Evolutionary Algorithm Approaches for Detecting Computer Network Intrusion (Extended Abstract)

Kevin P. Anchor, Paul D. Williams, Gregg H. Gunsch, and Gary B. Lamont

Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology, Wright-Patterson AFB, (Dayton)OH 45433

1 Introduction

Attacks against computer networks are becoming more sophisticated, with adversaries using new attacks or modifying existing attacks. With increased global interconnectivity, reliance on e-commerce, network services, and Internet communication, computer security has become a necessity. Organizations must protect their systems from intrusion and computer-virus attacks. Such protection must detect anomalous patterns by exploiting known signatures while monitoring normal computer programs and network usage for abnormalities. Current antivirus and network intrusion detection solutions can become overwhelmed by the burden of capturing and classifying new viral stains and intrusion patterns. To overcome this problem, a self-adaptive distributed agent-based defense immune system based on biological strategies is developed within a hierarchical layered architecture using genetic algorithms. A prototype interactive system is designed, implemented in Java, and tested. The results validate the use of a distributed-agent biological- system approach toward the computer-security problems of virus elimination and intrusion detection. Also, a intrusion detection variation using evolutionary programming is introduced. This generic research is sponsored by the Defensive Information Warfare Branch, Information Directorate, Air Force Research Laboratory, Rome, NY.

In its purest form, intrusion detection (ID) is the process of identifying the presence of unauthorized access to an enterprises computing resources. In practice, ID is broader and includes the detection of: 1) misuse/abuse; unauthorized activities by authorized users (e.g., accessing pornography, theft of information, using corporate resources for personal gain); 2) reconnaissance; determination of systems and services that may be exploitable; 3) penetration; attempt of unauthorized activity to gain access to computing resources; 4) penetration; successful access to computing resources by unauthorized users; 5) trojanization; presence and activity of unauthorized processes; 6) denial of service; an attack that obstructs legitimate access to computing resources.

2 Computer Defense Immune System (CDIS)

The major objective of our prototype system is to detect the existence of nonself patterns within a potentially larger set of existing self patterns. This is modelled as a formal set

theory notation. This detection methodology can generate two types of errors: Type I, or false-positive errors, and Type II, or false-negative errors which need to be continually evaluated.

The antibodies for detecting file infections are simple byte strings. These patterns are compared to the bytes within the computer file system. Antibodies for network intrusion are longer and segregated because they utilize the IP packet structures as a template. There are many types of protocols flowing on our networks. For the purposes of this system, only the three most common protocols are monitored: TCP, UDP, and ICMP. All three of these protocols are layered on top of the IP. The generated antibodies, are deployed and compared to possible malicious attacks by a matching-rule function . It is this function that provides the core functionality of the detection process.

The many pattern-matching functions come in two varieties: distance measures, which express how different two sequences are, and similarity functions, which measure how alike they are. Intuitively, objects that are close together in the feature space must be similar, while those that are farther apart are dissimilar. A geometric modelling of this space defines self and nonself regions that are represented as hyper-rectangles. Those that are similar to a nonself pattern within a certain threshold can be classified as nonself. The matching rules investigated in this study utilize statistical, physical, and binary measures of distance or similarity. One statistically based similarity measure is the correlation factor or correlation coefficient.

The overall goal is to create an agent-based CDIS. This was accomplished successfully and two layers of defense have been implemented. Effective system and local models of immune system operation were constructed that realize improvements over current virus antibodies and packet-based ID solutions. Based on these models, the multilayered implementation provides an effective solution for the detection, identification, and elimination of computer viruses and network attacks. The prototype was used to gain insight into the efficiency and effectiveness of an agent-based AIS. The successful use of agents and the integration of pattern recognition principles are valuable contributions to the immunological computation community. This research was conducted by integrating many different domains including immunology, immunological computation, malicious code, multiagent systems, and parallel and distributed computation.

The system design integrates the power, flexibility, adaption, and capabilities of the BIS into an architecture realizable in the information system domain. Based on the models, the prototype implementation provides an effective solution for the detection, identification, and elimination of malicious code and bad packets. The level of effectiveness is tunable through the proper selection of the number of antibodies, the antibody length, and the detection threshold. These must be selected based on the contents of known self and with an understanding of their ramifications on negative-selection time, scan time, and nonself space coverage. The use of the agent paradigm facilitates the construction of an AIS because of the performance limitations of a monolithic implementation and the biological basis for the architecture can be viewed as a system of collaborating agents. While using agents improves the understanding of the system design and the mapping to the biological domain, the deployment of the agents must be done by considering the principles of parallel software design in order to improve performance. For an agent-based Similarly, the addition of a new application may require

recensoring of the antibodies. Care must be taken to ensure the new software is not already infected. An alternative is to scan with the current antibodies: a positive detection could indicate the presence of a virus or recognition that this self has not been encountered before by the system. The decision on whether this is a false-positive error rests with the system administrator, and is accommodated by the system through the costimulation function of the antibody lifecycle. Additional required features include the assurance of detection and elimination, features essential to system effectiveness.

Developing a CDIS involves reducing communication and placing detection agents near their I/O sources. This CDIS design is scaleable in terms of scope and coverage through the simple addition of new agent types and participating system nodes. The prototype implements file system and IP packet detection, but a more complete multilayered defense could be realized by adding agent types for monitoring memory, email, boot sectors, complex intrusions, and more. Additionally, because the JSDT provides lookup services, agents can join or leave the system at anytime. At its current level of maturity, the prototype does not provide for a practical implementation nor unobtrusive operation. The Java implementation provides a good prototype environment, but its speed limits the system usability. The negative-selection and scanning times measured in years are unacceptable for a practical system. An implementation improvement to increase the system speed is paramount to future system viability. The agent-based CDIS offers detection and management capabilities that are absent from current deployed solutions. The abilities of these facets working together promises an enterprise-wide computer-security solution. At the heart of CDIS is the ability to proactively generate antibodies capable of detecting nonself data; the research presented herein investigates a method of generating antibodies for the computer-virus and network intrusion problem domains. The preliminary results, though limited, indicate that this approach holds promise and deserves continuing investigation.

