Immunity-Based Intrusion Detection System:
A General Framework

Dipankar Dasgupta
Division of Computer Science
Mathematical Sciences Department
The University of Memphis
Memphis, TN 38152
Phone (901) 678-4147
Email: dasgupta@msci.memphis.edu

Abstract:

This paper focuses on investigating immunological principles in designing a multi-agent system for intrusion/anomaly detection and response in networked computers. In this approach, the immunity-based agents roam around the machines (nodes or routers), and monitor the situation in the network (i.e. look for changes such as malfunctions, faults, abnormalities, misuse, deviations, intrusions, etc.). These agents can mutually recognize each other's activities and can take appropriate actions according to the underlying security policies. Specifically, their activities are coordinated in a hierarchical fashion while sensing, communicating and generating responses. Such an agent can learn and adapt to its environment dynamically and can detect both known and unknown intrusions. This research is the part of an effort to develop a multi-agent detection system that can simultaneously monitor networked computer's activities at different levels (such as user level, system level, process level and packet level) in order to determine intrusions and anomalies. The proposed intrusion detection system is designed to be flexible, extendible, and adaptable that can perform real-time monitoring in accordance with the needs and preferences of network administrators. This paper provides the conceptual view and a general framework of the proposed system.

1. Inspiration from the nature:

Every organism in nature is constantly threatened by other organisms, and each species has evolved elaborate set of protective measures called, collectively, the *immune system*. The natural immune system is an adaptive learning system that is highly distributive in nature. It employs multi-level defense mechanisms to make rapid, highly specific and often very protective responses against wide variety of pathogenic microorganisms. The immune system is a subject of great research interest because of its powerful information processing capabilities [5,6]. Specifically, its' mechanisms to extract unique signatures from antigens and ability to recognize and classify dangerous antigenic peptides are very important. It also uses memory to remember signature patterns that have been seen previously, and use combinatorics to construct antibody for efficient detection. It is observed that the overall behavior of the system is an emergent property of several local interactions. Moreover, the immune response can be either local or systemic, depending on the route and property of the antigenic challenge [19].

The immune system is consists of different populations of immune cells (mainly B or T cells) which circulate at various primary and secondary lymphoid organs of the body. They are carefully controlled to ensure that appropriate populations of B and T cells (naive, effector, and memory) are recruited into different location [19]. This differential migration of lymphocyte sub-populations at different locations (organs) of the body is called *trafficking* or *homing*. The lymph nodes and organs provide specialized local environment (called *germinal center*) during pathogenic attack in any part of the body. This dynamic mechanism support to create a large number of antigen-specific lymphocytes (as effector and memory cells) for stronger defense through the process of the *clonal expansion* and differentiation. Interestingly, memory cells exhibit selective *homing* to the type of tissue in which they first encountered an antigen. Presumably this ensures that a particular memory cell will return to the location where it is most likely to re-encounter a subsequent antigenic challenge.

The mechanisms of immune responses are self-regulatory in nature. There is no central organ that controls the functions of the immune system. The regulation of the clonal expansion and proliferation of B cells are closely regulated (with a co-stimulation) in order to prevent uncontrolled immune response. This second signal helps to ensure tolerance and judge between dangerous and harmless invaders. So the purpose of this accompanying signal in identifying a non-self is to minimize false alarm and to generate decisive response in case of a real danger[19].

2. Existing works in Intrusion Detection:

The study of security in computer networks is a rapidly growing area of interest because of the proliferation of networks (LANs, WANs etc.), greater deployment of shared computer databases (packages) and the increasing reliance of companies, institutions and individuals on such data. Though there are many levels of access protection to computing and network resources, yet the intruders are finding ways to entry into many sites and systems, and causing major damages. So the task of providing and maintaining proper security in a network system becomes a challenging issue.

Intrusion/Anomaly detection is an important part of computer security. It provides an additional layer of defense against computer misuse (abuse) after physical, authentication and access control. There exist different methods for intrusion detection [7,23,25,29] and the early models include IDES (later versions NIDES and MIDAS), W & S, AudES, NADIR, DIDS, etc. These approaches monitor audit trails generated by systems and user applications and perform various statistical analyses in order to derive regularities in behavior pattern. These works based on the hypothesis that an intruder's behavior will be noticeably different from that of a legitimate user, and security violations can be detected by monitoring these audit trails. Most of these methods, however, used to monitor a single host [13,14], though NADIR and DIDS can collect and aggregate audit data from a number of hosts to detect intrusions. However, in all cases, there is no real analysis of patterns of network activities and they only perform centralized analysis. Recent works include GrIDS[27] which used hierarchical graphs to detect attacks on networked systems. Other approaches used autonomous agent architectures [1,2,26] for distributed intrusion detection.

