Architecture for an Artificial Immune System

Steven A. Hofmeyr[1] and **S. Forrest**[1,2]

[1]Department of Computer Science, UNM, Albuquerque, NM 87131
[2]Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501

Abstract

An artificial immune system (ARTIS) is described which incorporates many properties of natural immune systems, including diversity, distributed computation, error tolerance, dynamic learning and adaptation and self-monitoring. ARTIS is a general framework for a distributed adaptive system and could, in principle, be applied to many domains. In this paper, ARTIS is applied to computer security, in the form of a network intrusion detection system called LISYS. LISYS is described and shown to be effective at detecting intrusions, while maintaining low false positive rates. Finally, similarities and differences between ARTIS and Holland's classifier systems are discussed.

1 INTRODUCTION

The biological immune system (IS) is highly complicated and appears to be precisely tuned to the problem of detecting and eliminating infections. We believe that the IS provides a compelling example of a massively-parallel adaptive information-processing system, one which we can study for the purpose of designing better artificial systems. The IS is compelling because it exhibits many properties that we would like to incorporate into artificial systems: It is diverse, distributed, error tolerant, dynamic, self-monitoring (or self-aware) and adaptable. These properties give the IS certain key characteristics that most artificial systems today lack: robustness, adaptivity and autonomy.

Robustness is a consequence of the fact that the IS is diverse, distributed, dynamic and error tolerant. *Diversity* improves robustness on both a population and individual level, for example, different people are vulnerable to different infections. The IS is *distributed* in a robust fashion: Its many components interact locally to provide global protection, so there is no central control and hence no single point of failure. The IS is *dynamic* in that individual components are continually created, destroyed, and circulated throughout the body, which increases the temporal and spatial diversity of the IS. Finally, the IS is robust to errors (*error tolerant*) because the effect of any single IS action is small, so a few mistakes in classification and response are not catastrophic.

The IS is *adaptable* in that it can learn to recognize and respond to new infections and retain a memory of those infections to facilitate future responses. This adaptivity is made possible by the *dynamic* functioning of the IS, which enables the IS to discard components which are useless or dangerous and to improve on existing components.

The IS is *autonomous* in that there is no outside control required, and the IS is an integrated part of the body, and hence the same mechanisms that monitor and protect the rest of the body also monitor and protect the IS. Furthermore, the distributed, decentralized nature of the IS contributes to its autonomous nature: Not only is there no outside control, but there is no way of imposing outside control or even inside, centralized control.

These properties of robustness, adaptability, turnover of components, and autonomy are closely related to the design principles of complex adaptive systems articulated by Holland in, for example, [22, 23]. Furthermore, the immune system appears to reflect many aspects of a less well-articulated design aesthetic illustrated by Holland's genetic algorithms, classifier systems, and Echo. Common features in those systems include: fine-grained representations and actions, emergence of coordinated behavior, competition among components, random variation, evolution, and close coupling with a perpetually novel environment.

Representations and actions in the immune system are fine-grained (short protein fragments, called peptides, are the basic unit of representation). Coherent coordinated behavior arises (or *emerges*) from the interactions of literally trillions of cells and molecules. Each individual action of the immune system (forming a chemical bond, secreting molecules from a cell, killing a single cell, etc.) is also fine-grained. Another feature of the Holland design aesthetic is the notion of competition for survival among the basic units of an adaptive system. This is seen in the immune system when individual immune cells compete with one another to bind foreign antigen. Immune receptors are created randomly through genetic recombinations and mutations. Mutations take place when gene fragments are joined into a single gene (junctional diversity) and during affinity maturation (somatic hypermutation). Evolutionary processes play a central role in the Holland aesthetic. The immune system illustrates the use of evolution as an engine of innovation in its affinity maturation of B-cells in response to foreign antigen, which quite closely resembles a genetic algorithm without crossover. Finally, the notion of an adaptive system being closely coupled with its environment, responding to perpetually novel stimuli in a dynamic and flexible way, is a basic tenet of Holland's view of adaptive systems. This view is perhaps better illustrated by classifier systems and Echo than by conventional genetic algorithms.

We describe a system called (ARTIS[1]) which incorporates these properties. To preserve generality, ARTIS is described independently of any particular problem domain. However, to ground these concepts, we *situate* ARTIS in a networked environment as a computer security system called LISYS[2]. We follow Brooks [3] and others [17, 5] in believing that it is fruitless to design intelligent systems in complete isolation from the environments in which they exist. The hope is that situating an intelligent artifact will simplify it, because it can use its environment to reduce computations, and it will be less likely to include unnecessary features or mechanisms.

Computer security is an important and natural application domain for adaptive systems. Computer systems are dynamic, with continually changing patterns of behavior; programs are added and removed, new users are introduced, configurations change. These and other changes allow intruders to find novel means to gain improper access to computers. Traditional computer security mechanisms are largely static and so cannot easily cope with dynamic environments. We believe that an adaptive system is needed to track both changes in the environment and the way in which intruders exploit systems. A computer security system should protect a machine or set of machines from intruders and

[1] **ART**ificial **I**mmune **S**ystem.
[2] This stands for "**L**ightweight **I**ntrusion detection **SYS**tem. The word LISYS is a corruption of "lysis", which is the process whereby the immune system destroys bacteria by rupturing the bacterial membrane.

foreign code, which is similar in functionality to the immune system protecting the body from invasion by inimical microbes. Because of these similarities, we have designed and implemented LISYS, an intrusion detection system that monitors network traffic. LISYS demonstrates the utility of ARTIS when applied to a specific problem domain.

In earlier papers we presented our results in the context of computer security [11, 9, 8, 13, 12, 19], deemphasizing more general considerations. The goal of this paper is to rectify that, making the biological connections more concrete and emphasizing the adaptive systems framework in which our implementation resides. In the next section (2) we briefly introduce the immune system, and in the following section (3) we describe the organization of ARTIS, further explaining immunological concepts where necessary. The results of testing the system out in a real environment are described in section 4. The following section (5) discusses the relation of ARTIS to classifier systems [24], and the section after that (6) briefly describes other domains to which ARTIS might be applied. Finally, the paper is concluded with general comments concerning computer security and adaptive systems.

2 THE IMMUNE SYSTEM

The IS consists of a multitude of cells and molecules which interact in a variety of ways to detect and eliminate infectious agents (pathogens). These interactions are localized because they depend upon chemical bonding—surfaces of immune system cells are covered with receptors, some of which chemically bind to pathogens, and some of which bind to other immune system cells or molecules to enable the complex system of signalling that mediates the immune response. Most IS cells circulate around the body via the blood and lymph systems, forming a dynamic system of distributed detection and response, where there is no centralized control, and little, if any, hierarchical organization. Detection and elimination of pathogens is a consequence of trillions of cells interacting through simple, localized rules. A consequence of this is that the IS is very robust to failure of individual components and attacks on the IS itself.

The problem of detecting pathogens is often described as that of distinguishing "self" from "nonself" (which are elements of the body, and pathogens, respectively). However, many pathogens are not harmful, and an immune response to eliminate them may damage the body. In these cases it would be healthier not to respond, so it would be more accurate to say that the problem faced by the IS is that of distinguishing between *harmful* nonself and everything else [28, 29]. We adopt the viewpoint that "nonself" is synonymous with any pathogen that is harmful to the body, and "self" is synonymous with harmless substances, including all normally functioning cells of the body.

Once pathogens have been detected, the IS must eliminate them in some manner. Different pathogens have to be eliminated in different ways, and we call the cells of the IS that accomplish this *effectors*. The elimination problem facing the immune system is that of choosing the right effectors for the particular kind of pathogen to be eliminated. We shall touch only briefly on the problem of elimination, or response, in section 3.9; the main focus of this paper is on the detection problem.

3 ARCHITECTURE OF AN ARTIFICIAL IMMUNE SYSTEM

In this section we describe the architecture of our artificial immune system (ARTIS). ARTIS is closely modeled on the biological immune system, and so in the course of describing ARTIS, we shall describe the equivalent biological mechanisms that inspired the model. Of necessity, the immunological details will be sparse and fragmentary; for a detailed overview of immunology that is still accessible to non-immunologists, consult [20][3].

3.1 DEFINING THE PROBLEM

All discrimination between self and nonself in the IS is based upon chemical bonds that form between protein chains. To preserve generality, we model protein chains as binary strings of fixed length ℓ. The IS must distinguish self from nonself based on proteins; ARTIS addresses a similar problem, which we define as follows. The set of all strings of length ℓ forms a universe, U, which is partitioned into two disjoint subsets, which we call self, S, and nonself, N (i.e. $U = S \cup N$, $S \cap N = \emptyset$). ARTIS faces a discrimination or classification task: Given an arbitrary string from U, classify it as either normal (corresponding to self) or anomalous (corresponding to nonself)[4].

ARTIS can make two kinds of discrimination errors: A false positive occurs when a self string is classified as anomalous, and a false negative occurs when a nonself string is classified as normal. The IS also makes similar errors: A false negative occurs when the IS fails to detect and fight off pathogens, and a false positive error occurs when the IS attacks the body (known as an autoimmune response). In the body, both kinds of errors are harmful, so the IS has apparently evolved to minimize those errors; similarly, the goal of ARTIS is to minimize both kinds of errors. See figure 1.

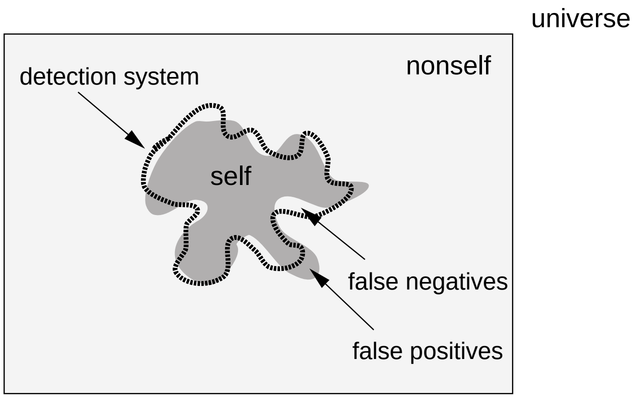

Figure 1: A two-dimensional representation of a universe of strings. Each string can belong to one of two sets: self or nonself. In this diagram, each point in the plane represents a string; if the point lies within the shaded area it is self, otherwise it is nonself. The immunological detection system attempts to encode the boundary between the two sets by classifying strings as either normal (corresponding to self) or anomalous (corresponding to nonself).

