CSC 540 Graduate Research Seminar

Usage of biologically inspired methods in computer security
by XXXXXXXX

Introduction

Researchers have been taking inspiration from biology and applying it toward the goal of securing our computing systems. Techniques being used to achieve this goal include positive and negative selection (modeled on the immune system found in mammals) and genetic algorithms (based on the process of evolution.)

Using these ideas, researchers hope to develop systems capable not only of protecting our systems, but actively searching for, identifying, and stopping new viruses and exploits before they have a chance to infect and attack our systems.

Important Ideas

In an attempt to mimic the immune system, researchers have used the process of negative selection to build detectors to be used in a classification task. Negative selection is a process by which the body filters out self-reactive antibodies, that is, antibodies which would react to the host (self) rather than a foreign body (non-self). Negative selection has some attractive properties when applied to anomaly detection, specifically, it provides a means of generating detectors which do not match known good patterns, thus the patterns they do match can be considered anomalous.

Stibor et al. [1] examine the topic of negative selection in regard to its application to anomaly detection. After examining several examples of previous research on negative selection, Stibor suggests that negative selection alone is insufficient for anomaly detection and proposes a different method based on the process of positive selection in which elements belonging to the self class are placed in a hypercube and given a detector radius. Any subsequent elements which fall within the radius are determined to be self elements and are not considered to be anomalous.

One of Stibor’s main points as to why this self-detector method performs better than their non-self-detector counterparts is that attempts to fill a non-self space with detectors while ensuring that no members of the self class are covered results in incomplete coverage of the space. Stibor suggests that by considering the entire detector space to be non-self then only detecting for self elements the entire detector space can be covered. Experimental results comparing the two methods generally favor Stibor’s self-detector method.

The ability for a classifier to perform well using examples of only one class is quite interesting given that information which helps define the boundaries between two classes is not available. It may be the case that these types of classifiers only work well when members of the class are clustered to only a few areas.

Researchers Xiaoshu Hang and Honghua Dai [2] make use of both negative and positive selection in their work. When working with classifiers in anomaly detection, a common problem is the underrepresentation of the anomalous class. Since anomalous events are less frequent, there are typically fewer examples with which to train the classifiers. If the unbalanced datasets are left uncorrected, classifiers will often overfit to the normal class resulting in poor performance on unseen data. Hang and Dai begin by first learning the patterns of the normal class using a genetic algorithm seeking to maximize a fitness function which measures how many patterns of the normal class a candidate pattern matches. If a pattern matches any anomalous samples, it is removed from the population.

Once the patterns have been found they are used to generate additional anomalous samples by using a somewhat straightforward process. If there are examples of the anomalous class, they are used as "seed examples." Each sample, s, is represented as a hypersphere into an n-dimensional space, each s has at most 2n neighbors. For a given s, each neighbor location is checked to determine if it is vacant, that is, not occupied by either a normal or anomalous sample. If it is vacant, it will be matched against the patterns of the normal class. If the potential location does not match, it will be added to the set of synthetic anomalous samples. This process will be repeated until enough synthetic examples have been generated to balance the dataset. In the event there are no anomalous samples to serve as seeds, the algorithm will run in much the same way, however, random points will be used as seeds rather than samples. After balancing the datasets with the synthetically generated data, standard classifiers such as C4.5 decision trees and Naive Bayes are trained.

To test their algorithms, the authors ran 3 trials on 14 different data sets. The first used the data in its natural distribution with no balancing, the second trial used data that had been balanced with seed examples included, and the third had all anomalous samples removed prior to balancing. Classifier performance was best when trained on data which included both normal and anomalous samples, however, in the trials where all anomalous samples were removed, the classifiers were able to successfully identify over 70% (and frequently more) of the anomalous samples.

An interesting observation about this work is the connection that can be drawn to Stibor et al.'s research. Their method uses the area within a defined radius from the normal samples to represent the normal class whereas Hang and Dai use genetic algorithms to learn the patterns of the normal class. Both then treat anything not covered by their definition of the normal class as anomalous. An idea for future investigation may be to determine if there is a connection between the radius in the work of Stibor et al. and how well samples match the patterns found by Hang and Dai's genetic algorithm optimizing their fitness function. One of the difficulties of Stibor's research was in determining the proper radius for the detectors, exploring this possible connection may lead to a method for finding an appropriate radius. In addition, while Hang and Dai's method shows positive results, the inclusion of trials which used traditional dataset balancing techniques such as oversampling the anomalous class, would have provided a another data point for determining the efficacy of their method.

