Immunology, Diversity, and Homeostasis: The Past and Future of Biologically-Inspired Computer Defenses*

Anil Somayaji

August 17, 2007

Abstract

While biological metaphors have a long history of use in computer security, they have been more successful in describing malicious software than in helping create better defenses. Immune system mechanisms, diversity, and homeostasis have helped inspire the development of promising computer security technologies. To fulfill the promise of biologically-inspired security, however, more work is needed in understanding how living systems defend themselves and in how those ideas can be brought to computers. This papers presents past successes, limitations, and opportunities for future work in this promising area. It then also addresses why there are significant barriers to such advances.

1 Introduction

From the coining of the term "computer virus" [6], biological metaphors have been used to describe computer security problems. Computer security defenses, however, have generally been described using other terminology, e.g., firewalls, virus scanners, patches, and intrusion detection systems. This difference in terminology is telling: while self-propagating malware has lifelike characteristics [29], current computer security defenses have few similarities to those used by living systems.

This is not to say, however, that there have not been effort to change this. The ideas of computer immune systems [19] and autonomic, "self-healing" software [20] have been heavily promoted both in the literature and in the press. One possible explanation for this discrepancy is that biological defense strategies may be inappropriate for computers. After all, living systems are not good at keeping secrets—we give away our genomes every time we lose a hair, for example. Further, parts of computer systems are not nearly as "disposable" as those of living systems—we don't like individual programs or computers to die.

While biological defense strategies may not always be appropriate, it is more accurate to say that we don't know whether they are appropriate or not. As explained in the rest of this paper, biologically-inspired defenses have barely tapped the richness of biological systems. Successes or failures, then, are better seen as evidence of immaturity, not inapplicability.

*To Appear in *Information Security Technical Report (ISTR)*. Preprint version.

To explore both the potential and the limitations of past work in biologically-inspired security, this paper first reviews the structure of natural immune systems in Section 2 and then selectively reviews past work on applying the biological concepts of immune systems, diversity, and homeostasis to building computer security defenses in Sections 3-5. Section 6 discusses future opportunities for biologically-inspired defenses, while Section 7 discusses potential barriers to those opportunities. Section 8 then concludes.

2 Immunology

Perhaps the most important sources of biological inspiration for computer security researchers have been natural immune systems. Like most aspects of living systems, natural immune systems are remarkable in their robustness, flexibility, and complexity. While the broad outlines of how the immune systems of mammals function are known (particularly those of mice and humans), there are numerous unknowns that greatly complicate any effort to use these systems as design templates for engineered systems. What follows, then, is a high-level description of the threats facing natural immune systems and the general approaches that are taken to address them. For a more detailed (but still high-level) introduction to the human immune system, see Hofmeyr [17]; for more on the design principles underlying the immune system, see Somayaji [27]. (For readers with some background in immunology, please skip to Section 3.)

The human immune system consists of numerous cell types: T-cells, B-cells, macrophages, mast cells, natural killer (NK) cells, neutrphils, and others—and each of these types is made up of many subtypes. These cells are constantly in motion, patrolling independently and self-organizing to classify and respond to problems. To keep these many cells coordinated and controlled, dozens of distinct chemicals (some of which are identical to those used by nerve cells) are used to exchange messages. The details of this extremely complex system, and many of the basic operating principles, are still being uncovered by immunologists everyday, meaning that textbooks are out of date the moment they are published. Because of these factors, it can be extremely difficult to understand what the immune system does and how it does it; however, by understanding the "threat model" of immune systems, though, we can begin to see some patterns.

Organisms must always contend with the fact that others wish to take advantage of their success. Being reproductively successful means that one becomes a potential resource for others. In the case of animals, predators are best addressed through changes in high-level behavior (e.g., running away, fighting, hiding). For threats where the attack takes place inside of the body, however, other strategies are needed. The skin protects most of the perimeter of an animal. Organisms are more vulnerable at entry and exit points, but those are also protected. For example, the stomach uses enzymes and powerful acids to destroy foreign organisms, while the lungs use mucous and cilia to trap and expel invaders.