When real-world problems are mapped to this abstraction, self and nonself may not be disjoint, because some strings may characterize both self and nonself. In this case, the categorization of strings as either one or the other category will lead to unavoidable errors. We do not consider that case here. However, it illustrates the importance of choosing the right characteristic for the application domain: It is essential to choose the equivalent of proteins that can be used to reliably discriminate between self and nonself.

[3]For truly comprehensive references, the interested reader should consult [35, 25, 33].

[4]This definition can be generalized to include classification in multiple categories, not only the two categories of self and nonself.

3.2 DETECTORS

Natural immune systems consist of many different kinds of cells and molecules which have been identified and studied experimentally. In our system, we will simplify by introducing one basic type of detector which is modeled on the class of immune cells called *lymphocytes*[5]. This detector combines properties of B-cells, T-cells, and antibodies. ARTIS is similar to the IS in that it consists of a multitude of mobile detectors, circulating around a distributed environment. We model the distributed environment with a graph $G = (V, E)$; each vertex $v \in V$ contains a local set of detectors (called a *detection node*) and detectors migrate from one vertex to the next via the edges. The graph model also provides a notion of locality: Detectors can only interact with other detectors at the same vertex. This notion of locality is useful, as we shall see in section 3.5.

Lymphocytes have hundreds of thousands of identical receptors on their surface (and hence are termed *monoclonal*). These receptors bind to regions (epitopes) on pathogens. Binding depends on chemical structure and charge, so receptors are likely to bind to a few similar kinds of epitopes. The greater the likelihood of a bond occurring, the higher the *affinity* between the receptor and epitope. In ARTIS, both epitopes and receptors are modeled as binary strings of fixed length ℓ, and chemical binding between them is modeled as approximate string matching. In effect, each detector is associated with a binary string, which represents its receptors.

Obvious approximate matching rules include Hamming distance and edit distance, but we have adopted a more immunologically plausible rule, called *r-contiguous bits* [34]: Two strings match if they have r contiguous bits in common (see figure 2). The value r is a threshold and determines the specificity of the detector, which is an indication of the size of the subset of strings that a single detector can match. For example, if $r = \ell$, the matching is completely specific, that is, the detector will match only a single string (itself), but if $r = 0$, the matching is completely general, that is, the detector will match every single string of length ℓ.

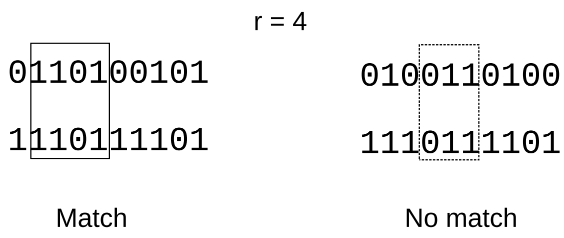

Figure 2: Matching under the contiguous bits match rule. In this example, the detector matches for $r = 3$, but not for $r = 4$.

A consequence of a partial matching rule with a threshold, such as r-contiguous bits, is that there is a trade-off between the number of detectors used, and their specificity—as the specificity of the detectors increases, so the number of detectors required to achieve a certain level of detection also increases. The optimal r is one which minimizes the number of detectors needed, but still gives good discrimination.

A lymphocyte becomes *activated* when its receptors bind to epitopes. Activation changes the state of the lymphocyte and triggers a series of reactions that can lead to

[5]The lifecycle of a detector is shown in section 3.7, figure 4. It may help understanding to refer forwards to this figure.

elimination of the pathogens (this is discussed in section 3.9). A lymphocyte will only be activated when the number of its receptors binding to epitopes exceeds a threshold[6]. Chemical bonds between receptors and epitopes are not long-lasting, so to be activated, a lymphocyte must bind sufficient receptors within a short period of time. We model this with *activation thresholds*: A detector must match at least τ strings within a given time period to be activated. This is implemented by allowing the detector to accumulate matches, but decaying the match count over time, i.e. there is a γ_{match} probability that the match count will be reduced by one at any timestep. This models the probability of a bond between a receptor and an epitope decaying. Once a detector has been activated, its match count is reset to zero.

3.3 TRAINING THE DETECTION SYSTEM

Lymphocytes are called *negative* detectors because they are trained to bind to nonself; i.e. when a lymphocyte is activated, the IS responds as if nonself were detected. This simple form of learning is known as *tolerization*, because the lymphocytes are trained to be *tolerant* of self.

Lymphocytes are created with randomly generated receptors, and so could bind to either self or nonself. One class of lymphocytes, T-cells, is tolerized in a single location, the thymus, which is an organ just behind the breastbone. Immature T-cells develop in the thymus, and if they are activated during development, they die through programmed cell death (*apoptosis*). Most self proteins are expressed in the thymus, so T-cells that survive to maturation and leave the thymus will be tolerant of all those self proteins. This process is called *negative selection*, because the T-cells that are not activated are the ones selected to survive.

Lymphocytes are trained to perform *anomaly* detection. The IS uses a *training* set of self (proteins present in the thymus) to produce detectors that can distinguish between self and nonself. This clearly will not work if nonself is frequently expressed in the thymus, because then the IS will also be tolerant to that nonself. The underlying assumption is that self occurs frequently compared to nonself. This assumption is the basis of most anomaly detection systems, which define normal as the most frequently occurring patterns or behaviors.

ARTIS uses the *negative selection algorithm*, which is based on negative selection in the IS (see figure 3) [13]. The primary difference is that we do not accumulate the self set in a single location, but rather use a form of asynchronous and distributed tolerization[7]. Each detector is created with a randomly-generated bit string (analogous to a receptor), and remains immature for a time period T, called the tolerization period. During this time period, the detector is exposed to the environment (self and possibly nonself strings), and if it matches any bit string it is eliminated. If it does not match during the tolerization period, it becomes a mature detector (analogous to a naive B-cell). Mature detectors need to exceed the match threshold in order to become activated, and when activated they are not eliminated[8], but signal that an anomaly has been detected. Clearly, the assumption here is that if a circulating immature detector matches some self string, it will, with high probability, encounter that self string during its tolerization period, whereas immature detectors that match nonself strings will with low probability encounter those nonself strings during their tolerization period.

---

[6] In the case of T-cells, it suffices for several hundred receptors to bind.
[7] Hence, we use a distributed form of the negative selection algorithm.
[8] They could still be eliminated through lack of costimulation; see section 3.6.

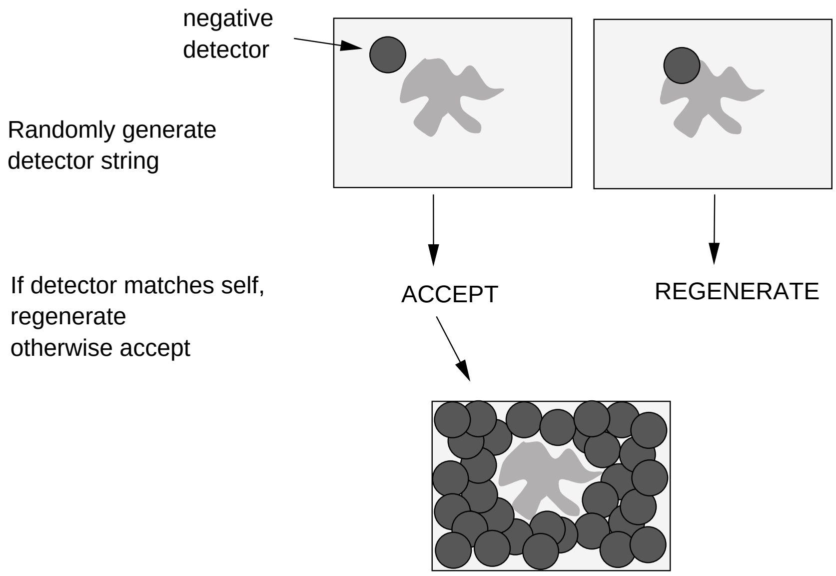

Figure 3: The negative selection algorithm. Candidate negative detectors (represented by dark circles) are generated randomly, and if they match any string in the self set (i.e. if any of the points covered by the detector are in the self set), they are eliminated and regenerated. This process is repeated until we have a set of valid negative detectors that do not match any self strings.

3.4 MEMORY

The IS has an adaptive response that enables it to learn protein structures that characterize pathogens it encounters, and "remember" those structures so that future responses to the same pathogens will be very rapid and efficient. We call this memory-based detection, because the IS "remembers" the structures of known pathogens to facilitate future responses. A memory-based detection system is trained on a subset of nonself to detect particular elements of that subset. When the IS encounters pathogens of a type it has not encountered before, it mounts a *primary response*, which may take several weeks to eliminate the infection; during the primary response the IS is learning to recognize previously unseen foreign patterns. When the IS subsequently encounters the same type of pathogens, it mounts a *secondary response* which is usually so efficient that there are no clinical indications of a re-infection. The secondary response illustrates the efficacy of memory-based detection.

Memory-based detection in the IS has another important property: It is *associative* [38]. Memory detection allows the IS to detect new pathogens that are structurally related to ones previously encountered. This concept underlies immunization, where inoculation with a harmless form of pathogen, A (such as an attenuated virus) induces a primary response that generates a population of memory cells which are cross-reactive with a harmful kind of pathogen, B. This population of memory cells will ensure that the IS mounts a secondary response to any infections of B.

Primary responses are slow because there may be very few lymphocytes that bind to a new type of pathogen, so the immune response will not be very efficient. To increase efficiency, activated lymphocytes clone themselves, so that there is an exponentially growing population of lymphocytes which can detect the pathogens. The higher the affinity between a lymphocyte's receptors and the pathogen epitopes, the more likely it is that the lymphocyte will be activated. Hence, the lymphocytes that are replicating are those with the highest affinity for the pathogens present. During this time the pathogens are also replicating, so there is a race between pathogen replication and lymphocyte replication. The IS improves its chances in this race through a class of lymphocytes called B-cells, which are subject to high mutation rates (known as somatic hypermutation) when cloning (we currently do not model this aspect). Hypermutation combined with clonal expansion is an adaptive process known as *affinity maturation*. Once the infection is eliminated, the IS retains a population of *memory* cells: long-lived lymphocytes which have a high affinity for the pathogen. This population of memory cells is of sufficient size and specificity to enable the very rapid secondary response when a re-infection occurs.

ARTIS uses a similar form of memory-based detection. When multiple detectors at a node are activated by the same nonself string, s, they enter a competition to become memory detectors. Those detectors that have the closest match (under r-contiguous bits) with s will be selected to become memory detectors[9]. These memory detectors make copies of themselves, which then spread out to neighboring nodes. Consequently, a representation of the string s is distributed throughout the graph; future occurrences of s will be detected very rapidly in any node because detectors that match s exist at every node. In addition, memory detectors have lowered activation thresholds (for example, $\tau = 1$), so that they will be activated far more rapidly in future to re-occurrences of previously encountered nonself strings, i.e. they are much more sensitive to those strings. This mimics the rapid second response seen in the IS.