Similar to Hang and Dai's work using genetic algorithms to generate additional anomalous samples, researchers Kayacik, Heywood, and Zincir-Heywood [3] applied genetic programming to enable them to generate examples of a particular class of exploits. In their paper, which focuses on generating buffer overflow exploits, they develop a fitness function by which potential exploits can be measured and evolved. A brief overview is as follows: generate a random set of instructions, execute the instructions in a simulated runtime environment, when the system call which actually initiates the exploit is executed halt execution and inspect the stack and program registers. If any requisite pieces are missing -- for example if the EDX register does not contain NULL -- deduct points from the fitness of the individual. Select the top performing members from a generation, do crossover and mutations, and run additional generations until suitable exploits have been generated. In initial trials, only the minimum set of instructions necessary were supplied for the algorithm to select from, however, in later tests mathematical and logical operators were also included.

Results showed the ability to generate successful exploits which somewhat obfuscated the function of the code. As additional operators were made available for use, the exploits became increasingly different from the core exploit from which the fitness function was developed. The authors attribute this, in part, to the mathematical and logical operators providing instructions which serve as introns (sections of the code which are functionally irrelevant and could be removed with no change in the final outcome.) Their software was also able to find different methods for achieving the goals of the fitness function. For example, where the core attack used the instruction XOR EAX, EAX a generated exploit may use SUB EAX,EAX both resulting in a value of zero being stored in the EAX register.

Using this method, the authors generated over 2000 successful exploits. When transmitting these exploits over a network being monitored by the intrusion detection software, Snort, all were able to evade detection.

The process Kayacik, et al. implemented is quite intriguing, it would be interesting to see how well it performs on other, more sophisticated types of exploits. If it were able to generate thousands of examples of various classes of exploits, these examples could potentially be used to train machine learning classifiers for the purposes of intrusion and virus detection.

A system along the same line of thought was developed by Edge et al. [4]. REALGO (REtrovirus ALGOrithm) is a system heavily influenced by biological immune systems. This algorithm expands upon previous work in the area using artificial immune systems (AIS) for virus detection by incorporating a memory construct comparable to the biological function of RNA which allows a genetic algorithm to revert back to a known good solution if the current solution fails to produce a better result. From this point, the search can continue in another direction. The authors suggest this saves the overhead of requiring a new solution to go through the initial rounds of evolution, it can simply pick up where the previous solution "went wrong."

The algorithm begins by using a genetic algorithm to generate detectors for the patterns of previously known virus definitions. Detectors may also be mutated in an effort to identify novel viruses. These detectors then go through a negative selection process to remove any which may incorrectly identify good processes and programs as a virus, finally the detectors are deployed to the monitoring program. In an attempt to keep the number of detectors small, if a detector has not been triggered in a specified period of time, it is subject to removal and replacement. This also serves to keep the detectors relevant to exploits in current use. One criticism of this, however, is the possibility for old viruses to pass undetected if their detector is no longer being used. It may be beneficial to run such a system alongside a traditional pattern matching virus detection system. REALGO could potentially identify new viruses and pass their definitions on to the pattern matching scanner which could protect against a larger number of exploits.

Concluding Remarks

As previously shown, these biologically inspired methods are showing promising results in the field of computer security. A common attribute which makes these systems so appealing is the proactive nature of their protection. Rather than a reactive approach of antivirus systems which require manual identification and analyzation prior to providing the user protection.

References

[1] Thomas Stibor, Philipp Mohr, and Jonathan Timmis; “Negative Selection Appropriate for Anomaly Detection” in *GECCO’05*, 2005, pp. 321-328

[2] Xiaoshu Hang and Honghua Dai; “Applying both Positive and Negative Selection to Supervised Learning for Anomaly Detection” in *GECCO’05*, 2005, pp. 345-352

[3] Hilmi G. Kayacik, Malcom Heywood, Nur Zincir-Heywood; “On Evolving Buffer Overflow Attacks Using Genetic Programming”, *GECCO’06,* 2006, pp. 1667-1673

[4] Kenneth S. Edge, Gary B. Lamont, and Richard A. Raines; “A Retrovirus Inspired Algorithm for Virus Detection and Optimization”, *GECCO’06,* 2006, pp. 103-110