Evolving discrete-valued anomaly detectors for a network intrusion detection system using negative selection

Simon T. Powers
School of Computer Science
University of Birmingham
Birmingham, B15 2TT
UK
simonpowers@blueyonder.co.uk

Jun He
School of Computer Science
University of Birmingham
Birmingham, B15 2TT
UK
J.He@cs.bham.ac.uk

Abstract

Network intrusion detection is the problem of detecting unauthorised use of, or access to, computer systems over a network. One approach is anomaly detection, where deviations from a model of normal network activity are reported. The negative selection algorithm, inspired by the immune system, can be used to generate anomaly detectors. Previous work has applied a genetic algorithm to real-valued detectors. However, we argue that at least some discrete fields are required in detectors, e.g. the port number. The system reported in this paper evolves discrete-valued detectors, which we show are able to outperform real-valued detectors.

1 Introduction

Network intrusion detection is the problem of detecting unauthorised use of, or access to, computer systems over a network. Two broad approaches exist to solving this problem; anomaly detection and misuse detection [13]. Misuse detection systems store a database of attack signatures, and then recognise attacks in the database through pattern matching. This technique has a low false alarm rate, but is incapable of detecting anything not contained in the database and so will miss novel attacks. An alternative approach is anomaly detection, where deviations from a model of normal network activity are monitored, rather than the signatures of known attacks. A system using anomaly detection is therefore able to detect novel attacks, providing that the attack is sufficiently different from normal activity as defined by the intrusion detection system (IDS). However, such a system is also prone to a higher false alarm rate.

The field of natural computation takes inspiration from natural systems and applies it to computational problems. In recent years, the field of artificial immune systems has begun to flourish. An artificial immune system uses ideas from the operation of the human immune system. In the case of intrusion detection, the immune system can be viewed as performing anomaly detection as it distinguishes between normal self and harmful non-self in the body. In the body, self is the normal cells and non-self is invading pathogens. In an IDS, self is normal network activity and non-self is an attack. In the body, one type of anomaly detector is a certain type of lymphocyte known as a T-cell. T-cells are capable of binding to non-self (antigens) but not to self. This is ensured during a maturation process, whereby any T-cells that bind to self are destroyed.

The negative selection algorithm [6] copies this approach, by randomly generating anomaly detectors and then discarding any that match self. However, while this approach has worked successfully with a binary string representation, it has been argued that such a low-level representation is inappropriate for an IDS [7]. This is because it is difficult to analyse the operation of a binary string detector. A more intuitive approach is to use a higher level representation in the form of IF-THEN rules. González has proposed such a system using a real-valued representation [7] and a genetic algorithm (GA) for detector generation. However, his system only considered 3 statistics of network activity and so is extremely limited as an IDS. Furthermore, it is not possible to incorporate important features of TCP connections such as the port number into his detectors. This is because such features are discrete, not real-valued.

In this work, we propose a discrete-valued representation for detectors, along with a GA for detector generation. Specifically, we build an IDS that monitors important features of individual TCP connections. Our system is novel in the sense that a GA incorporating negative selection has not been used to construct discrete-valued anomaly detectors before. We also consider different statistics of network activity to those in other works. Experimental results comparing the performance of our system to that of González are presented in this paper, where we show that for the same false positive rate, our system is able to achieve a greater attack detection rate.

The remainder of this paper is organised as follows. Section 2 provides a review of previous work applying negative selection to network intrusion detection. Section 3 presents our discrete-valued representation for detectors, and describes the GA used to generate them. Section 4 presents empirical results analysing the attack detection and false positive rate trade-off of our system, along with a comparison of our system to that of González. Finally, section 5 makes some concluding remarks along with suggestions for further improvement.

2 Negative selection approaches to network intrusion detection

This section provides a review of existing works that apply negative selection to the problem of network intrusion detection.

2.1 Negative selection on binary strings

The research group of Stephanie Forrest [6] applied negative selection to binary strings. They represent both detectors and antigens as strings of the same length, and declare a measurement to be anomalous if the corresponding antigen binary string is matched by a detector binary string. There are many ways in which this matching can be defined, however, they chose to use the simple r-contiguous bits rule. This rule states that two strings match if they share the same values in an uninterrupted stretch of r bits. One detector can therefore match many similar antigen strings.