3.5 SENSITIVITY

A detection event in the IS often results in the production of chemicals (cytokines) which signal other nearby IS cells. To model this, we use the notion of locality inherent in the graph defining the environment for ARTIS. Each detection node D_i (where $i = 1, 2, \ldots, |V|$) has a local sensitivity level, ω_i, which models the concentration of cytokines present in a physically local region in the body. The activation threshold of detectors at D_i is defined as $\tau - \omega_i$, i.e. the higher the local sensitivity, the lower the local activation threshold. Whenever the match count for a mature detector at node i goes from 0 to 1, the sensitivity level at D_i is increased by 1. The sensitivity level also has a temporal horizon: over time it decays at a rate given by a decay parameter γ_ω, which indicates the probability of ω_i being reduced by 1. This mechanism ensures that disparate nonself strings will still be detected, providing they occur in a short period of time.

3.6 COSTIMULATION

Unfortunately, tolerization in the IS is not as straightforward as described in section 3.3. Some self proteins are never expressed in the thymus, and so lymphocytes that are tolerized centrally in the thymus may bind to these proteins and precipitate an autoimmune reaction. This does not happen in practice because T-cells require *costimulation* to be activated: In addition to binding to proteins (called *signal one*), a T-cell must be costimulated by a *second signal*. This second signal is usually a chemical signal which occurs when

[9]Because matching is approximate, and the activation threshold, τ, can be greater than one, it is possible that the final string s_τ that activates a detector differs somewhat from the strings $s_1, \ldots, s_{\tau-1}$ that were previously matched and hence contributed to activation. We deal with this problem in the simplest possible way, by assuming that the final string s_τ is suitably representative of all the strings that contributed towards activation.

the body is damaged in some way. The second signal can come either from cells of the IS or other cells of the body. When a T-cell receives signal one in the absence of signal two, it dies. Hence, autoreactive T-cells (those that bind to self) will be eliminated in healthy tissues. However, if the tissues are damaged, autoreactive T-cells could survive. But they would only survive while the damage persisted; as soon as they left the area of damage, they would receive signal one in the absence of signal two and die. Moreover, they would have a high likelihood of dying before they ever reached the area of tissue damage, because of the healthy tissue passed through on the way.

Likewise, we cannot assume that in ARTIS a detector will encounter every string $s \in S$ during its tolerization period, so it is possible that detectors will mature that match some strings in S. We implement a crude form of costimulation. Ideally, the second signal should be provided by other components of the system, but our first approximation is to use a human operator to provide the second signal. When a detector d is activated by a string s, it sends a signal to a human operator, who is given a time period T_s (called the costimulation delay) in which to decide if s is really nonself. If the operator decides that s is indeed nonself, a second signal is returned to d. If the operator decides that s is actually self, no signal is sent to d and d dies off and is replaced by a new, immature detector. Consequently, a human operator need make no response in the case of false positives; the system will automatically correct itself to prevent similar false positives in future.

3.7 THE LIFECYCLE OF A DETECTOR

If detectors lived indefinitely and only died off when they failed to receive costimulation, most detectors would only be immature once. Any nonself strings that occurred during the period of immaturity of these detectors would not be detected in future because all detectors would be tolerant of them and would remain tolerant. In the IS this is not a problem because lymphocytes are typically short-lived (a few days) and so new, immature lymphocytes are always present, i.e. the population of lymphocytes is dynamic. We introduce a similar measure: Each detector has a probability p_{death} of dying once it has matured. When it dies, it is replaced by a new randomly-generated, immature detector. Ultimately, every detector dies sooner or later, unless it is a memory detector. Figure 4 presents the lifecycle of a detector. Now a nonself string will only be undetected if it is continually present to tolerize the continual turnover of new detectors, i.e. the only false negatives will occur when nonself is frequent, which violates a fundamental assumption underlying our model.

An exception to the finite lifespan is memory detectors. In the IS, memory cells are long-lived so that the patterns that they encode are not lost over time. For example, exposure to measles early in life confers life-long protection against the disease. Similarly, memory detectors in ARTIS are long-lived: They can die only as a consequence of a lack of costimulation. A problem with this mechanism is that eventually all detectors could become memory detectors, with a loss of the advantages conferred by dynamic detector populations. To combat this problem, we limit the number of memory detectors to some fraction m_d of the total detectors. If a new detector wins a competition and becomes a memory detector, and the fraction of memory detectors has reached the limit, then the least-recently-used (LRU) memory detector is demoted to an ordinary mature detector (consequently it once more has a finite lifespan). We demote the LRU detector because the LRU detector is the one that has not been activated for the longest time period of any memory detector, and hence we assume that it is the least useful memory detector.

An additional benefit of a dynamic detector population is that the system can adapt

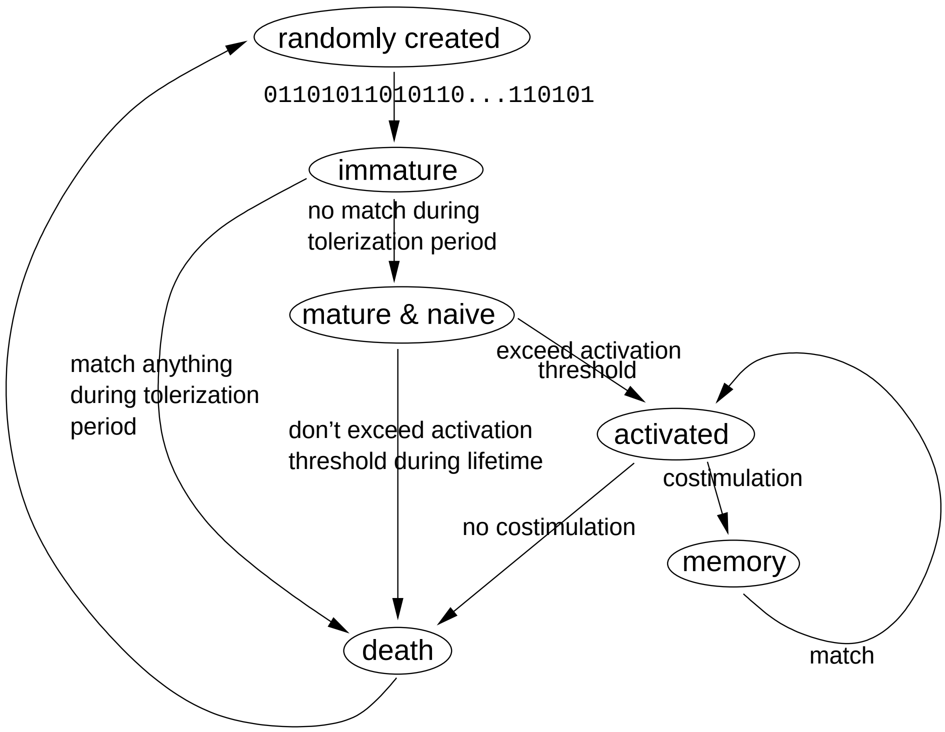

Figure 4: The lifecycle of a detector. A detector consists of a randomly created bit string that is immature for the tolerization period T. If it matches anything once during that period it dies and is replaced by a new randomly generated detector. If it survives tolerization, it becomes a mature, naive detector that lives for an expected $1/p_{death}$ time-steps. If the detector accumulates enough matches to exceed the activation threshold τ, it is activated. If the activated detector does not receive costimulation, it dies. If it receives costimulation, it enters a competition to become a memory detector. Once a memory detector, it lives indefinitely, and only requires a single match for activation.

to changing self sets. As the self set changes, it will tolerize new immature detectors, and mature detectors that were causing false positives will either die from lack of costimulation or from age. Eventually all detectors will be tolerant of self, providing self does not change too quickly. If self changes rapidly compared to the life span of a detector, there will be a sizeable portion of detectors that are immature, because mature detectors will continually die due to lack of costimulation.

3.8 REPRESENTATIONS

Molecules of the *major histocompatibility complex* (MHC) play an important role in the IS, because they transport peptides (fragments of protein chains) from the interior regions of a cell and *present* these peptides on the cell's surface. This mechanism enables roving IS cells to detect infections inside cells without penetrating the cell membrane. There are many variations of MHC, each of which binds a slightly different class of peptides. Each individual in a population is genetically capable of making a small set of these MHC types (about ten), but the set of MHC types varies in different individuals. Consequently, individuals in a population are capable of recognizing different profiles of peptides, providing an important form of population-level *diversity*[10].

We speculate that MHC plays a crucial role in protecting a population of individuals from *holes* in the detection coverage of nonself. A hole is a nonself string for which no valid detectors can be generated [8]: A nonself string $a \in N$ is a hole if and only if, $\forall u \in U$, such that u and a match, then u matches some self string, $s \in S$. See figure 5. Holes can exist for any approximate match rule with a constant probability of matching (such as the r-contiguous bits) [8], and it is reasonable to assume that they will exist in the biological realm of receptor binding, because binding between receptors in the IS and peptides is approximate. Moreover, pathogens will always be evolving so that they are more difficult to detect (they evolve towards becoming holes in the detection coverage). Those pathogens that are harder to detect will be the ones that survive better and hence are naturally selected.

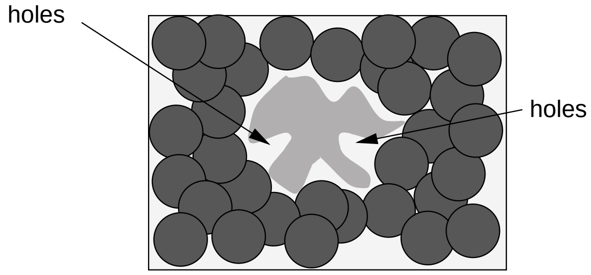

Figure 5: The existence of holes. There are strings in the nonself set that cannot be covered by valid negative detectors of a given specificity (match length r). The size of the dark circles representing detectors is an indication of the generality of those detectors. The detectors depicted in the example here are too general to match certain nonself strings without also matching self.