3. Computer Immune Systems:

The security in the field of computing may be considered as analogous to the immunity in natural systems. In computing, threats and dangers (of compromising privacy, integrity, and availability) may arise because of malfunction of components or intrusive activities (both internal and external).

The idea of using immunological principles in computer security [9-11,15,16,18] started since 1994. Stephanie Forrest and her group at the University of New Mexico have been working on a research project with a long-term goal to build an artificial immune system for computers [9-11,15,16]. This immunity-based system has much more sophisticated notions of identity and protection than those afforded by current operating systems, and it is suppose to provide a general-purpose protection system to augment current computer security systems. The security of computer systems depends on such activities as detecting unauthorized use of computer facilities, maintaining the integrity of data files, and preventing the spread of computer viruses.

The problem of protecting computer systems from harmful viruses is viewed as an instance of the more general problem of distinguishing **self** (legitimate users, uncorrupted data, etc.) from dangerous **other** (unauthorized users, viruses, and other malicious agents). This method (called the *negative-selection* algorithm) is intended to be complementary to the more traditional cryptographic and deterministic approaches to computer security. As an initial step, the *negative-selection* algorithm has been used as a file-authentication method on the problem of computer virus detection [9].

3.1 Virus Detection

In this application, Forrest et al. [9] used the *negative-selection* algorithm to detect changes in the protected data and program files. A number of experiments are performed in a DOS environment with different viruses, including file-infector and boot sector virus samples. Reported results showed that the method could easily detect the modification that occurred in the data files due to virus infection.

This algorithm has several advantages over the existing change (or virus) detection methods: it is probabilistic and tunable (the probability of detection can be traded off against CPU time), it can be distributed (providing high system-wide reliability at low individual cost), and it can detect novel viruses that have not previously been identified. However, since the stored information in a computer system is volatile in nature, the definition of self in computer systems should be more dynamic than in the case of natural immune systems. For example, computer users routinely load in updated software systems, edit files, or run new programs. So this implementation seems to have limited use - only to protect static data files or software.

3.2 UNIX Process Monitoring

As an on-going research on computer security, Forrest et al. [10,11,15] studied the proposed *negative-selection* algorithm to monitor UNIX processes. The purpose is to detect harmful intrusions in a computer system. This implementation aimed at identifying a *sense of self* for UNIX processes, they redefined *self* to accommodate the legitimate activities in dynamic computer environment so that the definition is sensitive to malicious attacks.

This work is based on the assumption that the system calls of root processes [8] are inherently more dangerous to cause damage than user processes. Also root processes have a limited range of behavior, and their behavior is relatively stable over time. The *normal or self* is defined by short-range correlation in a process' system calls. This definition of self seems to be stable during normal behavior for several standard UNIX programs. Further, it is able to detect several common intrusions involving *sendmail*. Their reason of monitoring *sendmail* is that its behavior is sufficiently varied and complex that it provides a good preliminary test, and there are several documented attacks against *sendmail* that can be used for testing. The experiments generated traces of three types of behavior that differ from that of normal *sendmail*: traces of successful *sendmail* attacks, traces of *sendmail* intrusion attempts that failed, and traces of error conditions. They have been able to execute and trace two attacks.

Their preliminary experiment [10] suggests that short sequences of system calls provide a stable signature that can detect some common sources of anomalous behavior in *sendmail*. Because the current measure is easy to compute and is relatively modest in storage requirements, it would be plausible to implement it as an on-line system, in which the kernel checks each system call made by processes running as root. Under this scheme, each site would generate its own normal database, based on the local software/hardware configuration and usage patterns. One advantage of using local usage patterns is that every site would then have its own unique identity, slightly different from everyone else. This would mean that a successful intrusion at one site would not necessarily be successful at all sites running the same software and it would increase the chance of at least one site noticing an attack. This work appears to be very promising and opens new venue in computer security research [16].