3 Evolutionary Programming and MOEAs

This evolutionary programming research uses two types of multiobjective approaches, lexicographic and Pareto-based, in a multiobjective evolutionary programming algorithm to develop a new method for detecting such attacks. The approach evolves finite state transducers to detect attacks; this approach may allow the system to detect attacks with features similar to known attacks. Initial testing shows the algorithm performs satisfactorily in generating finite state transducers cable of detecting simulated attacks. 1 Introduction Attacks, or intrusions, against computer systems and networks have become commonplace events. Many intrusion detection systems and other tools are available to help counter the threat of these attacks; however, none of these tools is perfect, and attackers are continually trying to evade detection. This paper presents research into detecting attacks using an evolutionary algorithm, specifically evolutionary programming. The algorithm uses multiple objectives to learn to recognize computer attacks, thereby treating the problem of intrusion detection as a multiobjective search problem.

As indicted, an intrusion detection system (IDS) helps detect and identify an attack, which is defined as any inappropriate, incorrect, or anomalous activity, on a computer

system or network. The EP research system is a hybrid of two forms, as it uses knowledge about an attack and information based on a partial model of known good network traffic, or self, to evolve finite state transducers (FSTs) that can detect the attack and other similar or related attacks.

The goal of this investigation is to develop an innovative method for detecting new or stealthy attacks on the network. One type of stealthy attack, called a low and slow attack, is a probe or intrusion attempt that is stealthy in that it takes place over a long period of time, covers a large number of targets, or originates from a number of different, coordinated sources. Because these attacks are designed to be stealthy, they are hard to detect using current intrusion detection systems. Current IDSs can be tuned to detect some stealthy attacks, but the resulting false alarm, or false detection, rate usually increases to an unacceptable level. Thus, new methods for detecting these types of attacks are needed. New attacks may be modifications of existing attacks, so an approach for an ID system is to use knowledge of existing attacks to develop generalized detectors. These generalized detectors might have the ability to detect unknown attacks that are based on existing attacks or that are similar to existing attacks. Developing such generalized detectors is one aspect of the Intrusion Detection (ID) problem. This approach appears to map to the Time Series Prediction problem, in which a sequence of symbols is input and the correct output symbol must be predicted based on the input symbols. In this mapping, the input symbols are a sequence of network packets, and the output symbols represent whether the sequence is assumed to be an attack or not.

Although the packet content or payload is an important part of each packet, it is not used in this research for two reasons. The main reason is that the size of the search space increases immensely if this field is used; the second reason is that existing signature-based detectors can be used to examine the content field in an efficient manner. Network traffic consists of a sequence of packets, and an attack is also a sequence of packets. The packet features and relationships between features of multiple packets can be used to determine if a particular sequence of packets is an attack or not. The previous discussion motivates a new method for detecting attacks. This method is to use a finite state transducer (FST) to examine the relationships between packets coming across the network to determine if any particular sequence contains an attack. This type of detection provides the ability to define patterns of known attacks and variations or modifications of known attacks. This method might also detect new attacks that have similar packet relationships as do existing attacks. In addition, this method allows for distinguishing between attack sequences and non-attack sequences because the FST can be built to accept an attack sequence while rejecting a non-attack sequence. The genotype, or internal representation, of a detector in this scheme is an FST, which represents some regular language or pattern. The phenotype, or outward expression, of the detector is a Detect or Not Detect signal, which corresponds to the FST rejecting or accepting the word, which represents the network packets that may constitute an attack. The fitness value of a particular FST is dependent on two factors: whether it detects an attack correctly and whether it does not detect a non-attack string as an attack. Because there are multiple factors involved, a multiobjective approach to solving this problem seems a natural fit.

The original EP algorithm for this problem used only the single objective for detecting a given attack string, while the second version added the ability to use a single

self-string along with the attack string in a lexicographic multiobjective fashion. The current implementation provides the ability to use an arbitrary number of self-strings along with the attack string in either a lexicographic or pareto dominance multiobjective optimization.

The testing performed on the algorithm is designed to determine whether the two multiobjective EP algorithms are capable of finding solutions, or good detectors, over a range of input attack strings. In this case, a good detector is one which detects the input string developed from a sequence of attack packets and generates a Detect signal. Thus, the outputs of the algorithms are the time required to find a good solution and the number of generations needed to find that solution. The algorithms are tested using five representative input strings, or benchmark attack packets. The false alarm rate plays a key factor in determining the usefulness of an intrusion detection system, so it is important to test. Our conjecture is that the quicker execution time of the appropriate tests leads to a larger false alarm rate, or lower solution quality; solution quality.

This research presents an initial step in detecting computer network intrusions through the use of two different multiobjective evolutionary programming algorithms. The testing shows that the evolutionary programming technique generates finite state transducers, or Mealy-type finite state machines, capable of matching or detecting an input attack string, for the simulated attacks tested. The lexicographic approach allows the use of the attack and self strings while performing significantly faster than the pareto dominance approach; however, further testing is required.

4 Conclusions

The use of EPs has shown to beneficial in finding network intrusions across a limited set of benchmarks. Also, the continuing development of an network intrusion software system (CDIS) has performed well over a series of tests. Future work includes more elaborate testing, analysis and evaluation. It should be mention that we are also studying the use of wavelet-based steganalysis using a computational immune system approach. And, we are using an artificial immune system techniques for identifying chemical spectra.