Eine Immungenetische Technik zur Erkennung von Anomalien Im Rechnernetzwerkverkehr

Autor: Dipl.-Inform. Jörg Angermayer
Studiengang: Master of Computer Science
Erstellt am: 10. Januar 2004

Inhaltsverzeichnis

Vorwort

Dieser Aufsatz beschreibt eine Technik die es erlaubt, basierend auf der Funktionsweise eines Immunsystems, Störungen innerhalb eines Rechnernetzwerkes zu erkennen. Dabei werden die Parameter eines Rechnernetzwerks als Muster verstanden die dann durch einen genetischen Suchalgorithmus in verschiedene Klassen aufgeteilt werden. Diese Klassen liefern eine Abstufung die von normalen Klassen bis hin zu vollkommen abnormalen Klassen geht. Die Muster abnormaler Klassen entsprechen einem abnormalen Rechnernetzwerkverkehr. Des weiteren wird die Nützlichkeit der hier vorgestellten Technik anhand von Experimenten überprüft und diskutiert. Abschließend werden die Ergebnisse der Untersuchung zusammengefasst.

Einleitung

Die meisten Techniken zur Erkennung von Störungen bei einem Rechnernetzwerkverkehr arbeiten mit einem Modell das ein Profil für die normalen Umgebungsparametern des Netzwerkes verwendet. Dabei werden für jeden Parameter obere und untere Grenzwerte festgelegt. Werden diese Grenzwerte über bzw. unterschritten liegt ein abnormales Verhalten des Rechnernetzwerkverkehrs vor. Häufig werden dabei statistische Modelle (Denning, 1986; Eskin, 2000) verwendet. Bei diesen Modellen werden die für das System spezifischen Werte, die bei einem normalen Betrieb des System vorkommen statistisch ausgewertet. Diese Informationen werden genutzt um zu berechnen, welche der gegebenen Werte am Wahrscheinlichsten vorkommen. Je kleiner diese Wahrscheinlichkeit ist, desto größer ist die Wahrscheinlichkeit das es sich um eine Störung im System handelt. Die starre Auswertung der Systemzustände, mit einer festgelegte Wahrscheinlichkeitsschranke, hat den Nachteil das zeitlich auftretende Veränderungen in den Systemzuständen, die während eines normalen Betriebs auftreten können, nicht berücksichtigt werden.

Andere Techniken versuchen aus den momentanen und den vorangegangenen Systemzuständen den Folgezustand des Systems zu bestimmen(Crosbie and Spafford, 1995; Dagupta and Gonzalez, 2001). Unterscheidet sich der Folgezustand wesentlich von dem Zustand der vorhergesagt wurde, so geht man von einer Störung aus. Ein großer Nachteil bei dieser Technik ist der enorme Zeitaufwand und der enorme Speicherplatzbedarf bei dem Trainieren des Modells und bei der Anwendung der Technik im Rechnernetzwerk.

Das Ziel ist es einen Algorithmus zu finden der Störungen (z.B. im Rechnernetzwerk) erkennt. Dabei soll der Algorithmus, anders als bei der statistischen Auswertung der Systemzustände und der dabei verwendeten Wahrscheinlichkeitsschranke, flexibel auf zeitliche Veränderungen der Systemzustände reagieren. Hinzu kommt das der Algorithmus eine großen Menge von Daten, die bei einem Rechnernetz auftreten können, effektiv auswerten soll. „Data mining“- Techniken wurden mit einigem Erfolg zur Lösung der genannten Probleme eingesetzt(Lane, 2000; Lee and Stolfo, 1998).

Diese Herangehensweise hat den Vorteil, das man aus großen Datenmengen nützliches Wissen extrahiert und dieses Wissen auf Regeln abbildet. Bei verschiedenen Arbeiten am Rechnernetzwerk möchte man Verbindungsinformationen bekommen und manchmal ist es notwenig zu wissen ob die Anfrage nach Verbindungsinformationen ein normaler Vorgang ist oder ob es sich um einen Angriff handelt.