3.3 An alternative approach to Virus Detection

Kephart suggested another immunologically inspired approach for virus detection [18]. In this approach, known viruses are detected by their computer-code sequences (signatures) and unknown viruses by their unusual behavior within the computer system.

In this immunity-based method, a diverse suit of *decoy programs* is kept at different strategic areas in memory (e.g. home directory) to capture samples of viruses. Decoys are designed to be as attractive as possible to trap those types of viruses that spread most successfully. Each of the decoy programs is examined from time to time, to see if it has been modified. If one or more have been modified, it is almost certain that an unknown virus is loose in the system, and each of the modified decoys contains a sample of that virus. In particular, the infected decoys are processed by - the *signature extractor* - so as to develop a recognizer for the virus. It also extract information from the infected decoys about how the virus attaches to its host program (attachment pattern of the virus), so that infected hosts can be repaired. The signature extractor must select a virus signature (from among the byte sequence produced by the attachment derivation step) such that it can avoid both false negatives and false positives while in use. In other words, the signature must be found in each instance of the virus, and it must be very unlikely to be found in uninfected programs. Once the best possible signature is selected from candidate signatures of the virus, it run against a half-gigabytes corpus of legitimate programs to make sure that they do not cause false positive. The repair information is checked out by testing on samples of the virus and further by the human expert.

Finally, the signature and the repair program is stored in archive of the AntiVirus database, and the updated (new) version is distributed to the customers. According to Kephart, this approach will also be used to stop the spreading of viruses in networked computers, where infected machines send out **kill signals** to warn other computers of the rampant virus. The signals tell how to kill the new virus as well as similar one.

However, it is not clear how the repair program works in different circumstances. Moreover, decoy programs should have some special characteristics to trap the viruses (no example of such decoy program is given in the paper [18]). Also keeping them in different strategic locations in a computer system is crucial for its success.

4. Proposed Immunity-based IDS:

The normal behavior of a computing system can be characterized by observing its properties over time. The problem of detecting anomalies (or intrusions) can be viewed as finding non permitted deviations of the characteristic properties in the monitored network system. This assumption is based on the fact that intruders' activities in someway must be different from the normal users' activities. However, it may be very difficult to realize or detect such differences in real-time before any damage has been done. The existing immunity-based intrusion detection methods [15,16] emulate one or the other mechanisms of the natural immune system and shown as promising in detecting some type of intrusions.

The important component of the proposed research is to analyze the computational aspects of the immune system and integrate them in a single framework in order to develop a multi-agent intrusion/anomaly detection and response system. Moreover, the detection system monitors several parameters at multiple levels (from packet to user-level) to determine the correlation among the observed parameters during intrusive activities. This immunity-based system has the same three appealing properties as that of other autonomous agent systems [28]: mobility, adaptivity, and collaboration. The immune agents can interact freely in the environment with other agents. Mobility gives agents the ability to move around in the network to explore or monitor the situation. They can mutually recognize each other's activities and can produce a specific response. Such an agent can learn and adapt to its environment dynamically. In collaboration with other agents, they can perform some complex decision-making task [4,31]. The types of agents and the scope of each agent type that we considered are similar in function and purpose as that of immune cells.

Monitoring agents: these agents are vigilant and patrol the network nodes and communicating devices with specific intention (look for malfunctions, faults, anomalies, deviations, etc.). Specifically, these agents monitor various parameters simultaneously at multiple levels. For example, at user level -- search for unusual user behavior pattern; at system level -- look at resource usage such as CPU, memory, I/O use etc.; at process level -- checks for invalid or unauthenticated processes and priority violations; at packet level -- number, volume, and size of packets along with source and type of connections. The role of these agents is very important, there are different sets of such agent roam the network with specific tasks and functions. Some of these agents may work in the complement (non-self) space for monitoring changes (as in the negative selection algorithm [9]), while others have the knowledge of known intrusions.

Communicator agents: they serve as message carriers or negotiators (with limited abilities) in order to maintain a liaison among other agents. They correspond to lymphokines secreted from T cells to stimulate B cells and antibodies in the natural immune system.

