İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

USE OF ARTIFICIAL IMMUNE SYSTEMS FOR NETWORK INTRUSION DETECTION

M.Sc. Thesis by

Orhan BIYIKLIOĞLU, B.Sc.

(504021519)

Date of submission : 24 September 2004

Date of defence examination : 14 October 2004

Supervisor (Chairman): Prof. Dr. Bülent ÖRENCİK

Members of the Examining Committee Prof. Dr. İlhami YAVUZ (Maltepe U.)

Assist. Prof. Dr. A. Şima ETANER-UYAR

OCTOBER 2004

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY BAĞIŞIKLIK SİSTEMLERİNİN AĞ SALDIRILARININ TESPİTİ İCİN KULLANIMI

YÜKSEK LİSANS TEZİ
Müh. Orhan BIYIKLIOĞLU
(504021519)

Tezin Enstitüye Verildiği Tarih : 24 Eylül 2004
Tezin Savunulduğu Tarih : 14 Ekim 2004

Tez Danışmanı: Prof. Dr. Bülent ÖRENCİK

Diğer Jüri Üyeleri Prof. Dr. İlhami YAVUZ (Maltepe Ü.)

Yrd. Doç. Dr. A. Şima ETANER-UYAR

EKİM 2004

PREFACE

I wish to thank my supervisor Prof. Dr. Bülent ÖRENCİK for all his help and guidance during the preparation of this thesis and Assist. Prof. Dr. Şima UYAR for introducing me with the topic of artificial immune systems, which I enjoyed to work on during this thesis.

I would also like to thank to all my friends for their valuable support and for their friendship.

My special thanks are due to all my family, whose great support, encouragement and patience brought me today.

Last but not least, my fiancé Fehime TÜFEKÇİOĞLU deserves special thanks for her support and motivation, as well as her great editorial help throughout this study.

September 2004 **Orhan BIYIKLIOĞLU**

TABLE OF CONTENTS

ABBREVIATIONS

AIS	: Artificial Immune System
BSM	: Basic Security Module
CDIS	: Computer Defense Immune System
CORBA	: Common Object Request Broker Architecture
CT	: Central Tolerization
CVIS	: Computer Virus Immune System
DARPA	: The Defense Advanced Research Projects Agency
DNA	: Deoxyribonucleic Acid
DOS	: Denial of Service
DT	: Distributed Tolerization
EA	: Evolutionary Algorithm
GA	: Genetic Algorithm
HTML	: Hypertext Markup Language
HTTP	: Hypertext Transfer Protocol
ICMP	: Internet Control Message Protocol
ID	: Intrusion Detection
IDIS	: Intrusion Detection Immune System
IDS	: Intrusion Detection System
IDEVAL	: IDS Off-line Evaluation
IS	: (Natural-Biological-Human) Immune System
IP	: Internet Protocol
LAN	: Local Area Network
LRU	: Least Recently Used
MHC	: Major Histocompatibility Complex
NN	: Neural Network
OS	: Operating System
R2L	: Remote-to-local Attack
SDM	: Sparse Distributed Memory
SNMP	: Simple Network Management Protocol
TCP	: Transmission Control Protocol
U2R	: User-to-root Attack
UDP	: User Datagram Protocol

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

ℓ	: String length
r	: Match threshold
U	: Universe
S	: Self
N	: Nonself
Λ	: Representation function
D	: Detection System
ε^+	: False positive error
ε^-	: False negative error
h	: Match Rule
c_d	: Cover set of detector d
n_d	: Number of detectors
m_d	: Maximum number of memory detectors
T	: Tolerization period
T_s	: Costimulation delay
w	: Effect of sensitivity
ρ	: Number of retries in deterctor generation
p_{death}	: Probability of detector death
p_M	: Probability of match

YAPAY BAĞIŞIKLIK SİSTEMLERİNİN AĞ SALDIRILARININ TESPİTİ İÇİN KULLANIMI

ÖZET

Bilgisayarların kritik sistemlerde kullanımının artmasının bir sonucu olarak bilgisayar güvenliği -verilerin ve bilgisayar sistemlerinin kasıtlı ve kötü niyetli müdahalelerden koruma sanatı ve bilimi- giderek daha çok ilgi çeken bir konu haline gelmektedir. Bu amaçla parolalar, güvenlik duvarları, şifreleme ve sayısal imza gibi pek çok savunma yöntemi zaten mevcuttur. Ancak saldırganların er geç sistemlere girmelerini engellemek hala çok zordur. Şu anda güvenlik ihlallerini tamamen engellemek imkansız gözükmektedir. Fakat sistemlere girme girişimlerinin veya başarılı sızmaların yakalanmaya çalışılması biraz daha fazla güvenlik sağlanmasına katkıda bulunabilir. Bu amaca yönelik araştırma konusuna Saldırı Tespiti (ST) adı verilir.