Amongst other applications, e.g. [5], Forrest's group have applied this technique to the problem of network intrusion detection. Their LISYS system [8], [2] encodes the source IP address, destination IP address and server-side port of TCP connections in a 49-bit binary string. They obtain a set of self strings by observing normal TCP connections over a period of time, and then use negative selection to generate detector strings that aim to match anomalous connections that may occur in the future.

Unfortunately, there are two problems with such an approach. Firstly, the use of binary strings and an r-contiguous bits matching rule makes it difficult to extract high-level domain knowledge from the detectors [7]. For example, in an intrusion detection system, we would like to be able to analyse the detectors that were activated during an attack, in order to discover the properties of the attack. However, analysing the part of a binary string that matched part of another such string is unlikely to yield much useful domain knowledge. This is because both the representation and the matching rule are too low-level to facilitate such a process.

The second problem with LISYS is one of applicability to a real-world scenario. It is certainly the case that simply looking at the IP addresses and ports of a connection is insufficient to detect many types of attacks. However, adding further information about the connection to the detector and antigen strings would rapidly increase their length, given that binary coding is used. Furthermore, as the detectors come to store more information, it becomes questionable whether random detector generation would be feasible. For example, Kim & Bentley [10] showed random generation to be infeasible when they attempted to use 33 features of a connection.

2.2 Real-valued detectors

To overcome these problems, González [7] has proposed the use of real-valued anomaly detectors for network intrusion detection. The most significant improvement that his work offers over those previously discussed is a distinction between a detector *genotype* and a detector *phenotype*. At the genotypic level, his detectors are vectors of real numbers. At the phenotypic level, they are interpreted as specifying intervals on the space of real numbers. These intervals are then read as conditions for an IF-THEN rule, where the consequent is that an anomaly has been detected. This means that an antigen vector is matched by a detector if the components of the antigen vector lie within the corresponding intervals specified by the detector. A GA is used to generate detectors in this work, rather than the random generation of Forrest.

In applying this work to network intrusion detection, González used intervals on 3 aggregate network traffic statistics. Specifically, he used the total number of packets, the number of ICMP packets, and the number of bytes of data transmitted, over a period of 1 second. In addition, he also used a sliding time window to attempt to detect temporal anomalies, e.g. if the window size was 3 then he would consider the last 3 observations together as a sequence.

The key advantage of this approach is that it is easy to interpret the detectors in terms of domain knowledge. This is because at the phenotypic level they can be interpreted as conditional rules specifying intervals on the three network traffic statistics. By contrast, with the binary string representation used in LISYS there is no corresponding phenotype, and the *r*-contiguous bits matching rule does not have an intuitive interpretation at the domain level.

However, we argue that there is still a problem of scalability up to a real-world IDS with his approach. This is because his system only considers aggregate network traffic statistics. While such information is undoubtedly useful, it could not be used in isolation in a real system. For example, it is surely important to know the service ports that are being accessed. However, it is non-trivial to incorporate information about specific connections with aggregate traffic statistics into the same detector. This is because when discrete fields such as the port number are included in the detectors then the fitness of a detector must be calculated in a different way.

In this paper we therefore present a system that still uses the genotype-phenotype distinction and conditional rule interpretation, but which deals with properties of individual TCP connections. In so doing, our system is forced to deal with discrete values. We have designed a detector representation scheme and generation algorithm that is able to cope with this.

3 An artificial immune system for network intrusion detection that evolves discrete-valued detectors

This section describes the representation scheme used for our detectors and the GA used to generate them.

3.1 Detector representation

Figure 1 lists the fields contained in a detector phenotype. All of the fields reference properties (network features) of an incoming TCP connection.

The choice of features for our detectors was motivated by a desire to be able to detect port scans and certain types of denial of service attack (see [9] for an explanation of these). Our system uses more features than both González's

1. The port category.
2. Interval on the duration of the connection, in milliseconds.
3. Interval on the total number of packets received over the connection
4. Interval on the number of packets received over the connection that have a data payload.
5. The number of FIN packets received (0, 1 or more than 1).
6. The number of packets received with the urgent flag set (0, 1 or more than 1).
7. Interval on the maximum time the connection was idle.

Figure 1: Specification of the detector phenotype.

system and LISYS, as the use of only 3 network features in those systems seems excessively limiting and does not give any idea of how the algorithms will scale. However, a detector does not have to use all of the features, it is free to leave any of the fields undefined. By contrast, in the system proposed by González, every field had to be specified by every detector.