Decision/Action agents: they are involved in making decisions (determining the agents that need to be activated) or performing specific tasks according to the underlying security policies. Action agents may activate an appropriate set of response agents (helper agents, killer agents, suppressor agents) depending on the nature and severity of intrusion.

i) Helper agents: activated by action agents through communicator agents. Once activated, they report the status of the environment to the end user or display the decision report. For example, intrusion alert events are reported to the security manager by giving a warning (through email, pager, broadcast message to console, etc.) of possible deviation or violation of norms in the monitored network. Also a GUI based alert interface can be implemented to show the severity of the intrusive activities.

ii) Killer agents: these agents are supposed to take a drastic action in case of real intrusion or malicious activities. For example, at system's level, these agents shut down a machine or disconnect a node; at process level, they kill a process (using kill -9 process id.); at user level, disconnect user session or disable a user account; and at network packet level, the agents may discard a stream of packets if it constitute suspicious transaction. They may be activated (by co-stimulation) through communicator agents by multiple agents in case of a real danger.

iii) Suppressor agents: may be activated by action agents through communicator agents, but they will generally suppress any further action that may be taken by other (decision) agents. They can prevent any action due to false positives in later stages of the intrusion detection/response process.

The proposed intrusion detection can evaluate the current situation, and follow a sequence of actions as a part of the decision making process. In this framework, the activities of different types of agents are coordinated in a hierarchical fashion (as shown in *figure 1*).

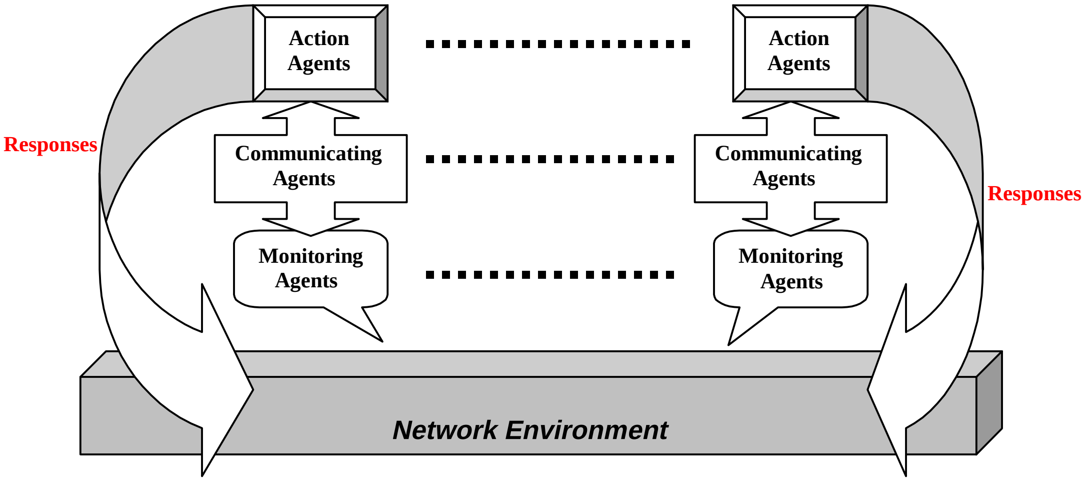

Figure 1. Conceptual view of the proposed multi-agent intrusion detection system.

Roles of each type of agents are unique, though they may work in collaboration. Accordingly, different parts (nodes of the monitored network) of this distributed agent-based detection system can be in one of the following mode of operations, depending on the state of network environment:

Sensing mode: the detection system will monitor information sources from the network environment and is primarily involved in identifying abnormal events at different levels (as mentioned earlier).

Recognition mode: when communicator and action agents are activated in a particular node (or nodes), the detection system is in recognition mode locally and makes an appropriate decision based on the predefined security policies.

Response mode: the detection system takes appropriate action at the infected nodes by activating specific type of agents such as killer agents, suppressor agents, and/or helper agents.

Most of the time monitoring agents navigate the network (check the status of process parameters, usage, connectivity, etc.). If certain situations arise (in any part of the network) and are detected by the monitoring agents, the collaborative agents go to recognition mode, in an attempt to understand the event and take a decision. In some situations, it takes a decision not only by considering the agent's intended role, but also the context of the situation by consulting with other agents in the neighborhood (a second signal for co-stimulation). Once the decision is made, it goes to response mode where specific actions are taken. In principle, different parts of the network can be in different mode depending on the current activities, which may result in asynchronous intrusion detection. Different functional modules of the intrusion detection system are shown in figure 2. In this figure, functionality of inner modules is more distributed compared to outer modules.