It is when foreign organisms get past these barriers that the immune system gets involved. In architecture, the immune system is a distributed, decentralized network of autonomous agents (cells) that collectively can recognize and respond to attacking entities, or pathogens. Immune systems such as the human immune system face four basic challenges:

- **Scale:** Parasites can be single or multicellular eukaryotic organisms, so they are often much,

much larger than the individual immune system cells that must identify and destroy them. In contrast, bacteria and viruses are much smaller than the cells of the immune system, meaning that they can often hide *inside* the cells of the body—even immune system cells. This range of scales means the immune system needs a variety of detection and response strategies.

- **Communication:** Immune system cells coordinate their actions through the use of chemical signals; those same chemicals, however, can also be produced by pathogens. Compromised communications can result in responses being dulled or even prevented.

- **Errors:** Most responses produce some amount of "friendly fire"—the body's own cells are killed in the process of defending the body. Some deaths are unavoidable; however, others results from classification errors, some of which are exacerbated by the actions of the attackers. In extreme cases the actions of the immune system can cause more damage than the pathogen itself, even resulting in the death of the organism.

- **Evolution:** As pathogens are killed, those that survive reproduce. This evolutionary process means that they are constantly getting better at bypassing immune systems, and doing so on relatively short timescales: bacteria reproduce every 20 minutes, but humans reproduce every 20 years.

To address these challenges, the human immune system has to be able to respond quickly to well-known threats yet retain the capability to deal with the unknown. To address this dichotomy, the cells of the immune system are divided into two classes, the innate and adaptive immune system. The innate immune system functions in a way analogous to most modern virus scanners: its cells can recognize specific molecular markers and behaviors that are "known" signatures of pathogens. Because such signals can be easily distiguished from those produced by the normal cells of the body, the responses of the innate immune system are both swift and accurate. Unfortunately, like virus scanners, their responses are insufficient for novel threats.

To bridge this gap, the adaptive immune system does not attempt to develop detectors that match specific new pathogens a-priori; instead, it creates a large variety of detectors that can match virtually anything and then destroys or carefully regulates those that match the body's own cells, or *self*. By being so selected, the remaining detectors will only match against *nonself*, which in general will be pathogens. Note that the primary advantage of this *negative selection* process is that individual cells decide on their own whether something doesn't belong without each of them requiring a full inventory of what is allowed. Coordination only needs to take place to decide on the type and magnitude of response, rather than on the detection side.

When deployed in the form of antibodies or B-cells, the detectors primarily match exposed components of pathogens (e.g., proteins on the surface of a parasite). To detect cells that have been compromised internally (e.g., a virus-infected cell), additional mechanisms are required because under normal circumstances one cell cannot look inside another. One way this problem is circumvented is through the the interaction between T-cells and MHC-peptide complexes.

Proteins are the workhorse molecules of the cell: they are the ones that make chemical reactions happen inside of a cell, and they provide much of its structural support as well. As each cell recycles its proteins, they are chopped up into fragments known as *peptides* that are then recycled

into new proteins. The MHC molecule intervenes in this process by snatching up peptides and displaying them on the surface of the cell. These displayed peptides are queried by passing T-cells in the negative selection style of the immune system: if the T-cell matches the peptide, it was part of an abnormal protein, meaning that the cell has been somehow compromised. Thus, the T-cell signals that the identified cell should be killed.

3 Artificial Immune Systems

Given the complexity and scope of natural immune systems, it has not made sense to translate them "wholesale" into computer mechanisms; instead, what has happened is that aspects of immune systems have become templates for specific security mechanisms or architectures. Broadly speaking, applications of immune system ideas in computer security can be divided into three categories: negative selection-based systems, MHC-based systems, and other systems. What follows is a brief overview of the first two; we discuss other approaches in Section 6.

3.1 Negative Selection-based Systems

As discussed previously, negative selection is the process whereby the immune system selects detectors (e.g., T-cells): it randomly generates detectors that have the potential to match everything, and then it throws away those that match normally-behaving cells. Forrest [11] first applied this process to a computer security problem, specifically the detection of virus-infected files. Later, Hofmeyr [16] proposed LISYS, a network intrusion detection system based upon the negative selection algorithm, along with various immunologically-inspired mechanisms for managing false positives.