Regarding the port category field, there are 65535 possible ports, far too many to consider each one individually for the purpose of detector generation. Our system therefore groups ports into functional categories, as shown in Table 1. As a consequence of this categorisation, the port number field of a detector can now only store a value between 1 and 9, rather than between 1 and 65535. This greatly assists the search for detectors, by reducing the size of the search space.

Category	Description
1	Remote shell
2	FTP
3	HTTP
4	Mail
5	SQL
6	Several ports known to be unsafe
7	Network diagnostics
8	0 - 49151 (excluding those above)
9	49152 - 65535

Table 1: Port categories.

A genotype consists of two genes for each interval field (specifying the lower and upper limits of the interval) and one gene for each non-interval field. At this level, it was deemed necessary to cluster the values of the interval fields in to one of 9 categories. The reason for this is that fields such as the number of packets received can take on values from a very large range, and it was felt that this resulted in too large a search space. There is also no need in this application to represent the values with such a precision, e.g. whether the number of packets received is 131 or 132 is unlikely to provide any useful indication of whether an intrusion has occurred.

The decision to use 9 categories was somewhat arbitrary, but was chosen to match the port categorisation. It also seems to produce acceptable results. The clustering was performed for each gene by taking the values observed for that gene in the training data and dividing them into 9 bins, such that approximately the same number of training examples were in each bin. This is the standard equal frequency binning algorithm.

3.2 Detector generation

This section describes how a set of detectors is produced using a steady-state GA.

Detector generation is a multi-modal search problem, since we require individual detectors to cover different parts of the non-self space. This work follows González in the use of the deterministic crowding algorithm [11] for this purpose.

Uniform crossover is used to produce a single child from two parents. Each gene of the child is then mutated with a small probability. This mutation is performed by replacing the value of the gene with a randomly chosen value from the list of those allowed for that gene. Alternatively, the value of the gene is randomly set to -1, which means that the corresponding field is left undefined at the phenotypic level. Under the deterministic crowding scheme, the child replaces the parent that it is most similar to if it is fitter than that parent.

We define similarity at the phenotypic level as follows. A similarity score is computed for each corresponding field in the two detectors. For non-interval fields, a score of 1 is given if the fields store the same value, otherwise the score is 0. For interval fields, the score is the degree of overlap between the corresponding intervals, normalised to lie between 0 and 1. The sum of the scores from each field then yields the overall similarity between the two detectors.

There are two objectives to optimise during detector generation, shown in Figure 2. One approach for dealing with multiple objectives is to weight each objective, and then sum the weighted objective values to yield the overall fitness [4]. However, our two objectives yield values on different scales, making it inappropriate to weight them directly [4]. We therefore use the Sum of Weighted Ratios method proposed by Bentley & Wakefield [4], shown in Figure 3. The overall fitness of a detector is computed as shown in Figure 4.

- obj^1 = **Maximise** the *generality* of the detector. Generality is defined as the sum of the ranges specified by each interval field plus the number of undefined non-interval fields, normalised to lie between 0 and 1.
- obj^2 = **Minimise** the number of self samples in the training data matched by the detector.

Figure 2: The two objectives during detector generation. These are discrete versions of those used by González. However, our discrete representation means that they must be computed in a different way from in that work.

$$fr_i^j = \frac{(obj_i^j - min(obj^j))}{(max(obj^j) - min(obj^j))}$$

Figure 3: The fitness ratio score of individual i at objective j (i runs from 1 to the population size, j from 1 to 2). obj_i^j is the raw fitness value of individual i at objective j, as defined in Figure 2. $min(obj^j)$ is the lowest raw fitness value for objective j in the population, $max(obj^j)$ is the largest. This equation is taken from [4].

$$fitness_i = w_1 * fr_i^1 - w_2 * fr_i^2$$

Figure 4: The overall fitness of individual i. w_1 and w_2 are the objective weights, and must sum to 1. The negative sign is because the second objective is a penalty (see Figure 2).

At the end of the final iteration of the steady-state GA, it could still be the case that a detector could match some of the self samples in the training set. Through experimentation, we have found that a sensible thing to do with such detectors is to simply remove them from the detector set.

4 Experimental results

This section presents results comparing the performance of our IDS to that of González. The same training and testing datasets are used as in that work. Specifically, a subset of the 1999 DARPA intrusion detection evaluation dataset from MIT Lincoln Labs [12] is used, corresponding to connections from outside the LAN to the machine with host-name *Marx*. Although this dataset is now quite old, it is nevertheless still widely used to evaluate intrusion detection systems. Week 1 of this dataset, consisting of entirely normal network activity, is used for detector generation. The evolved detectors are then tested on the connections in the second week, which contains 5 attacks against *Marx*. 2 of these are denial of service attacks, while the remaining 3 are port scans that probe the machine for vulnerabilities.