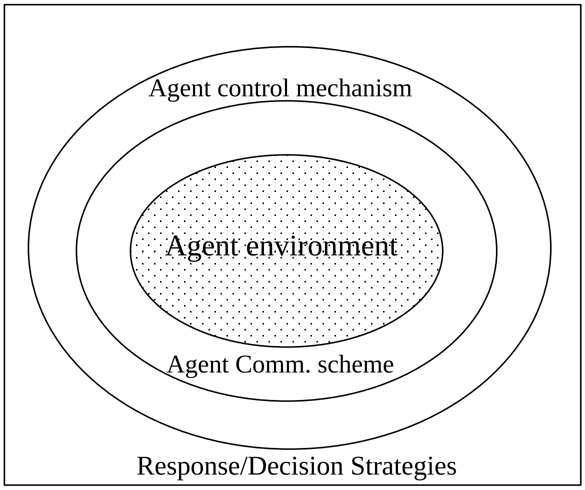

Figure 2. Different functional modules of the detection system.

The software architecture to support the proposed multi-agent system will accommodate necessary agent interaction components and the application environment in an object-oriented platform. It consists of three main modules, apart from design issues of agent's internal structure. These modules are: the agent communication scheme, the agent control mechanisms and the response/decision strategies. The design of heterogeneous agents requires knowledge of the network environment and the notion of neighborhood; in addition, some agents may have a limited life cycle (time dependent) and some of them may work at different time scale. Such a system should be able to operate in a large networked environment, and (detect and) act in response to events in real-time according to its broad decision objectives and security policies.

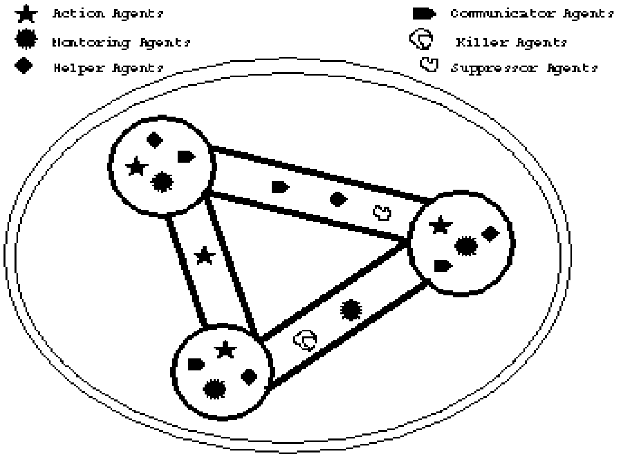

Figure 3. Exhibits a network environment with three nodes (computers) and various immune agents at different site.

5. Implementation Details

A prototype system is currently under implementation on agent software toolkit called **Aglets** [21] with Java visualization tools that works in UNIX network environment (as shown in *figure 3*). We are using some built-in tools (such as *vmstat, iostat, mpstat, netstat, snoop, etc.*), syslog files and shell commands for simultaneously monitoring relevant parameters at multiple levels.

5.1 Multi-level parameter monitoring:

Figure 4 shows different levels of the networked computing activities that the proposed system monitors for intrusion detection. The currently monitored parameters that are listed below, although the list may slightly change as we progress in our implementation.

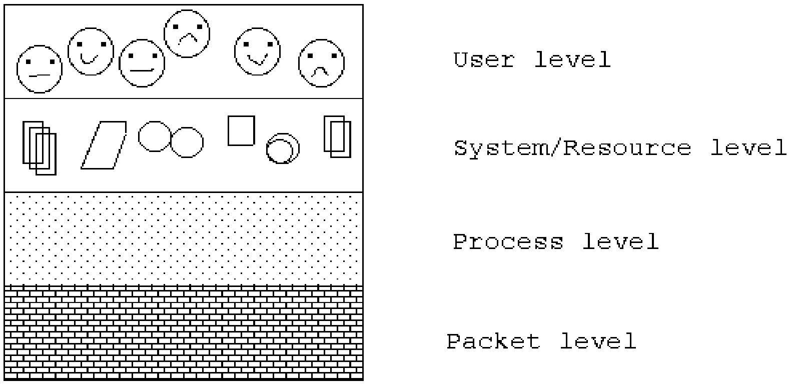

Figure 4. Logical diagram showing different levels of monitored parameters.