In the IS, each type of MHC can be regarded as a different way of representing a

[10]For example, there are some viruses, such as the Epstein-Barr virus, that have evolved dominant peptides which cannot be bound by particular MHC types, leaving individuals who have those MHC types vulnerable to the disease [25].

protein (depending on which peptides it presents); in effect, the IS uses multiple representations, or views, of proteins. Multiple representations can reduce the overall number of holes, because different representations will induce different holes. In ARTIS, each detection node uses a different representation: It filters incoming strings through a randomly-generated permutation mask[11]. For example, given the strings $s_1 = 01101011, s_2 = 00010011$, and a permutation, Λ, defined by the randomly-generated permutation mask 1-6-2-5-8-3-7-4, these strings become $\Lambda(s_1) = 00111110$, and $\Lambda(s_2) = 00001011$. Using the contiguous bits rule with $r = 3$, s_1 matches s_2, because the last 3 positions are the same, but under the new representation, $\Lambda(s_1)$ does not match $\Lambda(s_2)$. Having a different representation for each detection node is equivalent to changing the "shape" of the detectors, while keeping the "shape" of the self set constant (see figure 6). Consequently, where one node fails to detect a nonself string, another node could succeed.

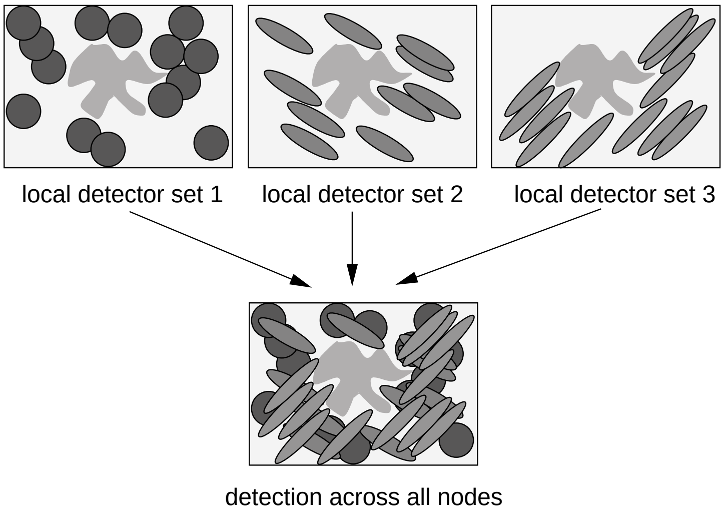

Figure 6: Representation changes are equivalent to "shape" changes for detectors. The problem of holes can be ameliorated by using different a representation for each detection node. There are different holes for different representations. or equivalently different shaped detectors can cover different parts of the nonself space, for a global reduction in holes.

3.9 RESPONSE

The immune system has a variety of *effector functions* because different pathogens must be eliminated in different ways. For example, intra-cellular pathogens such as viruses are eliminated by killer T-cells, whereas extracellular bacteria are eliminated by macrophages, complement, and so forth. After pathogens have been detected, the immune system must select the appropriate effectors so that the pathogens are efficiently eliminated. Selection of effectors is determined at least in part by chemical signals in the form of cytokines, but it is not clear how selection actually works. Mathematical models indicate ways in which selection could occur, if cytokines reflect the local state of the system (i.e. the damage suffered from pathogens, the damage suffered from the immune system, etc.) [37].

[11] We have also used other methods of changing the representation. This is the simplest. See [20] for details.

We focus on a particular form of effector selection, instantiated by a class of lymphocytes called B-cells. When B-cells are activated, they can differentiate to become *plasma* cells that secrete a soluble form of their receptors, called *antibodies*. Antibodies have a y-shaped structure, with three different regions, corresponding to the two arms and the tail of the y. The arms of the y are termed the *variable* regions, and the tail of the y is the *constant* region. The variable regions are randomly generated (as described in section 3.3) so that they bind to specific pathogen epitopes. The constant region, on the other hand, is not randomly generated (hence the name), but comes in a few structural varieties, called *isotypes*[12]. The constant region is the part of the antibody to which other immune system cells (such as macrophages) bind. Depending on the isotype of the constant region, different responses will be triggered upon binding, so it is this part of the antibody that determines effector function. A single B-cell can clone multiple B-cells, each with a different isotype, even while the receptor variable regions remain the same. This is known as *isotype switching*, and enables the immune system to choose between various effector functions via chemical binding.

To date we have focused on detection, and have not investigated the IS response in detail. Although the appropriate type of response will depend on the application domain (some application domains will require no forms of automated response), we can in abstract model effector selection, assuming that several different kinds of response are required, each associated with a different kind of effector [13]. Each detector can be augmented with the equivalent of a fixed region, which is a bit string that encodes effector choice. When detectors are copied after activation, they undergo a process similar to isotype switching, in which the fixed region bit string is set according to the type of response deemed appropriate. Migrating detectors will then not only carry information concerning the patterns of anomalies, but also specific information concerning how those anomalies should be eliminated. We expect that this will maintain a similar level of robustness and flexibility as we have achieved for the detection component of our system.

3.10 SUMMARY

Table 1 summarizes the differences and similarities between the IS and ARTIS. In ARTIS, we do not use central tolerization and so have no equivalent of the thymus (the elements labelled "NONE"). However, ARTIS could easily be implemented with negative detection occurring in a single location. We have avoided this because it reduces the robustness of the system, introducing a single point of failure.

4 APPLICATION DOMAIN: NETWORK SECURITY

In this section we take the abstract framework for ARTIS and demonstrate its utility by applying it to a particular domain, that of network intrusion detection (NID). We have previously applied the analogy to several aspects of computer security, including computer virus detection [13], host-based intrusion detection [11], and using diversity to make computers robust to wide-spread attacks [14]. In this paper we concentrate on a system (LISYS) that is designed to protect a local area network (LAN) from network-based attacks.

NID systems are designed to detect network attacks by analyzing and monitoring

[12] There are many different isotypes, for example, IgA, IgG, IgM.
[13] We discuss some possibilities for a specific application in section 4.4.

Immune System	ARTIS
peptide/protein/epitope	binary string
receptor	binary string
monoclonal lymphocyte (B-cell, T-cell)	detector
antibody variable region	detector string
antibody fixed region	bit string encoding response
memory cell	memory detector
pathogen	nonself binary string
binding	approximate string matching
locality	vertices in a graph
circulation	mobile detectors
central tolerization	NONE
thymus	NONE
MHC	representation parameters
cytokines	sensitivity level
peripheral tolerization	distributed negative selection
signal one	matches exceeds activation threshold
signal two (costimulation)	human operator
lymphocyte cloning	detector replication
pathogen detection	detection event
pathogen elimination	response
affinity maturation	memory detector competition

Table 1: Comparison between the IS and ARTIS. "NONE" indicates that there is no analogous mechanism.

network traffic[14]. Some good overviews of NID systems are given in [31]. Most NIDs perform signature-based detection which is similar, but more limited than memory-based detection. Unlike the IS, these NIDs have no facilities for automatic signature extraction; a human operator has to extract the signatures of attacks. Attack signatures are encoded in expert systems [18, 36, 40, 16], or state transition diagrams [43], or graphs [42]. Few NIDs perform anomaly detection; those that do [15, 36, 2] use some form of statistical analysis.

NIDs are usually distributed in the sense that they have multiple monitors across a network. Data from the monitors is collated in one of two ways: either centrally [18, 40, 43, 32] or hierarchically [36, 6, 42]. Although these multi-monitor systems are distributed in the sense that the monitors they employ are distributed throughout the network, they are not localized in the sense that the IS is localized: They have single failure points which are vulnerable to attack. Moreover, all the monitors are identical, and so a vulnerability in one monitor will be replicated across all of them; i.e. these NIDs are not diverse in the sense that the IS is. These disadvantages are overcome in LISYS.

4.1 MAPPING ARTIS TO NETWORK SECURITY

The first and in a sense most important step, when applying ARTIS to a domain, is to choose the equivalent of proteins. For example, in computer security, there are many different levels at which we can monitor performance and behavior; characteristics at each of these levels represent potential proteins. We have chosen to monitor network traffic, and so our protein is a "datapath triple", which consists of a source internet protocol (ip) address, a destination ip address, and a TCP service (or port) by which two computers communicate. Essentially, a protein is a connection between two computers. Currently,

[14]Systems that monitor network traffic are commonly labeled "network-based" [31].

we monitor only the start of TCP connections, i.e. we monitor TCP SYN packets. We have chosen this level (network connections) because the environment is naturally distributed (there are many computers communicating), and because other researchers have successfully implemented anomaly detection by monitoring network connections [15].

In our representation, the connection information is compressed to a single 49-bit string which unambiguously defines the connection. Self is then the set of "normally occurring" connections observed over time on the LAN. Thus, self is defined in terms of frequencies: We assume implicitly that any connection that occurs frequently over a long period of time is part of the self set. A connection can occur between two *internal* computers on the LAN, or between an internal computer on the LAN and an *external* computer outside the LAN. Each connection is represented by a 49-bit string (whether internal or external). Similarly, nonself is also a set of connections (using the same 49-bit representation), the difference being that nonself consists of those connections, potentially an enormous number, that are not normally observed on the LAN.

In ARTIS, we modeled the environment with a graph, where each vertex defined a locality corresponding to a detection node. For the NID application, each vertex corresponds to a computer within the LAN (an internal computer), and the network represents a fully connected graph (see figure 7), because we have assumed the network is broadcast. Broadcast LANs have the convenient property that every location (computer) sees every packet passing through the LAN.

In summary, LISYS is an implementation of ARTIS, as described in section 3. The binary strings are mapped from datapath triples and the environment is a network of computers, where each internal computer corresponds to a vertex in the graph, running a detection node. We have not yet incorporated mobile detectors, replication or response. We discuss these extensions in section 4.4.

4.2 EXPERIMENTS

LISYS was implemented and tested out using real data collected on a subnet of 50 computers at the Computer Science Department of the University of New Mexico. Two data traces were collected: self, S_t, consisting of normal traffic, and nonself, N_t, consisting of traffic generated during intrusive activity.

The self trace, S_t was collected over 50 days, during which a total of 2.3 million TCP connections were logged, each of which is a datapath triple. These 2.3 million datapaths were filtered down to 1.5 million datapaths. The filtering removed several classes of noisy traffic sources, such as web servers and ftp servers, because these are continually communicating with new hosts, and so have no stable definition of normal in terms of datapaths. Although we filtered out web and ftp server traffic, we did not filter out connections to web and ftp server ports on internal computers that should not have been running such servers. Moreover, for web servers and ftp servers, we only filtered out the traffic destined for the web and ftp server ports, but still continued to monitor other ports on the server computers. After this, a self set S, consisting of 3900 unique strings, was extracted from S_t.