Dieser Fachaufsatz zeigt einen Lösungsansatz der nicht auf einer Strukturierung der vorhandenen Systemdaten beruht, sondern nur normale Daten bzw. Parameter verwendet um ein Profil eines Systems zu erstellen. Der Lösungsansatz soll angewendet bzw. getestet werden, in wieweit abnormales Verhalten in einem System(z.B. in einem Rechnernetzwerk) erkannt wird.

Der Lösungsansatz basiert auf der Funktionsweise des Immunsystems. Der von Forrest's (self/non-self)(Forrest et al., 1994) vorgeschlagenen Algorithmus, der auf eine zwei Klassen Erkennung aufbaut, wird zu einer mehrfach Klassen Erkennung erweitert. Dabei wird die „non-self"-Klasse in mehrere Klassen unterteilt, um einen Grad für die Störung festlegen zu können. Die Technik besteht im wesentlichen aus einem genetischen Algorithmus, der Regeln aufstellt um die „non-self-Klasse" bzw. die „non-self-Klassen" abzudecken. Experimente werden vorgeführt und deren Resultate werden mit einer Technik verglichen, die nur die „self„-Klasse für die Erkennung von Störungen verwendet.

Hintergrund und vorangegangene Arbeiten

Die Idee, den logischen Aufbau des Immunsystems für die Sicherheit von Rechnern einzusetzen(Dasgupta, 1999; Hofmeyr and Forrest, 2000; Kephart, 1994) hatte man bereits 1994. Die Arbeitsgruppe um Stephanie Forrest an der Universität von Neu Mexiko hatte bereits an einem Forschungsprojekt gearbeitet, bei dem es um ein künstliches Immunsystem für Computer ging. Es wurden Änderungen von geschützten Daten überwacht. Dabei fanden sie heraus das, das eigentliche Problem darin bestand anhand der Daten zu bestimmen ob diese Daten zulässige und korrekte Daten des Systems sind oder ob diese Daten nicht zum eigentlichen System gehörten. Das Erkennen von nicht zulässigen Systemzuständen wird als „negative-selection“- Algorithmus beschrieben(Forrest et al., 1994). In einer anderen Arbeit wurde dieser Algorithmus verwendet, um UNIX-Prozesse zu überwachen und um zu erkennen welche Anweisungen sich schädlich auf ein Rechnersystem auswirkt. Kephart(Kephart, 1994) schlug vor, diesen Algorithmus für die Erkennung von Viren zu nutzen. Dabei werden bekannte Viren an einer dem Virus typischen Programmsequenz erkannt und unbekannte Viren würden anhand Ihres ungewöhnlichen Verhaltens im Rechnersystem auffallen und daran erkannt werden.
Kim and Bentley(Kim and Bentley, 2001) untersuchten den „negative-selection“-Algorithmus um Störungen in einem Rechnernetzwerk zu erkennen. Die Untersuchung ergab, daß der Algorithmus ernsthafte Probleme mit der Verarbeitung aller Daten eines normalen Datentransfers hatte. Daher wurde vorgeschlagen, den Algorithmus nur als Filter für die Erkennung von Störungen zu verwenden. Bei dem derzeitigen Stand der Forschung zu diesem Thema sieht man, daß es möglich ist dieses Problem zu überwinden, indem man den Datenfilter modifiziert.

Definition der Probleme bei der Anomalie Erkennung

Der Zweck der Erkennung von Abweichungen ist zu Erkennen, welcher Systemzustand normal und welcher Systemzustand nicht normal ist. Der Zustand eines Systems kann durch eine Reihe von Merkmalen des Systems beschrieben werden.

Definition 1. Zustandsraum eines Systems.

Der *Zustand eines Systems* läßt sich als Vektor von Merkmalen verstehen, $x^i = \left(x_1^i,..., x_n^i\right) \in [0.0,1.0]^n$. Der Raum der Zustände ist $S \subseteq [0.0,1.0]^n$.