While these systems were both major advances, they both had certain basic limitations. When applied to individual computer systems, the virus detection mechanism didn't have any clear advantages to storing cryptographic checksums of files (e.g., [21]). LISYS had the limitation that it detected unusual network connections on the basis of IP addresses and ports, a choice that restricts its ability to detect many kinds of network attacks. Efforts by other researchers to increase the number of features observed ran into issues regarding accuracy and detector generation time [22]; a response by Balthrop [2] argued that such problems qarose not from limitations in the negative selection algorithm, but from the algorithm being applied in inappropriate ways.

In the field of Artificial Immune Systems (AISs) [7], many researchers have taken on the task of expanding the bounds of negative selection, applying it to new domains, studying its properties, and also developing algorithms based upon other immune system mechanisms. Such work has now largely moved from the area of computer security to that of machine learning. However, the more general idea of representing a set (self) by using detectiors that match its complement (nonself) may have significant applications in the area of privacy-preserving databases [8], among others.

While the negative selection algorithm has inspired significant amounts of research, its legacy, for the moment, is found in work that has very little connection with the basic problems for which immune systems evolved. While disappointing, this should probably be expected: current computer systems are not composed of billions of autonomous, disposable, self-replicating computing

elements. Architectures and algorithms that work well in such circumstances, then, should not be expected to automatically apply to other systems.

3.2 MHC-based Systems

The MHC mechanism, by allowing T-cells to observe the internal state of other cells, acts as a kind of cell-level anomaly detector that allows the immune system to uncover cells running "malicious code" (viruses). By equating cells with running programs, MHC suggests that intrusions could be reliably detected through the use of program-level anomaly detection. The question then becomes, what is an appropriate analog for a peptide?

In 1996 we first proposed that a possible peptide analog could be short, contiguous sequences of system calls [10]. Because most modern operating systems are still relatively centralized (even on multiprocessor systems), we did not need the distributed properties of the negative selection algorithm; instead, we used simple arrays [10] and later, trees [15] to represent sequences seen when training the system to recognize normal program behavior. The result of this work was a system that could detect a wide variety of security violations with few false positives.

These results stimulated significant amount of work in the area of program-level intrusion detection. Some researchers studied whether sequence-based analysis could be applied to other datastreams [30]; many more, though, chose to see whether other algorithms could do better than the ones we proposed (e.g., [23]). Because program behavior is extremely regular at the level of invoked system calls, it turns out that complex learning algorithms are not needed and provide little benefit [34]. However, by adding additional state such as program counter information, it is possible to build more precise models [25].

The direction of research in program-level intrusion detection changed, however, with the development of *mimicry attacks* [32]. The idea here was that attacks could be crafted such that they looked "normal" (i.e., they produced no novel system call sequences). Mimicry attacks have been perceived as a significant weakness of the system-call sequence method. Many researchers have now proposed "mimicry-resistant" intrusion detection methods (e.g., [9]); unfortunately, these approaches generally can either detect fewer attacks overall and/or have significantly higher false alarm rates, making them less suitable for production use.

To summarize, the initial idea of MHC-style attack detection has given rise to the area of program-level intrusion detection. In its approaches and mechanisms, however, there is little left of the original biological inspiration.

4 Diversity

Computer security researchers often complain about the dangers of overly popular software: a single vulnerability can potentially lead to millions of compromised machines. Such criticisms have more recently been framed in terms of a "software monoculture," in reference to the disease suseptibility of farms that raise single varieties of plants on large numbers of fields [13].

Uniformity in software, however, is something that is generally advantageous: it reduces the effort required for users to learn systems and for administratiors to maintain them. However, it

has long been recognized in the fault-tolerance literature that having diversity in implementations can increase robustness [5]. Creating multiple implementations by hand, however, is an expensive proposition, one that is impractical for all but the most safety-critical systems.

To achieve the benefits of diversity without the typical costs, we proposed that software implementations be "diversified" [12]. The idea is that some attacks, particularly memory corruption-based attacks such as buffer overflows, require the attacker to understand how code and data are organized in a targeted program. By changing this organization, attacks can be made to fail without disrupting normal program behavior. We implemented a simple diversification based on padding stack variables; other researchers have diversified (or, randomized) memory layouts [4], instruction sets [3, 18], and other aspects of low-level code behavior. Some, such as ASLR [24], have been so successful that such mechanisms are now part of Microsoft Windows Vista and many Linux distributions.