The nonself trace, N_t was comprised of seven different intrusive incidents, that were faithful logs of real incidents that occurred on the network being studied. Most of these attacks consisted of probing of one sort or another, particularly of services with recently reported vulnerabilities. At least one incident involved compromise of an internal computer. The traffic tested for each incident consisted of all datapaths from the first nonself

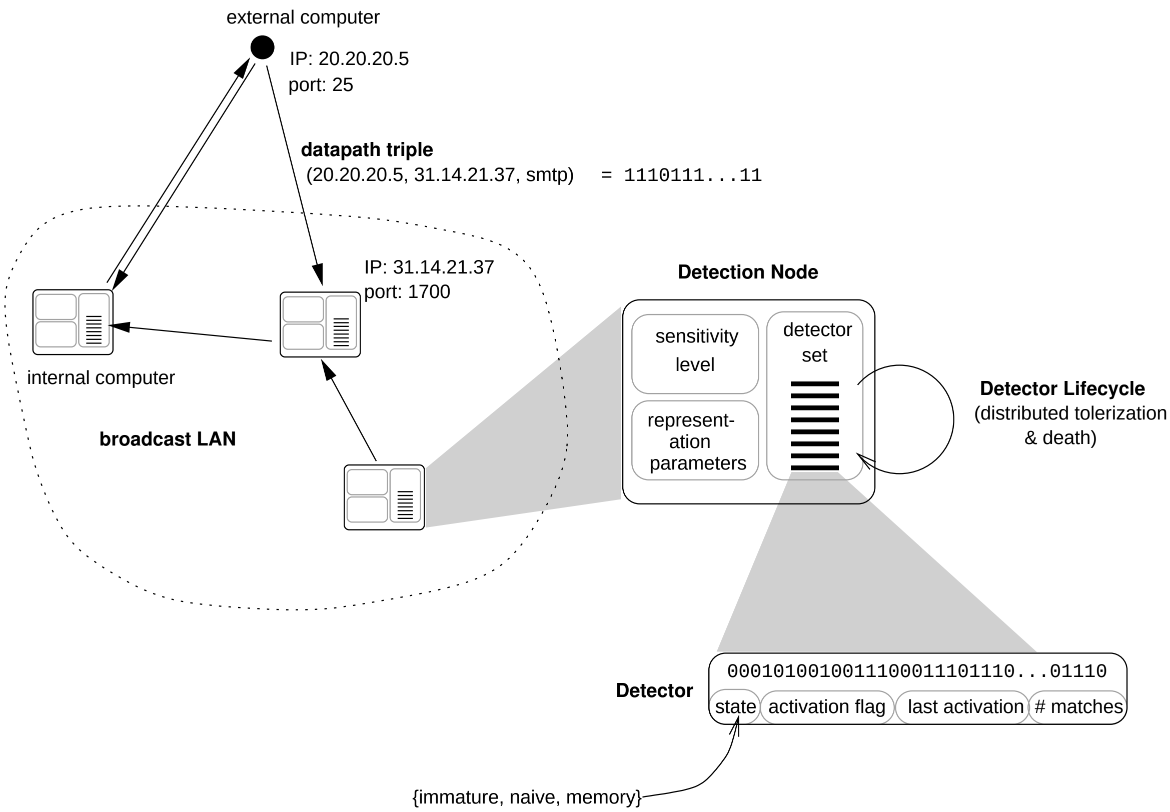

Figure 7: Architecture of LISYS. Each internal computer runs a detection node, which consists of a set of detectors, a local sensitivity level and a representation function with randomly generated parameters (for example, the permutation mask). Each detector contains a binary string, and is either immature, naive (and mature), or a memory detector. In addition, each detector keeps track of whether it is awaiting costimulation (using the activation flag) and how many matches it has accumulated.

Test Set	Number strings	Fraction nonself	Fraction unique nonself
AP	8600	0.540	0.340
PS	2966	0.435	0.196
LP1	1174	0.617	0.118
LP2	114	1.000	0.842
LP3	1317	0.102	0.002
SP1	36	1.000	0.833
SP2	285	0.165	0.130

Table 2: Features of nonself sets. *Number strings* is the total number in the trace, from first nonself string to last, including all self strings that occurred during that time. *Fraction nonself* is the fraction of the trace that consisted entirely of nonself strings, and *Fraction unique nonself* is the fraction of the total trace represented by unique nonself strings, which is the size of the nonself test set for the incident. AP = address probing, PS = large scale port scanning, LP = limited probing (scanning of a few ports), SP = single port probing.

datapath (the start of the incident), to the last nonself datapath. Thus, each incident reproduced the timing of the attack, as well as including all normal traffic that was interspersed throughout the attack. Table 2 gives a breakdown of the number of strings in each incident, including what fraction of the strings were nonself.

Although we have implemented and tested an on-line prototype, all results reported here are for off-line simulations of 50 detection nodes, corresponding to the 50 computers. Simulation increases the flexibility in the way the experiments can be set up and run; in particular, there is no limit on the number of time-steps for which the simulation can be run. This enables periods of months to be simulated in the space of hours, which was necessary to carry out extensive testing of various parameter settings. Another advantage of simulation is that it is easy to replicate results. We report elapsed time for the experiments in *days*. For all simulated data, one day is assumed to be equivalent to 25000 time-steps, that is, 25000 TCP SYN packets, because this is the average number of SYN packets observed in a day on the UNM subnet over a period of two months.

We assumed that the trace of self strings could be modeled by a strongly-stationary, discrete-time, random-process, described by the sample distribution extracted from S_t. In other words, at each timestep i, a string s_i was randomly drawn from S, weighted by its frequency of occurrence in the original trace, S_t. Figure 8 compares the distribution of a trace of simulated self strings with the distribution of S_t, over the same length of time. The curves are identical except at low probabilities (0.00001), which indicates that our simulated data has almost the same probability distribution as the original, real data. We have to generate self this way because our trace of self, S_t, covers only 50 days, and we may need to run experiments for more than 50 simulated days.

Many different simulations were performed to understand the effects of the various mechanisms and parameter settings [20]. In this paper we summarize results for one particular set of parameter values (see table 3)[15]. For these parameter settings, we ran 30 individual simulations and averaged results over these. All detectors began each simulation as immature, and the system was run for 30 simulated days of normal (self) traffic. After these 30 days, LISYS was challenged with the intrusive incidents. LISYS was presented

[15]For these experiments we implemented multiple representations using a "byte-and-hash" algorithm. See [20] for details.

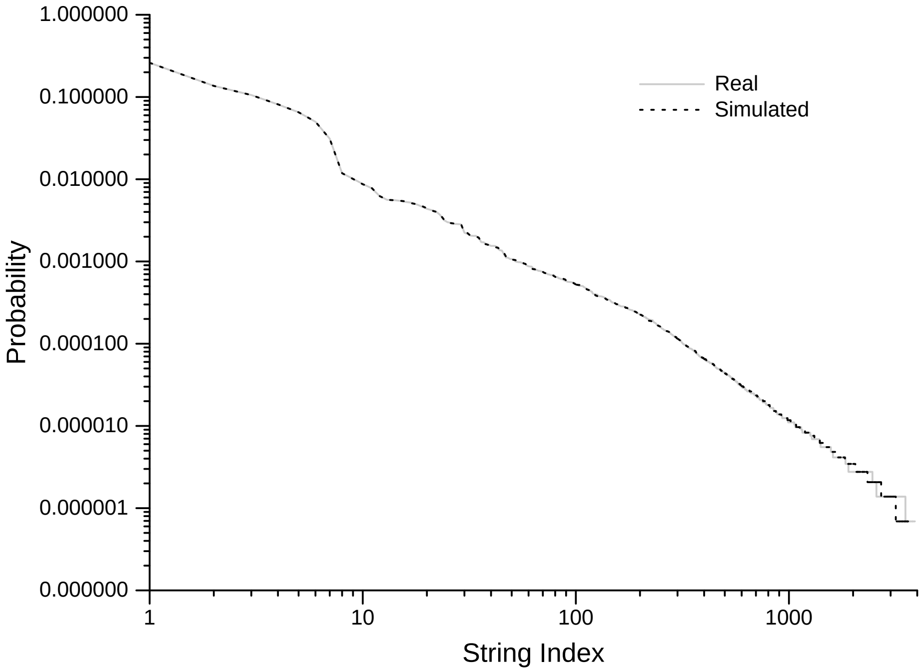

Figure 8: The probability distributions for real and simulated self. Note that both axes are logarithmic.

Parameter	Description	Value
ℓ	string length	49 bits
r	match length	12 bits
τ	activation threshold	10 matches
$1/\gamma_{match}$	match decay period	1 day
$1/\gamma_{\omega}$	sensitivity decay period	0.1 days
T	tolerization period	4 days
T_s	costimulation delay	1 day
$1/p_{death}$	life expectancy	14 day
n_d	number detectors per node	100 detectors

Table 3: The parameters for the simulation. Although the simulation uses 25000 time-steps per simulated day, parameter values are reported in terms of days for clarity. The decay periods are the expected time to decay by one. All 50 detection nodes have the same number of detectors, n_d.

with the incidents one after the other, each one separated by a buffer zone of a simulated day's self traffic.

The results of the simulations are reported in table 4. The false positive rate for each run of the simulation was averaged over the last 20 days, to ensure that transients have died out. Transients exist because all detectors are initially immature. These false positive rates are very encouraging; in the intrusion detection community, less than 10 false positives per day is regarded as very low [27]. Moreover, because of the implementation of costimulation, a false positive requires no action on the part of a system administrator. The low false positive rates were achieved without compromising the ability to detect intrusions: LISYS correctly detected all seven incidents. Note that because each intrusive

Aspect	Result
Fraction immature	0.23±0.01
False positive rate	1.76±0.20
Incidents detected	7 out of 7

Table 4: Summary of performance results. "Fraction immature" refers to the fraction of all detectors that are immature, averaged over the last 20 days of the simulation; "false positive rate" refers to the number of false positives incurred per day over the last 20 days; and "incidents detected" refers to the number of intrusive incidents detected. In the "result" column, the first number is the mean over 30 runs; the second figure (following the ± sign) is a 90% confidence interval in the mean.

incident consisted of multiple nonself strings (see table 2, it sufficed to detect only a few nonself strings during an intrusive incident. However, the more nonself strings detected per incident, the easier it is to separate false positives from true positives. In the results reported here, at least 44% of the nonself strings present in each incident were detected.

4.3 ARE ALL THESE MECHANISMS REALLY NECESSARY?

We have included several complicated mechanisms in LISYS, because they have been inspired by the immune analogy and because we believe them to be necessary. But are these mechanisms really necessary? In this section, we discuss the effects of some of the mechanisms, and show that they are indeed useful (and sometimes essential). These points summarize experimental results reported elsewhere [20].