Bu konu uzun yıllardır araştırılmaktadır ve Saldırı Tespit Sistemleri (STS) geliştirmek amacıyla pek çok değişik yaklaşım denenmiştir. Saldırı Tespit Sistemleri iki farklı saldırı tespit yönteminden birini kullanırlar. Bu yöntemler kötüye kullanım tespiti ve anormallik tespitidir. Kötüye kullanım tespiti yaklaşımı, bilinen sistem zayıflıkları ve sistem güvenlik politikalarına dayanarak muhtemel kötüye kullanımların imzalarını oluşturur. Daha sonra bu imzaların sistemde izlenen veriler içerisinde geçip geçmediğine bakarak muhtemel saldırıları yakalamaya çalışır. Anormallik tespiti yaklaşımı ise bir işletim sistemi veya ağ trafiği kayıt yazılımı tarafından üretilen verilerden sistemdeki kullanıcıların, sistem kaynaklarının, ağ trafiğinin veya servislerin normal davranış kesitlerini çıkartarak bu kesitlerle tanımlanan normal davranış kalıplarından sapmaları saldırı olarak tespit eder.

Her iki yönteminde güçlü ve zayıf tarafları vardır. Ancak anormallik tespiti yapan sistemlerin bir takım önemli kısıtlarından ötürü mevcut pek çok ticari STS kötüye kullanım tespiti yöntemini kullanmaktır. Bu da saldırı tespiti amaçlı etkin anormallik tespiti sistemlerinin geliştirilmesi için araştırma yapılmasını teşvik etmektedir. Anormallik tespiti için pek çok farklı yöntem denenmiştir. Bunlardan bazıları veri madenciliği, genetik algoritmalar, yapay sinir ağları, istatistiksel analiz yöntemleri ve yapay bağışıklık sistemleri yaklaşımlarıdır.

Yapay bağışıklık sistemleri (YBS) insan bağışıklık sisteminin kuramsal ve gözlemsel işlevlerinden, ilke ve modellerinden esinlenen ve bu mekanizmaları çeşitli problem alanlarına uygulayan uyarlanabilir sistemler olarak tanımlanırlar. Bu araştırma

konusu görece yeni olmasına karşın pek çok konuya uygulanmıştır. Bu sistemlerin en yaygın uygulandığı konulardan biri bilgisayar güvenliği konusudur. Bilgisayar güvenliği sistemlerinin amaçları ile insan bağışıklık sisteminin amaçları arasındaki benzerlikler doğal olarak araştırmacıları bu ikisi arasındaki benzerlikleri araştırmaya yönlendirmektedir.

Bu tez çalışması yapay bağışıklık sistemi yaklaşımları ve onların saldırı tespitine uygulanması üzerine odaklanmıştır. Yapay bağışıklık sistemi yaklaşımlarından birisi olan negatif seçim algoritması uygulanmış ve DARPA/Lincoln Laboratuarı 1999 STS değerlendirme veri setleri üzerinde denenmiştir. Bu algoritmanın başarımı ve anormallik tespit yetenekleri araştırılmış ve bu araştırmaların sonuçları sunulmuştur.

USE OF ARTIFICIAL IMMUNE SYSTEMS FOR NETWORK INTRUSION DETECTION

SUMMARY

As a consequence of the increased use of computers for critical systems, computer security, which is the art and science of protecting data and computer systems from intentional, malicious intervention, is attracting increasing attention. Many methods of defence already exist such as passwords, firewalls, encryption and digital signatures. However, it is still very difficult to keep the attackers away from entering the systems eventually. Currently, completely preventing breaches of security seems to be impossible. However, trying to detect intrusion attempts or possible intrusions to the systems may be useful to achieve a higher level of security. This field of research is called Intrusion Detection (ID).

