. CONCLUSION

Cyber security defenses face the dual threat of attackers who are both decentralized and constantly adapting to techniques used to thwart them. Current security system designs are often too centralized, giving them a single point of failure. Further, current designs require user actions at too low a level where human reaction time may be too slow to head off an attack.

CID keeps the human in charge of the hierarchy but does not require constant attention at the lower levels. The human Supervisor defines guidance for the high-level Sergeants, who then create policy for the lower-level agents to implement. Sentinels are responsible for managing host systems and interacting with Sensors. Sensors interaction with the Sentinel is modeled after ants foraging for food. By making the sensors as simple, diverse, and adaptive as possible, CID can adapt to attackers.

Even defense-in-depth architectures may be penetrated by attacking one layer at a time. Historically, what makes a network most secure is an astute systems/network administrator. Over time, the CID architecture will act as an amplifier of the important anomalous activity that will trigger the human to further investigate and expose any malicious activity.

CID avoids one of the major pitfalls of an IDS which is bombarding a human with false positives because the base rate of dangerous activity is actually quite low. Even a low Type-I rate (10E-05 for instance) can result in a huge number of false positive reports given a large amount of legitimate traffic [1]. In CID, a Sensor might report a false positive to the Sentinel and still get fed. Rewarding false positives is actually a desired behavior since the excited Sensor

will leave pheromone to draw other varieties of Sensors to the Sentinel. These Sensors will allow the Sentinel to gather enough evidence to make a clear decision on whether or not there is a problem. *The key advantage of CID over the state of the art in intrusion detection is that the interactions of software agents can spare the human supervisor the tedium of false positives yet keep his or her decision-making power robust.*

We believe CID's mixed-initiative, infrastructure-protection approach marks new territory in the security and agent-based programming research. The scientific contributions of this work include:

- *Creation of a mixed-initiative cyber defense solution* that engages humans without requiring them to directly control the defense.
- *A new defensive framework for infrastructures* to allow diverse organizations to cooperate in an adaptive cyber defense.
- *Unification of various types of rationality* including complex-adaptive swarms, rational agents, and humans.
- *A method that turns false positives into beneficial forms of positive feedback* to improve system performance without burdening human users.
- *A new geographic layout for computer networks* that will enable more direct adaptation of other nature-inspired algorithms and methods to cyber security.

We expect follow-on research will benefit from our efforts, and we look forward to joining the ongoing dialogue to mitigate the fundamental limitations of existing cyber security technology.

9. REFERENCES

[1] Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Trans. Inf. Syst. Secur.*, 3(3):186–205, 2000.

[2] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 164–173, 6-8 May 1996.

[3] E. Bonabeau, F. Henaux, S. Guérin, D. Snyers, P. Kuntz, and G. Theraulaz. Routing in telecommunications networks with "smart" ant-like agents. In *Intelligent Agents for Telecommunications Applications '98 (IATA'98)*, 1998.

[4] Jonathan Bredin, David Kotz, and Daniela Rus. Economic markets as a means of open mobile-agent systems. In *Proceedings of the Workshop Mobile Agents in the Context of Competition and Cooperation (MAC3)*, pages 43–49, May 1999.

[5] Sven Brueckner. *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. PhD thesis, Humboldt University Berlin, Department of Computer Science, 2000.

[6] Gianni Di Caro and Marco Dorigo. Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[7] Emad Abdel Rahim Dahiyat. Intelligent agents and intentionality: Should we begin to think outside the box? *Computer Law & Security Report*, 22(6):472–480, 2006.

[8] Ioanna Dionysiou. *Dynamic and Composable Trust for Indirect Interactions*. PhD thesis, Washington State University, 2006.

[9] Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the travelingsalesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[10] S. Farrell and R. Housley. An internet attribute certificate profile for authorization (rfc 3281). *Internet Engineering Task Force, Standards Track*, April 2002.

[11] Deborah Frincke, Andreas Wespi, and Diego Zamboni. Guest editorial: From intrusion detection to self-protection. *Comput. Netw.*, 51(5):1233–1238, 2007.

[12] T. Grandison and M. Sloman. Specifying and analysing trust for internet applications, 2002.

[13] Marti A. Hearst. Mixed-initiative interaction. *IEEE Intelligent Systems*, Sep/Oct 1999.

[14] Mohammed A. M. Ibrahim. Distributed network management with secured mobile agent support. *Hybrid Information Technology, 2006. ICHIT '06. Vol1. International Conference on*, 1:244–251, Nov. 2006.

[15] K. John, C. Dennis, H. Alexander, W. Antonio, C. Jonathan, H. Premkumar, and D. Michael. The willow architecture: Comprehensive survivability for large-scale distributed applications, 2002.

[16] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Computer*, pages 41–50, 2003.

[17] Nyree Lemmens, Steven de Jong, Karl Tulys, and Ann Nowe. Bee behavior in multi-agent systems: A bee foraging algorithm. *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*, 4865:145–156, 2006.

[18] Kwei-Jay Lin, Haiyin Lu, Tao Yu, and Chia en Tai. A reputation and trust management broker framework for web applications. *e-Technology, e-Commerce and e-Service, 2005. EEE '05. Proceedings. The 2005 IEEE International Conference on*, pages 262–269, 29 March-1 April 2005.

[19] W. Nicholson. *Microeconomic Theory Basic Principles and Extensions*. Dryden Press, 1989.

[20] H. Van Dyke Parunak, Paul Nielsen, Sven Brueckner, and Rafael Alonso. Hybrid multi-agent systems: Integrating swarming and bdi agents. In *Engineering Self-Organising Systems*, volume 4335/2007, pages 1–14. Springer, Berlin/Heidelberg, 2007.

[21] Xinzhou Qin, Wenke Lee, L. Lewis, and J.B.D. Cabrera. Integrating intrusion detection and network management. *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*, pages 329–344, 2002.

[22] Anand S. Rao and Michael P. Georgeff. Bdi agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, 1995.

[23] Michael B. Scher. On doing "being reasonable". *;login:*

The USENIX Magazine, 31(6):40–47, 2006.

[24] K.E. Seamons, T. Chan, E. Child, M. Halcrow, A. Hess, J. Holt, J. Jacobson, R. Jarvis, A. Patty, B. Smith, T. Sundelin, and Lina Yu. Trustbuilder: negotiating trust in dynamic coalitions. *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, 2:49–51 vol.2, 22-24 April 2003.

[25] O. G. Selfridge. Pandemonium: a paradigm for learning. In *Neurocomputing: foundations of research*, pages 115–122. MIT Press, Cambridge, MA, USA, 1988.

[26] F. Śmieja. The pandemonium system of reflective agents. *Neural Networks, IEEE Transactions on*, 7(1):97–106, Jan 1996.

[27] N. Stakhanova, S. Basu, J. Wong, and O. Stakhanov. Trust framework for p2p networks using peer-profile based anomaly technique. *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, pages 203–209, 6-10 June 2005.