To monitor *user-level* activities, the following parameters are recorded as audit trail and analyzed by statistical methods to develop profiles of the normal behavior pattern of users:

- Type of user and user privileges
- Login/Logout period and location
- Access of resources and directories
- Type of software/programs use
- Key stroke pattern (use in future)

The *system-level* parameters that provided indication of resource usage include

- Cumulative and per user CPU usage
- Usage of real and virtual memory
- Amount of swap space currently available
- Amount of free memory
- I/O and disk usage

Various *process-level* parameters monitored to detect intrusion are:

- The number of processes and their types
- Relationship among processes
- Time elapsed since the beginning of the process
- Current state of the process (running, blocked, waiting) and runaway processes
- Percentage of various process times (such as user process time, system process time and idle time).

Some of the parameters that are monitored to gather *packet-level* information:

- Number of connections and connection status (e.g. established, close_wait, time_wait)
- Average number of packets sent and received
- Duration of the connection
- Type of connection (Remote/Local)
- Protocol and port used

Intrusive activities in most cases involve external connection into the target network by an outsider. It is very important to monitor packets sent across the network both inbound and outbound along with packets those are internal to the system. Moreover, the number of external connections established and validity of each connection can be verified using these monitored parameters.

Historical data of relevant parameters are initially collected over a period of time during normal usage (with no intrusive activities) to obtain relatively accurate statistical measure of normal behavior patterns. During monitoring, mobile agents check for any deviation by comparing current parameter values (at different nodes) with the profile of the normal usage (and behavior pattern). The interaction among the agents and with other components of the detection system is coordinated through communicator agents. In our current experimentation, communicator agents are implemented through interfaces. As this monitoring and analysis is carried out in near real-time, this approach appears to be extremely useful in detecting intrusion immediately upon the occurrence and/or providing instant indication of abnormalities on the network.

5.2 Decision Support Component:

Action/Decision agents use various rules for taking decision to initiate a particular action. This rule set may be designed either manually (domain expertise) or evolved (as classifier) using historical data set. In both cases, the objective is to find correlation among the deviated values (from normal) of monitored parameters to determine the type of intrusion and to generate specific action accordingly. A comprehensive set of decision rules is currently under development based on the significance of monitored parameters at the various levels. However, a more effective and practical rule base will only be emerged after the implementation and observation of the network activity over a period of time during the testing phase. The parameters monitored will be fine tuned to yield better decision making and additional parameters that effect the system's performance will also be included. The proposed ID system use machine learning and data mining techniques [22] (in particular, *condition-action* rules can be evolved using genetic algorithms) to learn the normal behavior of the monitored systems, and can automatically adapt to accommodate legitimate changes in the network environment. Based on the designed (or evolved) decision rules the agents may undertake one or more of the following actions:

- Termination of network connection
- Restarting of a particular machine
- Informing the system administrator via e-mail or messaging system
- Block a particular IP address or sender
- Change access privileges of certain user
- Disallow remote connection request
- Change the priority of user processes
- Logout user or close session

However, the system administrator may prefer to take the recommendation (through red alert messages) from the decision/action agents instead of handing over the responsibility to the agents for taking actions. In practice, the action agents may also interact with network management system (NMS) to learn the topology, amplify a response, and/or to effect changes in the network. The scope of this paper is limited to the development of concepts and designing general a general framework along with important functional components of the proposed method. The implementation details and results of the experiments will be reported in subsequent publications.

6. Summary:

The increased network connectivity and easy access to information and resources through Internet and World Wide Web makes the security issues one of the most important factors in today's computing. The promise of Electronic Commerce also contributing to the explosive growth of the Internet and the underlying communication networks. Though there are many security-related products and technologies, yet the potential threats and vulnerabilities are intractable.