The field has existed for some years, and many different approaches are made to build Intrusion Detection Systems (IDSs) to detect attacks or intrusions to computer systems. Intrusion Detection Systems employ one of two Intrusion Detection methods: misuse or anomaly detection. The misuse detection approach defines suspicious misuse signatures based on known system vulnerabilities and a security policy. This approach checks whether these misuse signatures are present or not in the monitored data. The anomaly detection approach establishes the profiles of normal activities of users, systems or system resources, network traffic and services using the audit trails generated by a host operating system or a network-scanning program. This approach detects intrusions by identifying significant deviations from the normal behaviour patterns of profiles.

Both of these approaches have some strengths and limitations. However, most available commercial IDSs use only misuse detection because most developed anomaly detectors still cannot overcome some important limitations. This motivates many research efforts to build effective anomaly detectors for the purpose of intrusion detection. There has been various different approaches to anomaly detection including but not limited to data mining, genetic algorithms, neural networks, statistical analysis and artificial immune systems.

Artificial immune systems (AIS) have been defined as adaptive systems inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving. Although this research field is relatively new, it has already been applied to many research areas. One of the areas that AIS have

been most frequently applied to is the computer security. The similarity between the goal of a computer security system and the human immune system naturally leads researchers to examine analogies between the two.

This thesis focuses on the artificial immune systems approaches and their application to intrusion detection. Negative Selection algorithm, one of the artificial immune system approaches, is implemented and tested on the DARPA/Lincoln Laboratory 1999 IDS evaluation data sets. Its performance and detection capabilities are investigated, and the results are provided.

1. INTRODUCTION

The Internet has become a critical infrastructure for the modern society. The explosive growth of Internet users has motivated the rapid expansion of electronic commerce and other online-based services. These services provide great convenience and efficiency for the users. On the other side vulnerability to cyber threats makes network security a major concern. The wide availability of the information and tools needed to penetrate the security of corporate networks increases that concern. This increasing importance of computer security motivates various aspects of security related research that provide new solutions, which might not be achievable by more conventional security approaches.

When a private local network is connected to the Internet, it is opened to many online threats as well as great opportunities for information sharing. As more and more sensitive information is being stored in computer systems and transferred over computer networks, attacks to steal destroy or corrupt this information is also increasing. While most computer systems attempt to prevent unauthorized use by some kind of access control mechanism, such as passwords, firewalls, encryption and digital signatures, it is still very difficult to keep the attackers away from entering the systems eventually. Currently, completely preventing breaches of security seems to be impossible. However, trying to detect intrusion attempts or possible intrusion to the systems may be useful to achieve more security. This field of research is called Intrusion Detection (ID).

Traditionally, tools and methods used in the field of computer system security are created or implemented to make systems secure and protected against attacks. However, because of the increasing complexities of the systems, errors in configuring or administering the security products, or insider attacks by authorized users of the systems who attempt to gain additional privileges for which they are not authorized, making this systems invulnerable to attack becomes impossible. Because of this, the importance of intrusion detection is increasing.

The research in the field of intrusion detection dates back to 1980 and since then ID research has been very active. The researches started in 1980 and John Anderson's Computer Security Threat Monitoring and Surveillance, published in this year was one of the earliest papers in the field. Later in 1987 Dorothy Denning's, "An

Intrusion Detection Model," was published and this paper was very influential on the following researches. It also provided the basic methodologies of many commercial ID products. Although, significant amount of research is done in this field, ID technology is still immature and it could not be a complete defense mechanism against increasingly frequent and complicated attacks. [1] But, it is one of the main security components and very important for protecting computer systems from losses associated with network security problems.

Intrusion detection systems (IDSs) can be described as software or hardware systems that automate the process of monitoring the events occurring in a computer system or network, analyzing them for signs of security problems. As network attacks have increased in number and severity over the past few years, intrusion detection systems have become a necessary addition to the security infrastructure of most organizations. Intrusion Detection Systems can be seen as the computer equivalent of burglar alarms. They are used to monitor computer systems for detecting attacks and intrusions.

Intrusion detection allows organizations to protect their systems from the threats that come with increasing network connectivity and reliance on information systems. IDSs are becoming one of the main components of computer and network security and have already gained acceptance as a necessary addition to every organization's security infrastructure.