While these advances are significant, it should be noted that while the problem identified has been framed biologically, the solutions are not biological at all. Diversity in living systems comes from very different sources: sexual reproduction, death, and speciation. Sex allows for the production of unique but similar individuals, while death makes sure new variants (i.e., young individuals) have an opportunity to survive and prosper. Speciation allows groups of closely-related individuals to split so that they may become more different (genetically diverge) over time.

While sexual reproduction is the inspiration for genetic algorithms and related work, these methods cannot be directly applied to conventional code bases. Thus, it would appear the scope for truly biologically-inspired diversity techniques is quite limited. We will return to this issue in Section 6.

5 Homeostasis

All biological systems maintain a stable internal state by monitoring and responding to internal and external changes. This self-monitoring is one of the defining properties of life and is known as "homeostasis." Homeostatic mechanisms are typically autonomic, i.e., they operate below the level of consciousness. But homeostatic mechanisms have a very specific purpose, to minimize variations in the internal state of an organism. In effect, homeostasis refers to mechanisms that ensure that the "assumptions" made by an organism's internal systems stay true.

Two classic examples of homeostasis are the temperature and fluid balance mechanisms of our own bodies. Such mechanisms are essential because significant changes in these can have catastrophic consequences. The human body detects internal changes through sensors (specialized nerve cells) in the skin and inside the body. As we become too cold, our peripheral blood vessels constrict and we shiver; when we become too hot, peripheral blood vessels dilate and we sweat. Similarly, our kidneys secrete more or less water in our urine depending upon how hydrated we are.

Because electronics, like living systems, are sensitive to temperature and voltage extremes, computer systems have sensors (thermistors, silicon-embedded diodes) and regulatory mechanisms (fans, voltage regulators) analogous to biological homeostatic mechanisms. At the software level, though, there is little correspondence.

This discrepancy is significant because natural immune systems can be thought of more as homeostatic mechanisms than strict defense mechanisms. Any immune response must be proportional to the change detected, much as the violence of our shivering depends on how cold we are. Further, the type and severity of an immune response must be balanced against other factors rather than just being relative to the maliciousness of the threat. Because immune responses kill healthy cells, the immune system must always trade-off destroying the pathogen enemy with destroying the body itself. Thus, rather than just trying to destroy pathogens, the immune system controls pathogens such that the cells of the body can continue to live.

In an attempt to make a more homeostatic computer defense mechanism, I developed pH ("process homeostasis") [28], a Linux kernel extension that delays the execution of unusually behaving processes. pH is, in effect, an unusual CPU scheduler that favors normally-behaving processes. By greatly reducing the CPU cycles received by compromised programs, attacks are effectively stopped while legitimate computation proceeds.

The delay mechanism of pH was a forerunner to the network-level virus throttling of Twycross and Williamson [31]. In their system, the rate of network connections was throttled in order to limit the spread of malicious programs. While their approach is less adaptive than pH, it is also easier to predict and understand. No current commercial intrusion detection or prevention system delays unusually behaving processes (as measured by an adaptive anomaly detector); the virus throttle technology, however, has been commercialized by HP and is available in some of their router products.

6 Future Opportunities

From the preceding overview of work inspired by biological immune systems, diversity, and homeostasis, it should be clear that existing work has borrowed very few ideas from nature. There has been other work at biologically-inspired security mechanisms—innate immunity [14], the "danger" model of immune system function [1], self-healing systems [26], for example. However, negative selection, MHC, and the general ideas of diversity and (to a lesser extent) homeostasis are the biological ideas that have most influenced the development of existing computer security technology.

But even the influence of these ideas has been more inspirational than practical; while the respective phenomenas in nature are all immensely complicated and are far from being fully understood, the ideas at the heart of the computer science translations can be summarized in a few sentences. Translating negative selection does not mean that we understand the dynamics of T-cells, let alone B-cells or macrophages; translating MHC does not mean that we understand exactly how the immune system detects and responds to virus-infected cells. There are many, many more deep and inspiring ideas in living systems waiting to be translated.