Die Merkmale können gegenwärtige und vergangene Werte der Systemvariablen darstellen. Die aktuellen Werte der Systemvariablen können normalisiert werden, so daß sie sich in einem Intervall von $[0.0,1.0]$ befinden.

Definition 2. Zustandsraum der „Self“-Zustände

Ein Satz von Merkmalsvektoren , $Self \subseteq S$ stellt die normalen Zustände des Systems dar. Das Komplement dazu wird als Non_Self bezeichnet. $Non\_Self = S - Self$. In vielen Fällen wird $Self\left(or\ Non\_Self\right)$ als Funktion beschrieben $\chi_{self} : [0.0,1.0] \to \{0,1\}$

$$\chi_{self}\left(\vec{x}\right) = \left\{ \begin{array}{ll} 1 & \text{if } \vec{x} \in Self \\ 0 & \text{if } \vec{x} \in Non\_Self \end{array} \right\}$$

In natürlichen Systemen (z.B. bei einem Immunsystem) gibt es oft keine scharfe Abgrenzung zwischen einem normalen und einem nicht normalen Zustand. Anstatt dessen gibt es eine feine Abstufung, in wieweit der momentane Zustand normal bzw. abnormal ist. Die folgende Definition macht den Versuch, dies zu berücksichtigen.

Definition 3. Unscharfer Zustandsraum

Die Funktion von Definition 2 ist so zu erweitern, das jeder Wert in dem Intervall $[0.0,1.0]$ gültig ist. Bei $[0.0,1.0]: \mu_{self} : [0.0,1.0]^n \rightarrow [0.0,1.0]$ gibt es somit ein Maß für den Zustand, der Werte zwischen 0.0 für einen abnormalen bis hin zu 1.0 für einen normalen Zustand annehmen kann.

Durch die Einführung von unscharfen Zuständen kann man über den Schwellwert t steuern, wann ein abnormaler bzw. normaler Zustand des Systems erreicht wurde[GoDa2002].

$$\mu_{self,t}\left(\vec{x}\right) = \left\{ \begin{array}{ll} 1 & \text{if } \mu_{self}\left(\vec{x}\right) > t \\ 0 & \text{if } \mu_{self}\left(\vec{x}\right) \leq t \end{array} \right\}$$

Definition 4. Anomalie Erkennungsproblem

Gegeben sei ein Satz von normalen Systemzuständen $Self' \subseteq Self$, erzeuge die Funktion $\chi_{self}\left(or\ \mu_{self}\right)$ die eine möglichst gute Erkennung eines normalen Systemablaufs liefert. Anhand der Funktion sollte man feststellen können ob sich das System in einem normalen oder in abnormalen Zustand befindet.

Konzept

Um den Non_Self Bereich einzugrenzen wird derzeit ein genetischer Algorithmus eingesetzt, der unter dem Namen „deterministic crowding“(Maahfoud, 1992) bekannt ist. Crowding-Methoden sind Verfahren, “bei denen nur ein Teil der Individuen der Population durch Crossover und Mutation verändert wird und bei denen die Nachkommen die ähnlichsten Individuen in der Population ersetzen”(Maahfoud).
Die Erkennung von Non_Self - Bereichen wird mit Hyper-Rechtecken in einem multidimensionalen Raum korrespondieren. Die Erkennung von Non_Self Bereichen kann durch regeln folgender Struktur dargestellt werden[GoDa2002]:

$$R^1 : If \;\; Cond_1 \;\; then \;\; Non\_Self$$
$$\vdots$$
$$R^m : If \;\; Cond_m \;\; then \;\; Non\_Self$$

- $Cond_i = x_1 \in \left[low_1^i, high_1^i\right] \; and \;\; ... and \;\; x_n \in \left[low_n^i, high_n^i\right]$
- $(x_1,...,x_n)$ ist ein Merkmalsvektor
- $\left[low_i^j, high_i^j\right]$ repräsentiert die Merkmalswerte von x_i in dem Bedingungsteil der Regel R^j.