There are various approaches to building intrusion detection systems. They mainly employ two techniques: anomaly detection and misuse detection. The anomaly detection approach establishes the profiles of normal activities of users, systems or system resources, network traffic and services using the audit trails generated by a host operating system or a network-scanning program. This approach detects intrusions by identifying significant deviations from the normal behavior patterns of profiles. The misuse detection approach defines suspicious misuse signatures based on known system vulnerabilities and a security policy. This approach probes whether these misuse signatures are present or not in the auditing trails. Both of these approaches have strengths and limitations, which will be discussed later on.

This thesis addresses anomaly detection approach and specifically the application of natural immune system (IS) mechanisms for network intrusion detection. Human immune systems have been successful at protecting the human body against a vast variety of foreign pathogens or organisms. One most interesting feature of the human immune system is its power to detect harmful pathogens without attacking human body self cells. The human immune system distinguishes previously known and

unknown pathogens from human body self cells via its own evolutionary mechanism, which is similar to evolution of organisms in nature. These features of the natural immune system motivated computer scientist to study this natural mechanism and propose artificial immune models for solving various problems.

Artificial immune systems (AIS) is a biologically inspired computational intelligence paradigm such as artificial neural networks, evolutionary algorithms and genetic algorithms. Although it is relatively newer than these approaches, it has already been applied to many research areas (see section 3.2.6). One of the areas that AIS have been most frequently applied to is the computer security. The similarity between the goal of a computer security system and the human immune system naturally leads researchers to examine analogies between the two.

The main goal of this thesis is to investigate the application of artificial immune systems to network intrusion detection, implement an artificial immune system based network intrusion detection mechanism and evaluate it with a commonly used dataset.

The next chapter gives an overview of intrusion detection systems necessary for an understanding of the work that follows. Different ID approaches and features of the IDSs are summarized.

Chapter 3 presents a review of relevant immunology that creates the foundations for work presented in this thesis. Firstly, the natural immune system is reviewed and than an overview of the field of Artificial Immune Systems which consists of the applied immune mechanisms and the previous work, is given. Immunology is a very large topic. Therefore, only the relevant topics are discussed.

Chapter 4 examines the proposed architecture for the application of Artificial Immune Systems for distributed anomaly detection task and provides an outline of the architecture along with the natural mechanisms that inspired each property.

Chapter 5 investigates the application of the proposed model for the specific task of network intrusion detection and its evaluation.

Chapter 6 provides the evaluation results of the system and their analysis.

The last chapter concludes and summarizes the work in this thesis.

2. OVERVIEW OF INTRUSION DETECTION

This section gives an overview of the intrusion detection process, the need for intrusion detection, methods used in ID and general types of IDSs. It provides a preliminary knowledge about the topic so the reader can understand the work in this thesis.

2.1 Definition of intrusion detection

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network. Intrusions are caused by attackers accessing the systems from the Internet, authorized users of the systems who attempt to gain additional privileges for which they are not authorized, and authorized users who misuse the privileges given them. Intrusion Detection Systems (IDSs) are software or hardware products that automate this monitoring and analysis process.

Intrusion detection systems help computer systems prepare for and deal with attacks. They accomplish this goal by collecting information from a variety of system and network sources, then analyzing the information for symptoms of security problems. In some cases, intrusion detection systems allow the user to specify real-time responses to the violations.

Intrusion detection systems perform a variety of functions [2]:

- Monitoring and analysis of user and system activity
- Auditing of system configurations and vulnerabilities
- Assessing the integrity of critical system and data files
- Recognition of activity patterns reflecting known attacks
- Statistical analysis for abnormal activity patterns

- Operating system audit trail management, with recognition of user activity reflecting policy violations

The combination of these features allows system managers to more easily handle the monitoring, audit, and assessment of their systems and networks. This ongoing assessment and audit activity is a necessary part of sound security management practice.

Intrusion detection techniques can be divided into two main categories:

2.1.1 Misuse Detection

The concept behind misuse detection schemes is that there are ways to represent attacks in the form of a pattern or a signature so that even variations of the same attack can be detected. This means that these systems are like virus detection systems (they can detect many or all known attack patterns, but they are of little use for as yet unknown attack methods). An interesting point to note is that anomaly detection systems try to detect the complement of "bad" behavior. Misuse detection systems try to recognize known "bad" behavior. The main issues in misuse detection systems are how to write a signature that encompasses all possible variations of the pertinent attack, and how to write signatures that do not also match non-intrusive activity.