When evaluating the system, two performance metrics must be considered. The first is the attack detection rate (ADR), which is the percentage of the 5 attacks that were recognised by the detectors. The second is the false positive rate (FPR), which is the percentage of normal network activity mistakenly flagged as anomalous. Because our system monitors each individual incoming TCP connection, the FPR is computed as the number of normal connections marked as anomalous divided by the total number of normal connections. When computing the ADR, we consider an attack to be detected if one or more connections comprising the attack are marked as anomalous.

4.1 Experimental setup

In all of the experiments described below, the results reported are the mean from 100 trials. This is necessary because our detector generation algorithm is stochastic, i.e. two different runs of the algorithm are unlikely to produce exactly the same set of detectors. In all cases, the steady-state GA was executed for 25000 iterations, as preliminary experimentation had revealed that even for larger population sizes there was nothing to be gained by allowing it to run for longer. The mutation rate was set so that on average a single gene was mutated, while full crossover was used. The population was initialised by generating initial detectors that specified a random value for a gene with a probability of 0.5. The remaining genes were assigned a value of -1, meaning that the corresponding phenotypic field is left unspecified. Finally, results for population sizes of 400, 800, 1200 and 1600 are presented here because they provide a good illustration of the different ADR and FPR values obtainable by the system.

Our system has two parameters that can be varied to control the trade-off between the ADR and FPR. These are the population size and the objective weights. Recall that the size of the detector set is dependant on, but not completely determined by, the population size. This is because of the final negative selection filter, which removes any detectors that still match self at the end of the GA. Intuitively, one would expect a greater number of detectors to lead to a larger FPR, as the more detectors there are, the more likely it is that some of them will cover parts of the self space. Similarly, setting the objective weights directly specifies the importance of not covering the self space against having detectors that cover a large total area of antigen space. The weight setting therefore influences the trade-off between the ADR (covering a large total area of antigen space) and the FPR (covering a small area of the self space).

4.2 Receiver operator characteristics analysis

When a system has parameters that control the trade-off between the detection and false positive rates, a useful tool for evaluation is the ROC (receiver operating characteristics) curve [14]. In our case, this curve plots the FPR against the ADR for various settings of a parameter. The ideal system has a curve which passes through the point (0,1), corresponding to an FPR of 0% and an ADR of 100%. Therefore, a curve which passes closer to this point should be prefered.

A series of ROC curves showing the performance of our system on the test data is shown in Figure 5. Each curve represents the performance with a different population size. For a particular population size, the points along the curve were obtained by varying the weightings of the two objectives. Specifically, the weight of each objective was varied from 0.1 to 0.9 in increments of 0.1, respecting the constraint that the two weights must sum to 1.

From Figure 5, it can be seen that the curve for a population of size 400 reaches further to the left than any of the other curves. A point in the ROC plot further to the left indicates a lower FPR. Therefore, the plot shows that the lowest FPR is obtained with a population of size 400, i.e. the smallest population size that was

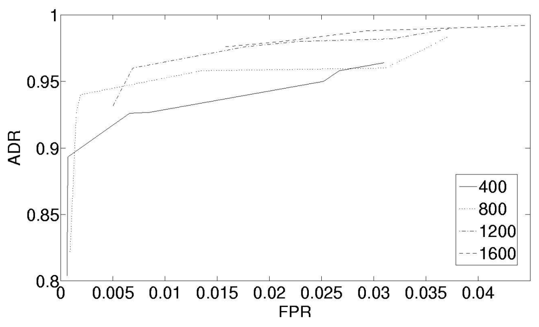

Figure 5: ROC (receiver operating characteristics) curves [14] for varying population sizes.

trialled. However, all of the other curves are able to reach larger values on the y-axis, corresponding to a larger ADR. Overall, the trend is that increasing the population size moves the curve upwards and to the right. This means that increasing the population size increases the largest obtainable ADR, but also increases the smallest possible FPR.

This observed trend was expected intuitively, as a greater number of detectors would be expected to cover a greater area of the total antigen space, and hence a greater area of the self space. Covering a greater area of the total antigen space therefore increases both the ADR and FPR.