There is also very good reason to continue with this translation process: many of the properties that we admire in living systems are not properties of our computers—we do not have computers that can autonomously detect and respond effectively to novel attacks, for example, even in the research literature. Software systems need constant monitoring and maintenance by people, and single flaws can still result in millions of computers being compromised completely silently. Process-

level anomaly detection, implementation diversity, delay-based automated response—these are steps forward, but only small steps.

For example, consider the lackluster adoption of program-level anomaly detection. One fundamental reason why anomaly detection has not been widely adopted is that when anomalies are detected, it is not at all clear what they mean. Right now an administrator has to examine system logs, network traces, and whatever other information they can obtain in order to decide whether an anomaly was or was not an attack.

It turns out that the "anomaly classification" problem is significant for living systems as well. A T-cell cannot just kill a matching cell on its own; it must first be activated in some way by a secondary signal. The processes controlling such secondary signals are very complex and are still being unravelled. Indeed, the complexity of the immune system seems to lie not so much in the sensors but in how information from them is combined with other inputs in order to make correct decisions reliably in the face of attacker interference. Thus, in order to addres the limitations of biologically-inspired security mechanisms, perhaps the solution is to see how biology dealt with those limitations [16].

Along such a path of adopting ever more biology, however, there are many barriers to be faced. We consider these issues in the next section.

7 Barriers

After having now long worked in this divide between computer security and biology, I've come to see that there are significant technical, cultural, and conceptual barriers that have hampered, and likely will continue to hamper, such efforts. There is reason to think, however, that we will have to overcome these barriers because of current trends in the computer systems. In what follows I discuss the technical, cultural, and conceptual barriers to biologically-inspired security, followed by a discussion of future trends.

7.1 Technical Barriers

As explained earlier, biological systems are extremely complex, interconnected systems. As such, biological mechanisms cannot be imported wholesale to computer systems—translations must be done piecemeal. But even if one chooses the "right part" to translate, how do you do the translation? How do you make the metaphor concrete? Simply giving parts of your system biological names isn't enough—the mapping has to somehow capture and use relevant patterns from biology.

As a step towards this, we previously identified several organizing principles that appear to be at work in natural immune systems [27]. Some principles, such as "defense in depth," are straightforward to apply and are already widely used; others, such as "no secure layer" (or, more properly, "no trusted layer") are much more problematic: currently, we simply don't have any design methodologies that produce systems that do not assume that some part of the system is always secure. The real problem, however, comes with concepts like "disposability"—while our bodes barely notice the destruction of hundreds or even thousands of cells, the corruption of a

single bit can bring down computers and even networks. Thus, solutions based upon disposability may have requirement that current computers cannot satisfy.

Of course, this all assumes that we understand what is going on in living systems—an assumption that is all too often false. For example, how exactly does the immune system detect rogue cells (e.g., precancerous cells) without aggressively attacking normal cells? How are specific responses chosen? How does the immune system reliably sample "self," and how does it account for the self peptides that it must overlook? These basic questions are not completely understood by immunologists. With such incomplete knowledge, it is hard to look to biology to create systems that solve analogous problems.

7.2 Cultural Barriers

Initially, biological terminology was welcomed in the computer security community as an interesting, new approach. Now, though, it is generally recognized that biological terms in new papers are signs of poor quality research. While I would prefer this not be the case, this rule of thumb is generally true.

Why should this be the case? One factor is that it is hard to master even one field; to translate from one field to another, you need to have a solid understanding of *both*. Current training programs are not very good at producing computer scientists who understand biology, or vice versa. Thus, most work in biologically-inspired security tends to be done by researchers with inadequate background, resulting in work that either draws little from biology or, more commonly, is poor quality security work.

Ultimately, good security mechanisms should be good security mechanisms, no matter their source of inspiration or design. Work in biologically-inspired security, then, should stand or fall without the crutch of biological rhetoric.

7.3 Conceptual Barriers

While technical and cultural challenges are significant, the biggest barrier to making advances in biologically-inspired security are conceptual. It comes down to this simple problem: how biological do we really want our computers, and if they do acquire biological properties, how will we deal with it?

While living systems may be robust, flexible, and relatively secure, they are also very hard to understand or "debug" and they can be extremely unpredictable at times. Really borrowing from biology may mean creating artifacts that we don't understand and cannot completely predict. Can we accept this?