Durch die Regeln werden Hyper-Kuben in dem Lösungsraum definiert($[0.0,1.0]^n$).
Die Hyper-Kuben sind so zu wählen, daß sie möglichst den ganzen Non_Self Bereich abdecken. Um dies zu erreichen wird ein „Genetischer“ Algorithmus verwendet. Dieses vorgehen entspricht der Definition 2. Dabei wird ein Satz von Regeln($R = \left\{R^1,...,R^m\right\}$) wie folgt definiert[GoDa2002]:

$$\chi_{non\_self,R}\left(\vec{x}\right) = \begin{cases} 1 \, if \; \exists R^j \in R \; derart \; das \; \vec{x} \in R^j \\ 0 \; ansonsten \end{cases}$$

Alles was von den Hyper-Kuben nicht abgedeckt wird, wird dem $Self$ Bereich zugeordnet.

Entsprechend Definition 3 kann man den Non_Self Bereich in mehreren Abstufungen einteilen. Dazu wird ein Schwellwert (v) genutzt, der bei einem höheren Wert einen größeren $Self$ Bereich abdeckt und bei einem niedrigeren Wert einen kleineren $Self$ Bereich abdeckt. Für jeden Schwellwert entstehen somit eine neue $Self$ und Non_Self Bereichseinteilung des Lösungsraums. Die dazugehörigen Regeln lauten[GoDa2002]:

$$
\begin{aligned}
R^1 &: If \;\; Cond_1 \;\; then \;\; \text{Level 1} \\
&\;\; . \quad . \quad\quad . \\
&\;\; . \quad . \quad\quad . \\
&\;\; . \quad . \quad\quad . \\
R^i &: If \;\; Cond_i \;\; then \;\; \text{Level 1} \\
R^{i+1} &: If \;\; Cond_{i+1} \;\; then \;\; \text{Level 2} \\
&\;\; . \quad . \quad\quad . \\
&\;\; . \quad . \quad\quad . \\
&\;\; . \quad . \quad\quad . \\
R^j &: If \;\; Cond_j \;\; then \;\; \text{Level 2} \\
&\;\; . \quad . \quad\quad . \\
&\;\; . \quad . \quad\quad . \\
&\;\; . \quad . \quad\quad .
\end{aligned}
$$

Die unterschiedlichen „Level“ bzw. Abstufungen sind hierarchisch geordnet. Level 1 beinhaltet Level 2, Level 2 beinhaltet Level 3 usw. Ein Zustand kann somit mehreren Levels zugeordnet werden. Je größer die Störung ist, desto größer ist der Level. Wenn dem aktuellen Zustand mehrere Levels zugeordnet wurden, werden ihm nicht alle, sonder nur der größte Level zugewiesen. Dieser Sachverhalt spiegelt sich in folgender Funktion wieder[GoDa2002]:

$$\mu_{non\_self}\left(\overrightarrow{x}\right) = \max\left\{ l \mid \exists R^j \in R, \vec{x} \in R^j and\ l = level\left(R^j\right)\right\} \cup \{ 0 \}$$

Der genetische Algorithmus für die Regelerzeugung

Die hier verwendete Technik (deterministic crowling, (Mahfoud, 1992)) verwendet einen genetischen Algorithmus der bei einem einmaligen Durchlauf die verschiedenen Regeln erzeugt. Dazu benötigt der Algorithmus als Eingabe einen n-dimensionalen Satz von Merkmalsvektoren $S = \left(x^1,...,x^m\right)$ die den normalen Zustand des Systems repräsentieren, die Anzahl von Abstufungen(numLevels) und die Angabe der dazugehörigen Schwellwerte $\left\{v_1,...,v_{numLevels}\right\}$. Der Algorithmus sieht wie folgt aus:

```
for i = 1 to numLevels
initialisiere eine Population mit zufällig gewählten Individuen
    for j = 1 to numGenerations
        for k = 1 to population_size/2

        wähle zwei Individuen mit vergleichbarer Wahrscheinlichkeit
        ohne sie zu ersetzen
        kreuze die Individuen um ein Kind zu erzeugen und
        mutiere das Kind

        if dist(child, parent1) < dist(child, parent2)
                and fitness(child) > fitness(parent1)
                ersetze parent1 mit dem child
        elsif dist(child, parent1) ≥ dist(child, parent2)
                and fitness(child) > fitness(parent2)
                ersetze parent2 mit dem child
        endif
        endfor
    endfor

entnehme die besten Individuen aus der Population und
erweitern damit Ihre endgültige Lösung.
endfor
```