- *Activation thresholds* reduce false positives. In our simulations, false positives were reduced by a factor of close to τ, for values of τ from 1 to 10 (τ is the activation threshold — see section 3.2). Moreover, activation thresholds did not have a significant negative impact on detection of intrusive incidents, because the nonself strings comprising the incidents occurred in a short period of time.

- *Sensitivity levels* increase the ability to detect diverse nonself strings, provided that they are temporally clumped. We simulated a distributed coordinated network attack (DCA), launched from many different locations so that each nonself string in the incident is very different from the others. DCAs can be particularly difficult to detect because the attack is launched simultaneously from many different locations, each of which carries out a limited portion of the attack, so that suspicion will not fall on a single location. Diagnosis of such attacks usually requires collation of information from many different locations. The sensitivity mechanism improves the detection of nonself strings in the simulated DCA by up to an order of magnitude.

- *Tolerization* is essential for reducing false positives. False positives increase exponentially with decreasing tolerization periods, changing from 1.7 per day for $T = 4$ to 15 per day for $T = 0.5$. There is a trade-off that determines the best length for the tolerization period. The longer the tolerization period the lower the probability of a false positive, but longer tolerization periods result in detectors remaining immature for longer, and hence not contributing to the detection of anomalies. Longer tolerization periods also increase the risk of the system being tolerized to infrequently occurring nonself.

- *Costimulation* eliminates mature, autoreactive detectors and consequently reduces false positives. Costimulation was found to reduce false positive rates by a factor of 3.

- *Multiple representations* improve detection rates when the relevant nonself strings are similar to self strings. In the NID application, using multiple representations improves detection by up to a factor of 3 when attempting to detect nonself strings similar to self.

- *Anomaly detection* is essential for detecting nonself strings that have not been encountered previously. In the simulations, we successfully detected 7 incidents that LISYS had never before encountered in the simulation.

- *Memory detectors* improve detection of previously encountered nonself. We used a set of 1000 nonself strings to test signature-based detection. The set of nonself strings was randomly-generated so that each string represented a new connection between two internal computers. We did this because such connections are closer to the normal profile of connections and thus we expect them to be harder to detect. When the (steady-state) system was initially challenged with the nonself strings, it detected 25% of them, in what can be regarded as a primary response. We allowed the system to retain up to 10% of its detectors as memory detectors (which had activation thresholds reduced from $\tau = 10$ to $\tau = 1$), and challenged it again 20 days later with the same set of nonself strings. This time it detected 75% of the strings, in a secondary response, clearly illustrating the utility of memory detectors [16].

- *Finite lifetimes* ensure that gaps in detection coverage are not static and predictable, and improve the secondary response (without finite lifetimes, the secondary response is no better than the primary response). Clearly, if detectors lived indefinitely and detection coverage was incomplete, any adversary that discovered the gaps in the coverage could repeatedly exploit them without fear of detection. Moreover, having finite lifetimes allows the system to more easily adapt to changing self sets, even without costimulation, because autoreactive detectors will eventually die off and be replaced by new immature detectors which will be tolerized to the new self set. However, there is a trade-off in selecting the lifespan: As the lifespan decreases, so the tolerization period must decrease, otherwise there will be an increasing number of immature detectors, but a decreasing tolerization period will increase false positives.

4.4 EXTENSIONS

There are several components of ARTIS that we have not included in LISYS. Our detectors are not mobile and they do not replicate when activated. As discussed in section 3.4, in ARTIS, memory detectors copy themselves and spread out to neighboring detection nodes. What is required in LISYS are mechanisms whereby copies of memory detectors migrate to other computers in the LAN. This would improve the robustness of signature-based detection in the distributed system, because all computers would have a similar set of memory detectors, so loss or compromise of a few computers would have no effect on efficacy of the secondary response. Mobility and replication should be tested on a live network.

Currently, LISYS is not completely autonomous, although its learning mechanisms are largely unsupervised. An important area of future work would be to make LISYS

[16]Note that a true secondary response would not only detect anomalies more quickly, but devote more resources to response, which we have not yet modeled.

autonomous by removing the human operator. This would require implementing an automated response system with possible selection of response mechanisms analogous to effector selection in the IS. The basis of the response system could be the modification of access to network services, so that hosts causing anomalous packets would be denied access to the relevant services. The response system could be viewed as having flexible, dynamically adaptive firewalls on every computer on the LAN. Automated response is a tricky issue, however, because an incorrect response can interfere with the legal functioning of the system and cause denial of service. Incorrect responses are analogous to autoimmune responses, and we hope that a form of costimulation can be implemented that will minimize these harmful responses. To do this requires some indicator of damage in the network environment; one idea that we are investigating is using a host-based intrusion detection system to provide damage indications [41]. We can also monitor system performance to characterize damage in terms of degraded performance during denial of service attacks.

4.5 PROPERTIES OF LISYS

As a consequence of the immune analogy, there are many important properties that LISYS exhibits. Most of these are not exhibited by current NID systems; although many systems exhibit some of these properties, none exhibit all of them.

- LISYS is *robust* in that it continues to function in the face of compromise or malfunction of some computers. This is desirable because it makes the system more reliable and makes it harder for an adversary to subvert. LISYS achieves this because it is distributed and no communication is required between detection nodes. Loss of a few detection nodes will only result in a gradual degradation in performance.

- We can *tune* the trade-offs between resources used and effectiveness of the system, and between false positives and false negatives. This is desirable because different domains will have different requirements, and even within a single domain, requirements may change over time. LISYS achieves this by varying the number of detectors: The more detectors, the better the detection rate, but the more computational resources required. The trade-off between false negative and false positive errors can be tuned by adjusting the activation threshold.

- LISYS is *scalable* in that adding more nodes will not increase the computational requirements of any existing node. This is desirable because the system should be scalable to very large networks. LISYS achieves scalability because there is no communication between nodes and detection nodes operate independently.

- LISYS performs *anomaly detection*. This is desirable because new attacks are always occurring and most NID systems will not be able to detect them. LISYS implements anomaly detection through the use of negative detectors and the negative selection algorithm.

- LISYS performs signature-based detection, automatically extracting signatures from nonself data. This is desirable because it can speed up detection and increase discriminatory accuracy. Moreover, it is desirable to automatically extract signatures because that reduces dependence on a human operator and speeds up the process of distributing information concerning attack signatures. The retention of memory detectors and the competition to become memory detectors implements signature-based detection in LISYS.

- LISYS is *accurate*: It achieves low false positive rates (two per day) with clear detection of intrusive activity (7 out of 7 incidents clearly detected). This is desirable because we want to minimize harm to the system, both in terms of damage an attacker could do, and in terms of damage an incorrect response (either automated or from a human operator) could do.

- LISYS is *adaptable*: It can initially train itself without human supervision, and it can later adapt to changes in normal behavior without human input. This is desirable to ensure autonomy of the system, because networks being monitored will always be changing. In LISYS, tolerization and finite detector lifetimes allow the system to adapt.

- The low resource requirements of LISYS make it *lightweight*. This is desirable because NID systems with expensive resource requirements will probably not get used. The results we achieved with LISYS required only 100 detectors per node, which is 100 49-bit binary strings per computer, with negligible running costs.

4.6 LIMITATIONS

Several limitations are related to the fact that LISYS currently monitors TCP datapaths on a broadcast network. Intrusions can involve network traffic at different protocol layers, for example UDP; these would not be detected by LISYS. LANs are moving away from broadcast networks to switched Ethernet, so that each computer sees only the packets that are destined for it. LISYS would lose many of the advantages of distributed monitoring in a switched network. Possible ways of overcoming this limitation include using programmed switches or active networks [39] to broadcast SYN packets. Characterizing self in terms of datapaths will not work for services which are expected to connect to any possible other computer at any time, such as WWW or FTP servers. Hence any attack that exploits vulnerabilities in these services will go undetected. Monitoring a different characteristic (e.g., packet frequencies) could address this problem.

Activation thresholds are useful for reducing false positives, but they also introduce paths of attacks. If activation thresholds greater than one are used to reduce false positives, then an attacker can evade detection in two ways. Firstly, by making anomalous connections occur infrequently enough so that they will never accumulate to the point where they could trigger a detector, and secondly, by ensuring that during any given attack the number of connections made is fewer than that required to activate a detector. As the frequency of connections used decreases, so the skill and patience of the attacker must increase. This limitation does not apply if an organization has the resources to cope with the higher false positive rates that are a consequence of minimal activation thresholds.

We have assumed that the problem domain can be divided into two sets of events: a set of legitimate and acceptable (according to some policy) events, and a set of illegitimate events. In reality, there can be events which are legitimate at some times and illegitimate at other times. Such ambiguous events violate the assumption and cannot always be correctly classified by LISYS. Another underlying assumption is that policy (under which the self and nonself sets are defined) can be implicitly inferred by observing the behavior of the system, and assuming that either self occurs more frequently than nonself, or that there is some period of time during which the self set can be collected separately from nonself. This is not only a limitation of this research, but a limitation for anomaly detection systems in general[17]: Some assumptions must be made about the relative frequency of occurrence

[17] Including the IS.

Classifier Systems	ARTIS
classifier condition	detector
classifier action	detector response
$1, 0, \#$ matching	r-contiguous bits
classifier strength	immature, mature, activated, and memory states
message list	bit strings representing patterns of interest
bidding for messages	competition to become memory detectors
more specific match wins	more specific match wins
support	activation threshold
message intensity	sensitivity level
internal messages	migrating detectors
bucket brigade	memory-based detection
genetic algorithm and triggering	negative selection
NONE	multiple representations

Table 5: Tentative comparison of ARTIS with classifier systems.

and/or distributions of the self and nonself sets.

5 RELATION TO CLASSIFIER SYSTEMS

ARTIS resembles the architecture of a classifier system [24]. As we mentioned earlier, immunologists often describe the problem solved by the immune system as that of discriminating "self" from harmful "other" (or "nonself") and eliminating other. In the language of classifier systems, the immune system must process external and internal messages, first classifying them as self or nonself, and in the case of dangerous messages taking appropriate action.

The parallel between immunology and classifier systems was noted as early as 1986 in [10]. In this work, a classifier system was used to model the immune system, by drawing an analogy between individual classifier rules and antibody types. Classifier strength represented the concentration of the antibody type, and interactions between classifier rules modeled Jerne's idiotypic network hypothesis [26]. Although the comparison was interesting, it had relatively little impact on classifier system research, and the two fields have continued to develop largely independently. Since that time, theoretical immunology has advanced significantly, and emphasis has switched from idiotypic network theory to other aspects of immunology.