The robustness and flexibility of biological control patterns arise, in significant part, from intertwined nonlinear feedback loops. By the fundamentally chaotic nature of such systems, they cannot be designed in a conventonal sense; instead, they must be developed through an empirical process, one in which potential solutions are developed, tested, and adjusted.

These are major changes in how computer security, and indeed computer science, is currently practiced. With such difficulties, is it any wonder that when borrowing from biology, computer scientists throw the biology away as soon as it is possible?

7.4 A Necessary Future

While borrowing from biology may seem to lead computer scientists down a dark and windy path, perhaps we have already chosen this path on our own. Modern computer systems consist of millions upon millions of lines of code—much more than can be understood in full detail by any single person. While we still engage in design work, the actual development and deployment of systems is a highly empirical affair: problems regularly arise that were not anticipated and, by design, should never occur—yet they do, and so we find ways to manage such problems on an ad-hoc basis.

Software systems have already become so complex that techniques used to study genetic data have been applied to diagnosing computer problems [33]; as software systems become ever bigger and more complex, we should expect such exchanges to become more common.

Further, the rapidly evolving and changing nature of security threats are making manual, vulnerability-targeted responses less and less tenable. Keeping up with patches and virus signature updates is hard today; what will happen when the rate and severity of attacks increases 10 or even 100-fold?

As the sheer complexity of computers begin to approach that of living systems, and as the threats become more dynamic and numerous, the world of biology will inevitably start looking less unfamiliar. Perhaps only then may we be able to better understand what living systems have to teach us about computer security.

8 Conclusion

Biologically-inspired computer security has given us ways to describe the problems facing computer systems and, in areas such as immune systems, diversity, and homeostasis, it has helped researchers develop interesting and sometimes practical technologies. Nevertheless, there still remains much to learn from biology: we have borrowed relatively few mechanisms and concepts, and work to date has not been sufficient to make computers as robust or self-defending as living systems.

Future work in biologically-inspired security faces significant barriers of a technical, cultural, and conceptual nature. As our systems become ever more complex and as they become more threatened by rapidly adapting adversaries, however, current defensive strategies are proving to be increasingly inadequate. In such circumstances, the concepts of biology and the techiques of biologists may come to play an increasingly important role in computer security.

References

[1] U. Aickelin, P. Bentley, S. Cayzer, J. Kim, and J. McLeod. Danger theory: The link between ais and ids? In *Proceedings of the Second International Conference on Artificial Immune Systems (ICARIS 2003)*, volume 2787 of *LNCS*, pages 147–155, 2003.

[2] J. Balthrop, S. Forrest, and M.R. Glickman. Revisiting LISYS: parameters and normal behavior. In *CEC '02. Proceedings of the 2002 Congress on Evolutionary Computation*, pages 1045–1050, 2002.

[3] Elena Gabriela Barrantes, David H. Ackley, Stephanie Forrest, and Darko Stefanović. Randomized instruction set emulation. *ACM Trans. Inf. Syst. Secur.*, 8(1):3–40, 2005.

[4] Sandeep Bhatkar, Daniel C. DuVarney, and R. Sekar. Address obfuscation: An efficient approach to combat a broad range of memory error exploits. In *Proceedings of the 12th USENIX Security Symposium*, pages 105–120, 2003.

[5] Liming Chen and Algirdas Avizienis. N-version programming: A fault-tolerant approach to reliability of software operation. In *The Twenty-Fifth International Symposium on Fault Tolerant Computing: Highlights from Twenty-Five Years*, pages 113–119. IEEE Computer Society, 1995.

[6] Fred Cohen. *Computer Viruses*. PhD thesis, University of Southern California, 1985.

[7] Dipankar Dasgupta, editor. *Artificial Immune Systems and Their Applications*. Springer-Verlag, Inc., Berlin, 1999.

[8] Fernando Esponda, Elena S. Ackley, Paul Helman, Haixia Jia, , and Stephanie Forrest. Protecting data privacy through hard-to-reverse negative databases. In Springer LNCS, editor, *In proceedings of the 9th Information Security Conference (ISC'06)*, pages 72–84, 2006.

[9] H. Feng, O. Kolesnikov, P. Fogla, W. Lee, and W. Gong. Anomaly detection using call stack information. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.