2.1.2 Anomaly Detection

Anomaly detection techniques assume that all intrusive activities are anomalous. This means that if we could establish a "normal activity profile" for a system, we could, in theory, indicate all system states varying from the established profile by statistically significant amounts as intrusion attempts. However, if we consider that the set of intrusive activities only intersects the set of anomalous activities instead of being exactly the same, we find a couple of possibilities: (1) Identification of anomalous activities that are not intrusive as intrusive result in false positives. (2) Intrusive activities that are not anomalous result in false negatives (events are not identified intrusive, though they actually are). This is a dangerous problem, and is far more serious than the problem of false positives. The main issues in anomaly detection systems thus become the selection of threshold levels so that neither of the above two problems is unreasonably magnified, and the selection of features to monitor. Anomaly detection systems are also computationally expensive because of the overhead of keeping track of, and possibly updating several system profile metrics.

2.2 Reasons to use Intrusion Detection Systems

Intrusion detection allows organizations to protect their systems from the threats that come with increasing network connectivity and reliance on information systems. IDSs have gained acceptance as a necessary addition to every organization's security infrastructure. There are several reasons to use IDSs as a security component:

1. To deter attackers or abusers by means of increased risk of discovery and punishment,
2. To detect attacks and other security violations that are not prevented by other security measures,
3. To detect and deal with the preambles to attacks,
4. To document the existing threat to an organization,
5. To act as quality control for security design and administration, especially of large and complex enterprises,
6. To provide useful information about intrusions that do take place, allowing improved diagnosis, recovery, and correction of causative factors.

2.2.1 Preventing problems by determent

A fundamental goal of computer security management is to affect the behavior of individual users in a way that protects information systems from security problems. Intrusion detection systems help organizations accomplish this goal by increasing the perceived risk of discovery and punishment of attackers. This serves as a significant deterrent to those who would violate security policy.

2.2.2 Detecting problems that are not prevented by other security measures

Attackers, using widely publicized techniques, can gain unauthorized access to many, if not most systems, especially those connected to public networks. This often happens when known vulnerabilities in the systems are not corrected. Although vendors and administrators are encouraged to address vulnerabilities in case they enable attacks, there are many situations in which this is not possible:

- In many legacy systems, the operating systems cannot be patched or updated.

- Even in systems in which patches can be applied, administrators sometimes have neither sufficient time nor resource to track and install all the necessary patches. This is a common problem, especially in environments that include a large number of hosts or a wide range of different hardware or software environments.

- Users can have compelling operational requirements for network services and protocols that are known to be vulnerable to attack.

- Both users and administrators make errors in configuring and using systems.

- In configuring system access control mechanisms to reflect an organization's procedural computer use policy, inconsistencies almost always occur. These inconsistencies allow legitimate users to perform actions that are ill advised or that overstep their authorization.

In an ideal world, commercial software vendors would minimize vulnerabilities in their products, and user organizations would correct all reported vulnerabilities quickly and reliably. However, in the real world, this seldom happens.

Therefore, intrusion detection can represent an excellent approach to protecting a system. An IDS can detect when an attacker has penetrated a system by exploiting an uncorrected or uncorrectable flaw. Furthermore, it can serve an important function in system protection, by bringing the fact that the system has been attacked to the attention of the administrators who can contain and recover any damage that results. This is far preferable to simply ignoring network security threats where one allows the attackers continued access to systems and the information on them.

2.2.3 Detecting the preambles to attacks

When adversaries attack a system, they typically do so in predictable stages. The first stage of an attack is usually probing or examining a system or network, searching for an optimal point of entry. In systems with no IDS, the attacker is free to thoroughly examine the system with little risk of discovery or retribution. Given this free access, a determined attacker will eventually find vulnerability in such a network and exploit it to gain entry to various systems.