Table 5 shows how the components of ARTIS correspond to the components of a learning classifier system, although we draw the analogy somewhat differently from [10]. The mapping between classifier systems and ARTIS is not 1-1, although the architectural similarities are striking. Because almost all of the implementation details (e.g., the details of matching rules) are different, while the overall architecture is largely preserved, ARTIS provides an interesting comparison with traditional classifier systems. In this section, we point out both the similarities and differences between these two systems.

Each detector d corresponds to the condition part of a classifier, where the match rule is r-contiguous bits instead of the traditional $1, 0, \#$ alphabet used in classifier systems. The parameter r is a measure of the specificity of the detectors, much like the number of don't cares in a classifier condition is a measure of its generality. In the current implementation of ARTIS, there is nothing corresponding to the action part of a classifier rule. However, if we concatenate some bits to each detector to specify a response (analogous to different antibody isotypes discussed in section 3.9), then each detector "cell" (detector plus response bits) would correspond quite directly to the condition/action rule format of classifier systems.

Less directly analogous are activation thresholds, which roughly correspond to Holland's proposal for *support*, and sensitivity levels which serve a similar role to message *intensity*. In both cases, the ARTIS mechanism is quite different from that used in classifier systems, but the reason for the mechanism is similar—in the one case to aggregate information from multiple sources and in the second case to vary the sensitivity of the system dynamically. Both activation thresholds and sensitivity levels decay over time, similarly to the role of tax in classifier systems.

In place of the message list we have a continuous flux of binary strings which represent the current state of the environment (for example, in LISYS these binary strings represented datapath triples). Internal messages in classifier systems are perhaps analogous to migrating detectors that move from one node to the next, although the analogy is not perfect.

There is no direct analog of the IS negative-selection algorithm in classifier systems, except for the learning rules (such as genetic algorithm and trigger conditions) under which new classifiers are generated. Bidding for messages in classifier systems is analogous to immune cells competing to bind to foreign pathogens. Likewise, we introduced pressure for specificity in ARTIS–which is reminiscent of classifier systems–by allowing the more specific match to win the competition to become a memory detector.

The role of the bucket brigade (credit assignment) and the genetic algorithm is played by our competition for memory system of learning, although ours is simpler in the sense that we assign credit directly from the environment to the detectors, and do not pass strength among immune cells. A more direct analog of the bucket brigade would occur if we tried to build up idiotypic networks of immune cells in which immune cells stimulate and repress other immune cells, as Jerne proposed [26]. Although this is appealing from an adaptive design perspective, there is little if any experimental evidence that such networks exist in natural immune systems.

ARTIS is essentially a stimulus/response system, where the stimuli are binary strings, classification of inputs does not involve a large amount of internal processing, and the response is an appropriate action (for example, in LISYS the response is an email message to a human operator). The natural IS is considerably more complicated, with highly complex internal regulatory mechanisms and several different kinds of potential responses. The regulatory mechanisms appear to be implemented through signaling molecules such as cytokines (discussed earlier); these cytokines can be viewed as analogous to internal messages in classifier systems. We could increase the role of internal feedbacks and self-regulation in ARTIS by extending the cytokine system (of which the sensitivity level is a primitive form).

In classifier systems, each classifier's *strength* is represented by a real number. A classifier's strength determines the probability of it being deleted or replicated through the genetic algorithm. In ARTIS, each detector is in one of several discrete states: immature, mature, activated, or memory. Which state it is in determines the likelihood of it being deleted, replicated, or mutated.

Multiple representations have no direct analog in classical classifier systems. However, they do provide a natural partitioning of the set of detectors. One weakness of the original classifier system proposals is that the architecture is completely flat. By "flat" we mean that all classifier rules are at the same level and there are no mechanisms for aggregating rules into "subassemblies" that form coherent units and have well-defined interfaces with the rest of the system. Tagging is sometimes proposed as a mechanism for automatically partitioning the rule sets into related groups of rules, but it has not been convincingly shown to work for this purpose. We speculate that different detector sets might discover different kinds of regularities (due to the combination of multiple representations with the locality of the r-contiguous bits matching rule). In this way, multiple representations could prove to be an initial step towards the kind of adaptive partitioning that has eluded classifier systems.

6 OTHER APPLICATIONS

Earlier we claimed that ARTIS is a general architecture for problem solving. To this end, we described ARTIS abstractly, and then applied it a particular domain, that of network intrusion detection. In this section, we briefly review some other applications for which ARTIS could be suitable, to demonstrate the generality of the approach. We also describe (in section 6.5) features of a problem domain that would make it suitable for ARTIS.

The single most important consideration when applying ARTIS to a domain is to choose a suitable peptide, or characteristic, to monitor. Such a peptide must result in a normal profile that separates normal from abnormal or anomalous behaviors satisfactorily, i.e. with minimal errors, both false positive and false negative.

6.1 MOBILE AGENT SECURITY

A mobile agent is a piece of software or code that copies itself to networked computers, and then runs on those computers. The code can also be modified, or *evolve*, as a consequence of interactions with other agents and software on computers. Already there are several examples of commercial mobile agent frameworks; see [4] for examples of mobile agent uses. Mobile agents offer increased flexibility for computing tasks. However, flexibility comes at the price of increased vulnerability, both to malicious users exploiting the system, and to poorly programmed or accidentally corrupted agents that could spread and cause damage like a cancer.

Conventional models of computer security are not suitable for a mobile agent framework; application of such models could limit the flexibility to such a degree that the mobile-agent frameworks become mere curiosities. An example of this is the "sand-box" concept for Java applets. Java applets are mobile agents that run within a sand-box on a computer, where the sand-box is an interpreter that prevents the applets from harming the host computer. However, there are two problems with using a sand-box: Firstly, the applets are limited in what they can do because they are constrained to the sand-box, and secondly, if what users care about is interactions between applets *within* the sand-box, then this form of security is useless. Essentially the sand-box is the same concept as the old fortress model of security, and although it is useful to a limited extent, it is generally too static to allow the kind of flexibility that makes mobile agents so promising.

We suggest that mobile agent security could be implemented using a form of ARTIS. The peptide could be a collection of MD5 hashes of agent code: We subdivide the agent code into several pieces and compute a hash for each piece. Then, small changes in the agent code will result in small changes in the overall peptide (i.e only one of the hashes will change). The anomaly detection problem is then one of identifying unacceptable agent behavior and associating it with particular hashes so that an agent and its (possibly mutated) offspring can be eliminated. The detectors could be implemented by agents themselves, which has the advantage that the detectors can monitor each other. The second signal could be provided by a human being, but it may also be possible to use some form of a damage signal. For example, agents that are terminated incorrectly by malicious agents could be an indicator of damage. Successful detection of a malicious agent would lead to elimination of that agent, and the detector agent would proliferate (copy itself) and spread across the network to eliminate all copies of that malicious agent.

6.2 CONSISTENCY OF DISTRIBUTED INFORMATION

There is a class of applications that require information to be distributed across many locations. This information is replicated for reasons of robustness, or speed of access. The information across these locations must be consistent, and there can be no central point of failure, or no bottleneck in communications. One example of such an application is the Domain Name Service (DNS) [30, 1]. The information used by DNS is domain name to IP address mappings, and for reasons of efficient access, these mappings are replicated throughout the Internet in local name server caches. This information needs to be consistent, to ensure accurate resolution of (domain name, IP address) pairs. Other examples are distributed fault tolerance, where the goal is to detect inconsistent server ordering; distributed authentication, where each location capable of authenticating must contain some information to implement that authentication (for example a database of users and their passwords); and key distribution, where cryptographic keys are copied across many locations to allow for encrypted communication.

We can apply ARTIS to this problem in the following way. A peptide would be a piece of information, perhaps compressed in some way. So, for example, with the DNS application, a peptide will be a compressed representation of a (domain name, IP address) pair. The anomaly detection problem facing ARTIS is to identify and eliminate incorrect pairs, which could be the consequence of accidental or deliberate corruption. Each detector could be encapsulated as a mobile agent, so that these agents could migrate through the network, checking for incorrect mappings. The detectors could also check each other, ensuring that corrupted detectors were controlled and kept in check. The response of ARTIS upon finding a corrupted mapping would be to eliminate that mapping and distribute copies of the relevant detector throughout the network to eliminate any spread of the corrupted information. The solution provided by ARTIS is asynchronous, and would only work for applications (such as DNS) where occasional inaccuracies are tolerated.

6.3 EPIDEMIOLOGICAL MONITORING

It is often difficult to detect the spread of new diseases. Doctors and medical practitioners faced with new symptoms will have difficulty associating them with a newly emerging disease and may not have the time or resources to investigate all possibilities. Furthermore, it is hard to correlate similar disease symptoms across widely diverse geographic areas, and so medical practitioners may not be aware that the novel symptoms they are observing are also being observed in several other places throughout the world. We propose that ARTIS could provide some assistance in early detection and correlation of diseases.

In this application, a peptide would be some representation of patient symptoms, and the anomaly detection problem would be to detect sets of unusual symptoms that were indicative of outbreaks of epidemics or new diseases. The detectors could be mobile agents moving over the Internet, from one local medical database to the next. Each local medical practitioner would retain an accessible database of their current patients' symptoms, and these would be checked using the roaming detectors. The second signal could be confirmation from a medical practitioner that the symptoms were indeed unusual. The response would be, as usual, to replicate the detectors and distribute them far and wide, and in addition to send an alarm and to inform the Centers for Disease Control, or some other similar body. The spread of detectors would enable local medical practitioners to more reliably identify a new disease, because detectors would be available that expressly identified a set of symptoms with the new disease. Furthermore, tracking clones of detectors and where they get activated would allow us to determine how and where a disease was spreading.

6.4 FINANCIAL TRANSACTIONS

Banks and other financial institutions depend upon a flow of money across many distributed locations. It is critical for them to detect fraud to protect their interests. If there is some sort of consistency of normal money flow, then we propose that ARTIS could be used to monitor financial transactions for the detection of unusual transactions indicative of fraud.

A peptide would be an encoding of a financial transaction. For example, we could encode in the peptide the source and destination of the money flow, and the amount involved. The anomaly detection problem would be that of detecting unusual transactions that are indicative of fraud. Detectors could once again be mobile agents that migrate across networks linking banks, financial institutions, etc. Upon detection of anomalous transactions, officials would be informed, who could then provide the second signal.