[10] S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff. A sense of self for Unix processes. In *Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy*. IEEE Press, 1996.

[11] 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*, pages 202–212. IEEE Computer Society Press, 1994.

[12] Stephanie Forrest and David Ackley Anil Somayaji. Building diverse computer systems. In *6th Workshop on Hot Topics in Operating Systems (HotOS-VI)*, 1997.

[13] Daniel Geer et al. Cyberinsecurity: The cost of monopoly how the dominance of microsoft's products poses a risk to security. http://cryptome.org/cyberinsecurity.htm, September 2003.

[14] Julie Greensmith, Uwe Aickelin, and Steve Cayzer. Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. In *Proceedings of the Fourth International Conference on Artificial Immune Systems (ICARIS 2005)*, volume 3627 of *LNCS*, pages 153–167, 2005.

[15] S. Hofmeyr, A. Somayaji, and S. Forrest. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 6:151–180, 1998.

[16] Steven A. Hofmeyr. *An Immunological Model of Distributed Detection and its Application to Network Security*. PhD thesis, University of New Mexico, 1999.

[17] Steven A. Hofmeyr. An interpretative introduction to the immune system. In Lee A. Segel and Irun R. Cohen, editors, *Design Principles for the Immune System and other Distributed Autonomous Systems*. Oxford University Press, 2000.

[18] Gaurav S. Kc, Angelos D. Keromytis, and Vassilis Prevelakis. Countering code-injection attacks with instruction-set randomization. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 272–280, New York, NY, USA, 2003. ACM Press.

[19] Jeffrey O. Kephart. A biologically inspired immune system for computers. In *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 130–139, 1994.

[20] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[21] Gene H. Kim and Eugene H. Spafford. The design and implementation of tripwire: a file system integrity checker. In *CCS '94: Proceedings of the 2nd ACM Conference on Computer and communications security*, pages 18–29. ACM Press, 1994.

[22] Jungwon Kim and Peter J. Bentley. An evaluation of negative selection in an artificial immune system for network intrusion detection. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1330–1337. Morgan Kaufmann, 7-11 2001.

[23] Wenke Lee, Salvatore Stolfo, and Patrick Chan. Learning patterns from unix process execution traces for intrusion detection. In *Proceedings of the AAAI97 workshop on AI methods in Fraud and risk management*, 1997.

[24] PaX Team. PaX address space layout randomization (ASLR). http://pax.grsecurity.net/docs/aslr.txt.

[25] R. Sekar, M. Bendre, P. Bollineni, and D. Dhurjati. A fast automaton-based method for detecting anomalous program behaviors. In *IEEE Symposium on Security and Privacy*, 2001.

[26] Stelios Sidiroglou, Michael E. Locasto, Stephen W. Boyd, and Angelos D. Keromytis. Building a reactive immune system for software services. In *Proceedings of the USENIX 2005 Annual Technical Conference*, 2005.

[27] A. Somayaji, S. Hofmeyr, and S. Forrest. Principles of a computer immune system. In *New Security Paradigms Workshop*, New York, 1998. Association for Computing Machinery.

[28] Anil Somayaji. *Operating System Stability and Security through Process Homeostasis*. PhD thesis, University of New Mexico, 2002.

[29] Eugene H. Spafford. Computer viruses as artificial life. *Journal of Artificial Life*, 1(3):249–265, 1994.

[30] Matthew Stillerman, Carla Marceau, and Maureen Stillman. Intrusion detection for distributed applications. *Communications of the ACM*, 42(7):62–69, July 1999.

[31] Jamie Twycross and Matthew M. Williamson. Implementing and testing a virus throttle. In *Proceedings of the 12th USENIX Security Symposium*, pages 285–294, 2003.

[32] David Wagner and Paolo Soto. Mimicry attacks on host-based intrusion detection systems. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 255–264, New York, NY, USA, 2002. ACM Press.

[33] Yi-Min Wang, Chad Verbowski, John Dunagan, Yu Chen, Helen J. Wang, Chun Yuan, and Zheng Zhang. STRIDER: A black-box, state-based approach to change and configuration management and support. In *Proceedings of the 17th Large Installation Systems Administration Conference (LISA '03)*, pages 159–172, 2003.

[34] Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, 1999.