Jedes Individuum(Chromosom) im genetischen Algorithmus repräsentiert den Bedingungsteil einer Regel. Dabei werden alle Regeln einem Level bzw. Abstufung zugeordnet. Dennoch werden über die „fitness“-Funktion verschiedene Level durch Änderung des Schwellwertes v durchlaufen.

Der Bedingungsteil der Regeln ist durch einen minimalen und einen maximalen Merkmalswert begrenzt. Das Chromosom, das die Merkmalswerte beinhaltet besteht aus einem Array von „float“-Werten. Die verwendete Kreuzung ist eine gleichförmige Kreuzung und die Mutation ist eine „Gaußsche – Mutation“. Unter einer gleichförmigen Kreuzung versteht man, das erzeugen eines neuen Chromosoms(Child) aus Zwei anderen Chromosomen(Parents), wobei die Parents zu gleichen Anteilen an der Erzeugung des Childs beteiligt sind(siehe Bild 1).

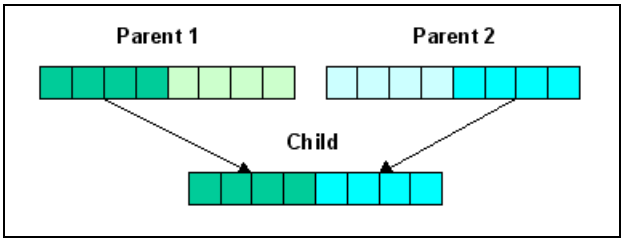

Bild 1: Abstraktes Beispiel der gleichförmigen Kreuzung

Die „Gaußsche – Mutation“ verändert zufällig(basierend auf der Gaußschen Zufallsverteilung(Gaußglocke)) einzelne oder mehrere Werte eines Chromosoms(siehe Bild 2).

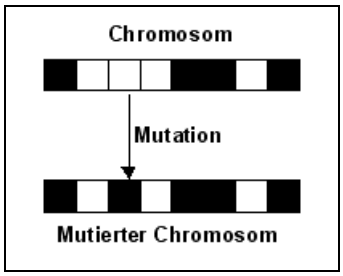

Bild 2: Abstraktes Beispiel der Mutation.

Fitness Bewertung

Gegeben sei eine Regel mit dem Bedingungsteil $x_1 \in [low_1, high_1]$and ...and $x_n \in [low_n, high_n]$, des weiteren sei $x^j = (x_1^j,...,x_n^j),\ x^j \in R$ ein gültiger Vektor und eine Hyper-Kugel mit Ihrem Mittelpunkt x^j und dem Radius v_i soll die Hyper-Kuben an den Punkten $(low_1,..,low_n)$ and $(high_1,...,high_n)$ unterbrechen.

Die Fitness einer Regel zerfällt nun in zwei Faktoren:

- In die Anzahl von Trainingsätzen S, die zu der Teilmenge die folgende Regel beschreibt:

$$num\_elements(R) = \left\{ x^i \in S \mid x^i \in R \right\}$$

- Das Volumen der Teilmenge ist durch folgende Regel zu beschreiben

$$volume(R) = \prod_{i=1}^{n} (high_i - low_i)$$

Die Fitness ist Definiert als:

$$fitness_R = volume(R) - C \cdot num\_elements(R)$$

C ist ein Faktor für die Sensibilität der Fitness. Es stellt ein Wert für die Bestrafung dar, der wirksam wird wenn die Regel versucht normale($Self$) Bereiche abzudecken. Die Fitness kann ein negatives Ergebnis liefern.