6.5 WHAT MAKES A DOMAIN SUITABLE FOR ARTIS?

The previous examples all have common themes. Here, we enumerate the features that we believe will characterize the suitability of ARTIS to any particular problem domain. As the number of these features exhibited by a problem domain increases, so we would expect the suitability of ARTIS to increase. However, ARTIS could still be used in domains that exhibit a few or none of these features, but in those cases there are likely to be other systems that will be more suitable. ARTIS is suited to applications that:

- Require pattern classification and response;
- Require a distributed architectural solution;
- Require a distributed architecture scalable to environments with arbitrary numbers of nodes;
- Have a high communication cost between nodes;
- Address problems for which there is some commonality of patterns across the nodes, i.e. multiple nodes see the same or similar patterns within some limited time period;
- Require the detection of novel anomalous patterns;
- Are dynamic in that normal behaviors change slowly over time;
- Require a robust solution with no centralized control;
- Have storage capacities on any single node that are small compared to the amount of information required to represent all possible normal patterns;
- Require a trade-off between resource consumption and accuracy of performance.

7 CONCLUSIONS

We have applied ARTIS to a specific aspect of intrusion detection, that of monitoring TCP packets in a network. However, we envision ARTIS being used as a broad framework for intrusion detection in general. Because ARTIS is described in the abstract, independent of particular applications, its architecture could be used as an intrusion detection system that monitors and responds to many characteristics other than network connections, for example, system calls executed by processes, profiles of user behavior, the contents of network packets, etc. The issue of effector selection becomes critical within a broad intrusion detection framework, because different violations will require different responses, for example, violations detected by monitoring system calls would require halting or suspending processes, whereas violations detected by monitoring user behavior may require terminating a user's session. Effector selection also includes determining responses to corrupted detectors, because ARTIS is self-monitoring (detectors are represented by bit strings and so can be monitored in the same manner as any other strings).

The IS is much more than a simple anomaly detection and response system. It can be viewed as a general pattern-learning system that is highly distributed and scalable. Not only does it learn to classify patterns as self or nonself, but because it must respond differently to different pathogens, it further learns to classify nonself into a variety of classes. Likewise, ARTIS is also a general pattern-learning system that can be used to learn to discriminate between a wide variety of specific patterns, and respond differently to different patterns. We can make this aspect more explicit in ARTIS by allowing the universe to be partitioned into multiple sets, $U = N_1 \cup N_2 \cup N_3 \cdots \cup N_k$, and training different detectors so that they only recognize a single subset as nonself. For example, assuming we have samples of all $N_1, N_2, \ldots, N_k$ subsets, we can generate a detector d_2 that recognizes only elements of N_2 using the negative selection algorithm, with the self set defined as $N_1 \cup N_3 \cdots \cup N_k$[18].

Holland has defined the notion of a complex adaptive system (CAS), and argued that the IS is indeed such a system [23]:

> The human immune system is a community made of large numbers of highly mobile units called *antibodies* that continually repel or destroy an ever-changing cast of invaders called *antigens*. The invaders—primarily biochemicals, bacteria, and viruses—come in endless varieties, as different from one another as snowflakes. Because of this variety, and because new invaders are always appearing, the immune system cannot simply list all possible invaders. it must change or adapt (latin, "to fit") its antibodies to new invaders as they appear, never settling to a fixed configuration. Despite its protean nature, the immune system maintains an impressive coherence. Indeed, your immune system is coherent enough to provide a satisfactory scientific definition of your *identity*. It is so good at distinguishing you from the rest of the world that it will reject cells from any other human. As a result, a skin graft even from a sibling requires extraordinary measures. [p. 2]

There are several other artificial systems commonly regarded as CAS, including neural networks, genetic and evolutionary algorithms [21], classifier systems [24], and Echo

[18] An approach similar to this has been successfully used in [7]. The main difference with this alternative approach is that d_2 would be initially positively selected for N_2 using a genetic algorithm (the closeness of matches between d_2 and elements of N_2 would determine the fitness of d_2), and then negatively selected to ensure that it does not match any other subsets.

[23]. We believe that ARTIS is an interesting addition to the repertoire of artificial CAS. We have already described the parallels between classifier systems and ARTIS in section 5; there are also similarities in the learning mechanisms of ARTIS and genetic or evolutionary algorithms. The memory competition between activated detectors supplies selective pressure, with fitness being measured in terms of how well an activated detector matches nonself, and variation is provided over time because detectors have finite lifetimes. Variation could be enhanced by incorporating somatic hypermutation, a feature of the IS whereby activated B-cells undergo high mutation rates during cloning.

Acknowledgments The authors gratefully acknowledge the support of the Intel Corporation, the Defense Advanced Research Projects Agency (grant N00014-96-1-0680), the National Science Foundation (grants IRI-9711199 and CDA-9503064), and the Office of Naval Research (grant N00014-99-1-0417).

References

[1] P. Albitz and C. Liu. *DNS and BIND*. O'Reilly and Associates, Sebastopol, CA, 1992.

[2] ASIM. Information security - computer attacks at deparment of defense pose increasing risks. GAO Executive Report - B266140, May 1996.

[3] R. A. Brooks and A. M. Flynn. Fast, cheap and out of control: A robot invasion of the solar system. *Journal of the British Interplanetary Society*, 42:478–485, 1989.

[4] CACM. Special edition on agents, 1999.

[5] H. J. Chiel and R. D. Beer. The brain has a body: Adaptive behavior emerges from interactions of nervous system, body and environment. *Trends in Neurosciences*, 20:553–557, 1997.

[6] Mark Crosbie and Gene Spafford. Defending a computer system using autonomous agents. Technical report, Department of Computer Sciences, Purdue University, March 1994.

[7] D. Dasgupta, Y. Cao, and C. Yang. An immunogenetic approach to spectra recognition. In *Genetic and Evolutionary Computation Conference (GECCO)*, pages 149–155, 1999.

[8] P. D'haeseleer. An immunological approach to change detection: Theoretical results. In *Proceedings of the 9th IEEE Computer Security Foundations Workshop*, Los Alamitos, CA, 1996. IEEE Computer Society Press.

[9] P. D'haeseleer, S. Forrest, and P. Helman. An immunological approach to change detection: Algorithms, analysis and implications. In *Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy*, Los Alamitos, CA, 1996. IEEE Computer Society Press.

[10] J. D. Farmer, N. H. Packard, and A. S. Perelson. The immune system, adaptation and machine learning. *Physica D*, 22:187–204, 1986.

[11] S. Forrest, S. Hofmeyr, and A. Somayaji. Computer immunology. *Communications of the ACM*, 40(10):88–96, 1997.

[12] S. Forrest, S. A. Hofmeyr, and A. Somayaji. A sense of self for unix processes. In *Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy*, Los Alamitos, CA, 1996. IEEE Computer Society Press.

[13] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, Los Alamos, CA, 1994. IEEE Computer Society Press.

[14] S. Forrest, A. Somayaji, and D. Ackley. Building diverse computer systems. In *Proceedings of the 6th Workshop on Hot Topics in Operating System*, Los Alamitos, CA, 1997. IEEE Computer Society Press.

[15] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. Wolber. A network security monitor. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE Press, 1990.

[16] T. Heberlein. NID overview. http://ciiac.llnl.gov/cstc/nid/niddes.html, October 1998.

[17] H. Hendriks-Jansen. *Catching Ourselves in the Act.* MIT Press, Cambridge, MA, 1996.

[18] J. Hochberg, K. Jackson, C. Stallings, J. F. McClary, D. DuBois, and J. Ford. NADIR: An automated system for detecting network intrusion and misuse. *Computeres and Security*, 12(3):235–248, 1993.

[19] S. A. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 1998.

[20] Steven A. Hofmeyr. *A Immunological Model of Distributed Detection and its Application to Computer Security.* PhD thesis, Department of Computer Sciences, University of New Mexico, April 1999.

[21] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[22] J. H. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, Cambridge, MA, 1992. Second edition (First edition, 1975).

[23] J. H. Holland. *Hidden Order.* Addison Wesley, 1995.

[24] J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. Thagard. *Induction: Processes of Inference, Learning, and Discovery.* MIT Press, 1986.

[25] C. A. Janeway and P. Travers. *Immunobiology: The Immune System in Health and Disease, 3rd Edition.* Current Biology Ltd., London, 1996.

[26] N. K. Jerne. Towards a network theory of the immune system. *Annals of Immunology*, 125:373–389, 1974.

[27] R. Lippman. Lincoln Laboratory intrusion detection evaluation. http://www.ll.mit.edu/IST/ideval/index.html, October 1999.

[28] P. Matzinger. Tolerance, danger and the extended family. *Annual Review in Immunology*, 12:991–1045, 1994.

[29] P. Matzinger. An innate sense of danger. *Seminars in Immunology*, 10:399–415, 1998.

[30] P. Mockapetris. RFC1034/1035, 1987.

[31] B. Mukherjee, L. T. Heberlein, and K. N. Levitt. Network intrusion detection. *IEEE Network*, pages 26–41, May/June 1994.

[32] NetRanger. Netranger web site. http://www.wheelgroup.com/netrangr/lnetrang.html, October 1999.

[33] W. E. Paul. *Fundamental Immunology, 2cnd Edition.* Raven Press Ltd., 1989.

[34] J. K. Percus, O. E. Percus, and A. S. Perelson. Predicting the size of the antibody-combining region from consideration of efficient self/nonself discrimination. In *Procedings of the National Academy of Science 90*, pages 1691–1695, 1993.

[35] J. Piel. Life, death and the immune system, special issue. *Scientific American*, 269(3):20–102, 1993.

[36] P. Porras and P. G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *Proceedings National Information Systems Security Conference*, 1997.

[37] L. A. Segel. The immune system as a prototype of autonomous decentralized systems. In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, 1997.

[38] D. Smith, S. Forrest, and A. S. Perelson. Immunological memory is associative. In *Workshop Notes, Workshop 4: Immunity Based Systems, Intnl. Conf. on Multiagent Systems*, pages 62–70, 1998.

[39] J. J. Smith, K. L. Calvert, S. L. Murphy, H. K. Orman, and L. L. Peterson. Activating networks: A progress report. *IEEE Computer*, 32(4):32–41, 1999.

[40] S.R. Snapp, J. Brentano, G.V. dias, T.L. Goan, L.T. Heberlein, C. Ho, K.N. Levitt, B. Mukherjee, S.E. Smaha, T. Grance, D.M. Teal, and D. Mansur. DIDS (distributed intrusion detection system) — motivation, architecture, and an early prototype. In *Proceedings of the 14th National Computer Security Conference*, pages 167–176, October 1991.

[41] A. Somayaji. Personal communication, 1999.

[42] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GriIDS - a graph based intrusion detection system for large networks. In *Proceedings 19th National Information Systems Security Conference*, 1996.

[43] G. Vigna and R. Kemmerer. NetSTAT: A network-based intrusion detection approach. In *Proceedings of the 14th Annual Computer Security Applications Conference*, 1998.