Intrusion detection is an important part of computer security. It provides an additional layer of defense against computer misuse (abuse) after physical, authentication and access control. Different models of intrusion detection have been developed, and many IDS software available for use [20]. Commercial IDS products such as NetRanger (www.cisco.com), RealSecure (www.iss.net), Omniguard Intruder Alert (www.axent.com) work on attack signatures. These signatures needed to be updated by the vendors on a regular basis in order to protect from new types of attacks. There is a working group established to design a common intrusion detection framework (CIDF)[17] for providing a common intrusion specification language. However, no detection system can catch all types of intrusions and each model has its strength and weaknesses in detecting different violations in networked computer systems. An influx of new approaches is needed to enhance security measures. Researchers have been exploring various artificial intelligence based approaches for intrusion/misuse detection [3,12,22,24,30]. Recent works on immune-based computer security [9,10,15,16,18] emulated one or the other functional components of the natural immune system. In particular, Forrest et al. [9] used a negative-selection algorithm to detect changes in the protected data and program files. In another work, they [10,11,16] applied the algorithm to monitor UNIX processes where the purpose is to detect harmful intrusions in a computer system. Kephart suggested another immunologically inspired approach for virus detection [18]. In this approach, known viruses are detected by their computer-code sequences (signatures) and unknown viruses by their unusual behavior within the computer system.

The proposed system attempts to integrate several potentially useful immunological properties in a single framework in order to develop a robust and intelligent detection system. This system will provide the user better monitoring the network environment and give an additional tool to making the computing systems secure. As intruders finding new ways to break in, security systems should be more flexible and intelligent enough to withstand both known and unknown intrusions.

The proposed system has some unique features compared to the existing agent-based detection systems [1,2]. They include simultaneous multi-level monitoring, detection of known and unknown intrusions, and hierarchical *sense and response* mechanisms. Moreover, immunity-based mobile agent's role, adaptivity, self-regulation, life cycle, specificity, and diversity will definitely contribute a new dimension to the existing agent-based intrusion detection systems. The developed system will perform real-time monitoring, analyzing, and generating appropriate response to intrusive activities. It is designed to be flexible and extendible to meet the specific security needs and preferences of an organization.

Though the development of this immunity-based detection system is an ongoing effort, and will take longer to complete, however, once ready it will be useful to any security conscious organization with sensitive data and software. As mentioned before, this multi-agent system can simultaneously monitor network activities at different levels (such as packet level, process level system level and user level), it can detect both inside misuse and outside attacks. Since the real intrusive activities usually affects a number of parameters in networked computers, monitoring them simultaneously (at a multiple time scale) at different levels should minimize false positive/negative in detection. The intent of the system, however, is to provide the least amount of impact to network performance by running the detection agents as background processes. Moreover, as the agents run as privileged processes, tempering of agents will be very difficult, if not impossible. Since the detection system is fully distributed, and the agents are generated and distributed all across the network, attack at one particular node will not compromise the detection ability at other nodes. Future work will address the issues of protecting the immune agents from corruption by malicious activities.

Acknowledgements:

This work is partially supported by a joint NSF research instrumentation grant (#98-18323) and the University of Memphis New Faculty Research Grant (FY98). The author would like to thank Tom Barton and the reviewers of this paper for useful comments and suggestion.

References:

1. J. S. Balasubramaniyan, J. O. G. Fernandez, D. Isacoff, E. Spafford, and D. Zamboni. An Architecture for Intrusion Detection using Autonomous Agents, COAST Technical report 98/5, Purdue University, 1998.
2. Mark Crosbie and Eugene Spafford. Defending a computer system using autonomous agents. In *Proceedings of the 18th National Information Systems Security Conference*, October 1995.
3. Mark Crosbie and Gene Spafford. Applying Genetic Programming to Intrusion Detection. COAST Laboratory, Purdue University, 1997 (also published in the proceeding of the Genetic Programming Conference).
4. Dipankar Dasgupta. An Artificial Immune System as a Multi-agent Decision Support System. In proceedings of the *IEEE International Conference on Systems, Man, and Cybernetics*, San Diego, pages 3816--3820, 1998.
5. Dipankar Dasgupta and Nii Attoh-Okine. Immunity-based systems: A survey. In proceedings of the *IEEE International Conference on Systems, Man, and Cybernetics*, pages 369--374, Orlando, Florida, October 12-15, 1997.
6. Dipankar Dasgupta (Editor). *Artificial Immune Systems and Their Applications, ISBN 3-540-64390-7, Springer-Verlag, 1999.*
7. Dorothy E. Denning. An Intrusion-Detection model. In *IEEE Symposium on Security and Privacy*, pages 118--131, 1986.
8. C. Ko, G. Fink, and K. Levitt. Automated Detection of Vulnerabilities in Privileged Programs by Execution Monitoring. In *Proceedings of the Computer Security Applications Conference*, 1994.
9. S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. Self-Nonself Discrimination in a Computer. In Proceedings of *IEEE Symposium on Research in Security and Privacy*, pages 202--212, Oakland, May 16-18 1994.
10. S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for unix processes. In Proceedings of *IEEE Symposium on Research in Security and Privacy*, Oakland, CA, 1996.
11. S. Forrest, S. Hofmeyr, and A. Somayaji. Computer Immunology. In *Communications of the ACM*, Vol. 40, No. 10, pages 88-96, 1997.
12. J. Frank. Artificial Intelligence and Intrusion Detection: Current and future directions. In Proceedings of the *17th National Computer Security Conference*, October 1994.
13. L. Todd Heberline, Gihan V. Dias, Karl N. Levitt, Biswanath Mukherjee, Jeff Wood, and David Wolber. A Network Security Monitor. In *IEEE Symposium on Research in Security and Privacy*, Pages 296--304, IEEE, May 1990.
14. L. T. Heberlein, K. N. Levitt and B. Mukherjee. A method to detect intrusive activity in a networked environment. In Proceedings of the 14th National Computer Security Conference, pages 362-371, 1991.
15. S. A. Hofmeyr, S. Forrest and A. Somayaji. Intrusion Detection using Sequences of System Calls. In Journal of Computer Security (*in press*).
16. S. A. Hofmeyr and S. Forrest. Immunizing Computer Networks: Getting All the Machines in your Network to Fight the Hacker Disease. In IEEE Symposium on Security & Privacy, 1999.

17. C. Kahn, P. A. Porras, S. S. Chen, and B. Tung. A Common Intrusion Detection Framework. *Draft submission to a nice publication*, 1998.
18. Jeffrey O. Kephart. A biologically inspired immune system for computer. In Proceedings of *Artificial Life*, Cambridge, M.A., July 6-8 1994.
19. Janis Kuby. *Immunology* (2nd Ed.), W. H. Freeman and Company, 1994.
20. S. Kumar and E. H. Spafford. A software architecture to support misuse detection. In Proceedings of *the 18th National Information Security Conference*, pages 194--204, 1995.
21. Danny Lange and Mitsuru Oshima. Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley, August 1998.
22. Wenke Lee and Salvatore J. Stolfo. Data Mining Approach for Intrusion Detection. In Proceedings of the *7th USENIX Security Symposium*, 1998.
23. Teresa F. Lunt. IDES: An Intelligent System for Detecting Intruders. In Proceedings of the symposium*: Computer Security, Threat and Countermeasures*, November 1990.
24. Jake Ryan, Meng-Jang Lin and Risto Miikkulainen. Intrusion Detection with Neural Networks. In *Advances in Neural Information Processing Systems* 10, MIT Press, 1998.
25. 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 the proceedings of the *14th National Computer Security Conference*, pages 167--176, October 1991.
26. E. Spafford, D. Zamboni. A framework and prototype for a distributed intrusion detection system. COAST Technical report 98/6, Purdue University, 1998.
27. S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, C. Wee, R. Yip and D. Zerkle. Computer Security Research Group. The Design of GrIDS: A Graph-Based Intrusion Detection System. *Technical report*, UC Davis, Dept. of Computer Sc., May 14, 1997.
28. M. Tambe. Implementing Agent Teams in Dynamic Multi-agent Environments. In *Applied Artificial Intelligence*, Volume 12, 1998.
29. H. S. Vaccaro and G. E. Liepins. Detection of anomalous computer session activity. In the Proceedings of the *IEEE Symposium on Security and Privacy*, pages 280--289. IEEE, 1989.
30. Gary M. Weiss, Johannes P. Ros and Anoop Singhal. ANSWER: Network Monitoring Using Object-Oriented Rules. In Proceedings of *American Association for Artificial Intelligence (AAAI)*, pages 1087-1093, 1998.
31. Gregory B. White, Eric A. Fisch and Udo W. Pooch. Cooperating security managers: A peer-based intrusion detection system. IEEE Network, pages 20-23, January/February 1996.