Individuelle Abstandsberechnung

Ein gute Messung der Abstände zwischen den Individuen(bzw. Chromosomen) ist wichtig für die Leistungsfähigkeit des Algorithmus. Daher ist es erwünscht das Individuen mit anderen Individuen ausgetauscht werden, die besser passen. Die Abstände werden wie folgt gemessen:

$$dist(c, p) = \frac{volume(p) - volume(p \cap c)}{volume(p)}$$

- c : „child“ Individuum
- p : „parent“ Individuum

Wenn ein „child“-Individuum einen Bereich abdeckt indem ein „parent“-Individuum ist, bedeutet dies eine gute Generalisierung des „parent“. In dem Fall das ein „child“-Individuum nur einen kleinen Bereich vom „parent“-Individuum abdeckt ist dies natürlich keine Generalisierung.

Experiment und Resultat

Um den Algorithmus zu testen wurde auf Daten vom MIT-Lincoln Lab(MIT-Lincoln Labs, 1999) zurück gegriffen. Die Daten sind Informationen über normale und abnormale Zustände eines Rechnernetzes. Die Daten bestanden aus Paketinformationen der Netzwerkschnittstellen(TCP-Dump), den Daten die ins Netzwerk übertragen und aus dem Netzwerk nach Außen gesendet wurden, BSM-Audit-Daten und Daten über das Dateisystem. BSM ist eine Funktion des Betriebssystem Solaris, die sämtliche Aktivitäten des Rechners protokolliert. Dazu gehören z.B. Zugriffe auf das Dateisystem, das Starten von Programmen oder die Versuche, sich in das System einzuloggen. Ein Experiment mit diesen Daten soll klären, wie gut der neue Algorithmus Störungen erkennt. Für das Experiment wurden die TCP-Dump Daten eines Rechners(hostname: marx) mit einer festgelegten Spezifikation genutzt. Für die statistische Auswertung wurden die Werte die „tcpstat“ für jede Minute lieferte genutzt. Um Verschiedene Arten von Angriffen erkennen zu können wurden drei Werte beobachtet(bytes/sec, packets/sec, ICMP/sec).
Eine Woche lang wurden diese Daten(ohne Netzwerkangriffe) dem Algorithmus zur Verfügung gestellt. In der darauf folgenden Woche wurde der Algorithmus mit Daten von internen und externen Netzwerkangriffen getestet.

Um zu sehen wie Leistungsfähig der Algorithmus ist und um ihn besser mit anderen Algorithmen vergleichen zu können wurde der Algorithmus mit folgenden Werten initialisiert:

- population_size = 200
- generations = 2000
- mutation_rate = 0.1
- sensitivity = 1.0 (sehr sensibel)
- Radien = 0.05, 0.10, 0.15, 0.2 (1 Radius pro Level)
- Durchlauf des Algorithmus pro Level = 10

Dem hier vorgeschlagenen „negative characterization with deterministic crowding“-Algorithmus, der auf eine Regelerzeugung für die Non_Self -Bereiche basiert, wurde ein Algorithmus entgegengesetzt der auf die Erkennung des $Self$ -Bereichs basiert. Dieser Algorithmus verwendete für die $Self$ -Bereichserkennung eine Abstandsmessung durch Berechnung des Euklidischen Abstands.

Nach Betrachtung der Ergebnisse stellte man fest, daß der „negative characterization with deterministic crowding“-Algorithmus mit weniger Regeln auskommt, als dies bei vergleichbaren Algorithmen der Fall ist. Zudem hat der Algorithmus bessere Regeln gefunden, da er bei vergleichbarer Leistung mit weniger Regeln auskommt. Ein weiterer Vorteil ist, daß der genetische Algorithmus nur wenige Male durchlaufen wird, um brauchbare Ergebnisse zu liefern.
Des weiteren konnte bestätigt werden das Algorithmen die eine Non_Self -Bereichserkennung verwenden in Platz und Zeit effizienter sind als Algorithmen die auf einer $Self$ - Bereichserkennung basieren. Die $Self$ - Bereichserkennung liefert bessere Ergebnisse, solange die Erkennungsbereiche klein bleiben.