The same network with an IDS monitoring its operations presents a much more difficult challenge to that attacker. Although the attacker may probe the network for weaknesses, the IDS will observe the probes, will identify them as suspicious, may actively block the attacker's access to the target system, and will alert security

personnel who can then take appropriate actions to block subsequent access by the attacker. Even the presence of a reaction to the attacker's probing of the network will elevate the level of risk the attacker perceives, discouraging further attempts to target the network.

2.2.4 Documenting the existing threat

Understanding the frequency and characteristics of attacks allows understanding what security measures are appropriate to protect the network against those attacks. IDSs verify, itemize, and characterize the threat from both outside and inside an organization's network, assisting in making sound decisions regarding the allocation of computer security resources. Using IDSs in this manner is important, as many people mistakenly deny that anyone (outsider or insider) would be interested in breaking into their networks. Furthermore, the information that IDSs give regarding the source and nature of attacks allows making decisions regarding security strategy driven by demonstrated need, not guesswork.

2.2.5 Quality control for security design and administration

When IDSs run over a period of time, patterns of system usage and detected problems can become apparent. These can highlight flaws in the design and management of security for the system, in a way that supports security management correcting those deficiencies before they cause an incident.

2.2.6 Providing useful information about actual intrusions

Even when IDSs are not able to block attacks, they can still collect relevant, detailed, and trustworthy information about the attack that supports incident handling and recovery efforts. Furthermore, this information can, under certain circumstances, enable and support criminal or civil legal remedies. Ultimately, such information can identify problem areas in the organization's security configuration or policy.

2.3 Classification of IDSs

There are several types of IDSs available today, characterized by different monitoring and analysis approaches. Each approach has distinct advantages and disadvantages. Furthermore, all approaches can be described in terms of a generic process model for IDSs.

2.3.1 Process model for Intrusion Detection

Many IDSs can be described in terms of three fundamental functional components:

- Information Sources - the different sources of event information used to determine whether an intrusion has taken place. These sources can be drawn from different levels of the system, with network, host, and application monitoring most common.

- Analysis - the part of intrusion detection systems that actually organizes and makes sense of the events derived from the information sources, deciding when those events indicate that intrusions are occurring or have already taken place. The most common analysis approaches are misuse detection and anomaly detection.

- Response - the set of actions that the system takes once it detects intrusions. These are typically grouped into active and passive measures, with active measures involving some automated intervention on the part of the system, and passive measures involving reporting IDS findings to humans, who are then expected to take action based on those reports.

2.3.2 Major IDS characteristics

2.3.2.1 Monitoring Approach

The most common way to classify IDSs is to group them by information source. Some IDSs analyze network packets, captured from network backbones or LAN segments, to find attackers. Other IDSs analyze information sources generated by the operating system or application software for signs of intrusion.

Network-Based IDSs

The majority of commercial intrusion detection systems are network based. These IDSs detect attacks by capturing and analyzing network packets. Listening on a network segment or switch, one network-based IDS can monitor the network traffic affecting multiple hosts that are connected to the network segment, thereby protecting those hosts.

Network-based IDSs often consist of a set of single-purpose sensors or hosts placed at various points in a network. These units monitor network traffic, performing local analysis of that traffic and reporting attacks to a central management console. As the

sensors are limited to running the IDS, they can be more easily secured against attack. Many of these sensors are designed to run in "stealth" mode, in order to make it more difficult for an attacker to determine their presence and location.

Advantages of Network-Based IDSs:

- A few well-placed network-based IDSs can monitor a large network.
- The deployment of network-based IDSs has little impact upon an existing network. Network-based IDSs are usually passive devices that listen on a network wire without interfering with the normal operation of a network. Thus, it is usually easy to retrofit a network to include network-based IDSs with minimal effort.
- Network-based IDSs can be made very secure against attack and even made invisible to many attackers.

Disadvantages of Network-Based IDSs:

- Network-based IDSs may have difficulty processing all packets in a large or busy network and, therefore, may fail to recognize an attack launched during periods of high traffic. Some vendors are attempting to solve this problem by implementing IDSs completely in hardware, which is much faster. The need to analyze packets quickly also forces vendors to both detect fewer attacks and also detect attacks with as little computing resource as possible, which can reduce detection effectiveness.
- Many of the advantages of network-based IDSs don't apply to more modern switch-based networks. Switches subdivide networks into many small segments (usually one fast Ethernet wire per host) and provide dedicated links between hosts serviced by the same switch. Most switches do not provide universal monitoring ports and this limits the monitoring range of a network-based IDS sensor to a single host. Even when switches provide such monitoring ports, often the single port cannot mirror all traffic traversing the switch.
- Network-based IDSs cannot analyze encrypted information. This problem is increasing as more organizations (and attackers) use virtual private networks.
- Most network-based IDSs cannot tell whether or not an attack was successful; they can only detect that an attack was initiated. This means that after a

network-based IDS detects an attack, administrators must manually investigate each attacked host to determine whether it was indeed penetrated.