4.3 Comparison of discrete and real-valued detectors

In addition to analysing the ADR-FPR tradeoff our system, we have also compared the performance of our discrete-valued detectors to the real-valued detectors used by González [7]. In order to do so, we adopt his approach of fixing the FPR at a maximum of 1% and then looking at the best ADR that can be achieved within that constraint. Table 2 shows the results achieved by our system, and Table 3 the results achieved by González. The correspondance between Table 2 and Figure 5 is that the table gives the y co-ordinates of the points on the ROC plot that have an x co-ordinate of 0.01. The result for a population of size 1600 is marked as N/A in the table, because an FPR as low as 1% was not obtainable with that population size.

The corresponding weight settings are also given in the caption of Table 2. Note that specifying a larger weight for the generality objective means that fewer detectors fail to match any self sample at the end of the GA. This makes the final detector set smaller, reducing the FPR, which explains why a larger weight was needed with larger population sizes.

Population size	Best ADR (FPR max 1%)
400	92.7%
800	94.0%
1200	96.0%
1600	N/A

Table 2: Best ADR with FPR fixed at 1% for our discrete-valued detectors.. The result is labeled as N/A for a population size of 1600 because it was not possible to obtain an FPR that low with such a large number of detectors. For a population of size 400, the weight of the generality objective, w_1, was set to 0.3. For a population size of 800 it was set to 0.6, and for size 1200 it was set to 0.8. These weight settings were determined empirically.

Window size	Best ADR (FPR max 1%)
1	82.1%
3	87.5%

Table 3: Best ADR with FPR fixed at 1% for González's [7] real-valued detectors. Note that the time window, rather than population, size was used as the variable parameter in that work.

Note that whereas in our work we vary the population size as a parameter, in the work of González the population size is fixed at 100 but the time window size is varied. We suspect that the reason that our system requires a larger population size is that we use more fields in our detectors (7 rather than 3). However, our choice of network features and representation means that we do not need to use a time window to detect the 5 attacks, whereas González was unable to detect two of them without such a window.

From the tables it can clearly be seen that our system is able to achieve a better ADR when the FPR is fixed at a maximum of 1%. In fact, this is the case when any of our evaluated parameter settings is compared against any of González's. Fixing the FPR at 1% is an appropriate way to compare such systems, as a system with a high FPR will effectively be useless due to the fact that it will overload the network administrator with alerts. Unfortunately, we were not able to run any statistical significance tests to compare our results to those of González, as González does not provide the variance of his results.

4.4 Examples of generated detectors

Earlier in this paper, we agreed with the claim made by González [7] that higher-level detector representations are more appropriate than binary. We supported his argument that this is because it is hard to understand the operation of binary detectors. This is particularly the case when there is no distinction between the detector genotype and phenotype, as detector-antigen matching is then often defined in terms of a stretch of bits on one string being identical to a stretch of bits somewhere on the other. By contrast, both the work of ourselves and González features detectors with a phenotype that corresponds to IF-THEN rules. In our case, the IF part of a rule is conditions on properties of a TCP connection, and the consequent is that an anomaly has occurred.

To demonstrate the fact that such rules are easy to understand, we present two examples of evolved detectors. Figure 6 shows a detector capable of detecting the *Back* denial of service attack. An explanation of this attack is provided in [9], however, it essentially involves a request to a server for a URL containing many slashes, which takes a long time to process. Should this attack occur, then the form of the attack is immediately obvious from the phenotype of the activated detector. In this case, from looking at the detector phenotype it can be seen that the attack involved a connection to a HTTP server, and that between 2 and 10 data packets were received. Such information would go a long way towards creating a profile of the attack.

```
Genotype:
3 -1 -1 -1 -1 3 -1 -1 -1 -1 -1
Phenotype:
Port Category = HTTP AND
Number of data packets = [3,*]
```

Figure 6: An evolved detector that detects the *Back* denial of service attack. The value of 3 in the first gene means the condition **port category = 3** in the phenotype. The value of 3 in the sixth gene is the lower bound on the number of data packets received. Note that because of our clustering procedure, this does not correspond to the integer 3 but to the third cluster from the equal-frequency binning algorithm. With this dataset, cluster 3 corresponds to a number of data packets between 2 and 10 (inclusive). Finally, the value of -1 in the other genes means that the corresponding fields are left unspecified in the detector phenotype.