Folgerung

Als Fazit lassen sich vier wichtige Folgerung erkennen.

- Der „negative characterization with deterministic crowding“-Algorithmus erkennt Störungen in einem Rechnernetz.

- Der immungenetische Algorithmus erzeugt eine gute Abschätzung in wieweit sich ein System in einem normal Zustand befindet. Es wurde gezeigt das es möglich ist mit einem „negativ selection“-Algorithmus Störungen in einem richtigen Rechnernetzwerk zu Erkennen.

- Der hier vorgestellte Algorithmus ist effizient. Vier von fünf Angriffen wurden erkannt(bei „positive characterization“). Die Erkennungsrate von Störungen liegt bei 87.5% und die Rate für Fehlalarme liegt bei einem Prozent.

- Das verwenden vom „deterministic crowding“-Algorithmus ist effizienter als der „sequential niching“-Algorithmus, da weniger und bessere Regeln gefunden werden.

Quellen

Gonzâlez, F. and Dasgupta, D. (2002).
An Immunogenetic Technique to Detect Anomalies in Network Traffic. http://www.cs.memphis.edu/~gonzalef/papers/gecco2002.pdf

Beasly, D., Bull, D. R., and Martin, R. R.(1993).
A sequential niche technique for multimodal function optimization. Evolutionary Computation,1(2):101-125.

Crosbie, M. and Spafford, E. (1995).
Applying genetic programming to intrusion detection. In Siegel, E. V. and Koza, J. R., editors, Working Notes for the AAAI Symposium on Genetic Programming, pages 1–8, MIT, Cambridge, MA, USA. AAAI.

Dagupta, D. and Gonzalez, F. (2001).
Information Assuarance in Computer Networks, chapter An intelligent decision support system for intrusion detection and response, pages 1–14. Lecture Notes in Computer Science. Springer-Verlag.

Dasgupta, D. (1999).
Artificial Immune Systems and Their Applications.
Springer-Verlag, New York.

Dasgupta, D. and Gonzalez, F. (2002).
An immunity-based technique to characterize intrusions in computer networks.
To appear in IEEE Transactions on Evolutionary Computation, 6(2).

Denning, D. (1986).
An intrusion-detection model. In IEEE Computer Society Symposium on Research in Security and Privacy, pages 118–31.

Eskin, E. (2000).
Anomaly detection over noisy data Using learned probability distributions. In Proc. 17th International Conf. on Machine Learning, pages 255–262. Morgan Kaufmann,
San Francisco, CA.

Forrest, S., Perelson, A., Allen, L., and Cherukuri, R. (1994).
Self-nonself discrimination in a computer. In Proc. IEEE Symp. On Research in Security and Privacy.

Hofmeyr, S. and Forrest, S. (2000).
Architecture for an artificial immune system. Evolutionary Computation, 8(4):443–473.

Kephart, J. (1994).
A biologically inspired immune system for computers. In Proceedings of Artificial Life,
pages 130–139, Cambridge, MA.

Kim, J. and Bentley, P. J. (2001).
An evaluation of negative selection in an artificial immune system for network intrusion detection. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001),
pages 1330–1337,
San Francisco, California, USA.
Morgan Kaufmann.

Lane, T. (2000).
Machine Learning Techniques For The Computer Security. PhD thesis,
Purdue University.

Lee, W. and Stolfo, S. (1998).
Data mining approaches for intrusion detection. In Proceedings of the 7th USENIX Security Symposium,
San Antonio, TX.

Mahfoud, S. W. (1992).
Crowding and preselection revisited. In Männer, R. and Manderick, B., editors, Parallel problem solving from nature 2,
pages 27–36, Amsterdam.
North-Holland.

MIT-Lincoln-Labs (1999).
Darpa intrusion detection evaluation.
http://www.ll.mit.edu/IST/ideval/index.html.