Figure 7 shows a detector capable of detecting a port scan. Note that this detector only specifies one field, the port category. The detector specifies that any connection to a port used for network diagnostics is anomalous. This is perfectly feasible, as it is rare to connect to such ports, and so if a connection to one is made then it may well be a port scan. It also illustrates how our system detects attacks by performing anomaly detection. In this case, a port scan is detected by observing a connection to a port that is not normally accessed. Finally, both of the detectors shown here do not specify many conditions. This is a consequence of the generality objective in the fitness function, which aims to evolve detectors that can match many connections.

```
Genotype:
7 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Phenotype:
Port Category = network diagnostics
```

Figure 7: An evolved detector that detects a port scan. The value of 7 in the first gene means the condition **port category = 7** in the phenotype. The value of -1 in the other genes means that the corresponding fields are left unspecified in the detector phenotype.

5 Conclusion

This paper has presented an anomaly based IDS inspired by the human immune system. Our work differs from previous artificial immune systems in that we use a GA to evolve discrete-valued detectors, rather than the real-valued detectors of González. This has enabled our representation to incorporate such pertinent information as the port numbers of connections and whether or not the connection was closed correctly. By contrast, such information could not be incorporated in a purely real-valued representation. Although some other works, e.g. [8] and [1], have used discrete fields in their detectors, they have used random generation, rather than a GA. Our work has shown how to formulate the competing objectives of maximising the area of the antigen space covered while minimising overlap with the self space, in the discrete case.

Through the use of discrete statistics on specific TCP connections, we have shown that our system is able to outperform that of González in terms of a higher attack detection rate for the same false positive rate. In addition, our system did not require the computational expense of a time window to detect the 5 test attacks, whereas González's system is unable to detect two of the attacks without such a window. However, in the general case it is likely to be beneficial to use both real-valued and discrete detectors.

In the future, we intend to look at two broad ways in which our system could be improved. Both of these are aimed at allowing the system to scale to handle larger amounts of network traffic. Large amounts of traffic will require more detectors, as the antigen space is more complicated, i.e. it is harder to distinguish self from non-self. However, our experimental analysis has revealed that increasing the number of detectors increases the false positive rate.

The first way that we propose to deal with this problem is to use some sort of data fusion technique [3] to combine the outputs of different detectors. For example, detectors close to each other in the antigen space could form a voting ensemble. Our second idea is to reduce the number of self training samples through the use of a clustering technique, thereby reducing the amount of data that the detector generation algorithm has to work with.

Acknowledgements

This work was supported by the School of Computer Science at the University of Birmingham.

References

[1] K. P. Anchor, P. D. Williams, G. H. Gunsch, and G. B. Lamont. The Computer Defense Immune System: current and future research in intrusion detection. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*, volume 2, pages 1027–1032. IEEE Press, 2002.

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

[3] T. Bass. Intrusion detection systems and multisensor data fusion. *Communications of the ACM*, 43(4):99–105, 2000.

[4] P. J. Bentley and J. P. Wakefield. Finding acceptable solutions in the Pareto-optimal range using multiobjective genetic algorithms. In P. K. Chawdhry, R. Roy, and R. K. Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, pages 231–240. Springer, 1997.

[5] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for Unix processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 120–128. IEEE Press, 1996.

[6] S. Forrest, A. 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 Press, 1994.

[7] F. González. *A Study of Artificial Immune Systems Applied to Anomaly Detection.* PhD thesis, The University of Memphis, May 2003.

[8] S. A. Hofmeyr and S. Forrest. Immunity by design: an Artificial Immune System. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99)*, volume 2, pages 1289–1296. Morgan Kaufmann, 1999.

[9] K. Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, Massachusetts Institute of Technology, 1998.

[10] J. Kim and P. J. Bentley. Evaluating negative selection in an Artificial Immune System for network intrusion detection. In *Proceedings of the 2001 Genetic and Evolutionary Computation Conference (GECCO'01)*, pages 1330–1337. Morgan Kaufmann, 2001.

[11] S. W. Mahfoud. Crowding and preselection revisited. In *Proceedings of the Second Conference on Parallel Problem Solving from Nature*, pages 27–36. North-Holland, 1992.

[12] MIT Lincoln Labs. 1999 DARPA intrusion detection evaluation. Available online at: http://www.ll.mit.edu/IST/ideval/.

[13] B. Mukherjee, L. T. Heberlein, and K. N. Levitt. Network intrusion detection. *IEEE Network*, 8(3):26–41, 1994.

[14] J. A. Swets. Measuring the accuracy of diagnostic systems. *Science*, 240:1285–1293, 1988.