- Some network-based IDSs have problems dealing with network based attacks that involve fragmenting packets. These malformed packets cause the IDSs to become unstable and crash.

Host-Based IDSs

Host-based IDSs operate on information collected from within an individual computer system. (Note that application-based IDSs are actually a subset of host-based IDSs.) This detection point allows host based IDSs to analyze activities with great reliability and precision, determining exactly which processes and users are involved in a particular attack on the operating system. Furthermore, unlike network based IDSs, host-based IDSs can "see" the outcome of an attempted attack, as they can directly access and monitor the data files and system processes usually targeted by attacks.

Host-based IDSs normally use information sources of two types, operating system audit trails, and system logs. Operating system audit trails are usually generated at the innermost (kernel) level of the operating system, and are therefore more detailed and better protected than system logs. However, system logs are much less easy to use and much smaller than audit trails, and are furthermore far easier to understand. Some host-based IDSs are designed to support a centralized IDS management and reporting infrastructure that can allow a single management console to track many hosts. Others generate messages in formats that are compatible with network management systems.

Advantages:

- Host-based IDSs, with their ability to monitor events local to a host, can detect attacks that cannot be seen by a network-based IDS.

- Host-based IDSs can often operate in an environment in which network traffic is encrypted, when the host-based information sources are generated before data is encrypted and/or after the data is decrypted at the destination host

- Host-based IDSs are unaffected by switched networks.

- When Host-based IDSs operate on OS audit trails, they can help detect Trojan horse or other attacks that involve software integrity breaches. These appear as inconsistencies in process execution.

Disadvantages:

- Host-based IDSs are harder to manage, as information must be configured and managed for every host monitored.
- Since at least the information sources (and sometimes part of the analysis engines) for host-based IDSs reside on the host targeted by attacks, the IDS may be attacked and disabled as part of the attack.
- Host-based IDSs are not well suited for detecting network scans or other such inspection activities that targets an entire network, because the IDS only sees those network packets received by its host.
- Host-based IDSs can be disabled by certain denial-of-service attacks.
- When host-based IDSs use operating system audit trails as an information source, the amount of information can be immense, requiring additional local storage on the system.
- Host-based IDSs use the computing resources of the hosts they are monitoring, therefore inflicting a performance cost on the monitored systems.

Application-Based IDSs

Application-based IDSs are a special subset of host-based IDSs that analyze the events occuring within a software application. The most common information sources used by application-based IDSs are the application's transaction log files.

The ability to interface with the application directly, with significant domain or application-specific knowledge included in the analysis engine, allows application-based IDSs to detect suspicious behavior due to authorized users exceeding their authorization. This is because such problems are more likely to appear in the interaction between the user, the data, and the application.

Advantages:

- Application-based IDSs can monitor the interaction between user and application, which often allows them to trace unauthorized activity to individual users.

- Application-based IDSs can often work in encrypted environments, since they interface with the application at transaction endpoints, where information is presented to users in unencrypted form.

Disadvantages:

- Application-based IDSs may be more vulnerable than host-based IDSs to attacks as the application logs are not as well protected as the operating system audit trails used for host-based IDSs.

- As Application-based IDSs often monitor events at the user level of abstraction, they usually cannot detect Trojan Horse or other such software tampering attacks. Therefore, it is advisable to use an Application-based IDS in combination with Host-based and/or Network-based IDSs.

Target-Based Approaches

This approach monitors specific files, system objects and system object attributes for change, looking at the outcome of attack processes rather than the details of the attack processes. Some systems use checksums (computations whose value depends on the original constitution of the system object) to detect breaches of integrity.

Advantages:

- Because it does not depend on historical records of behavior, integrity analysis may detect intrusions that other methodologies do not;

- This approach allows reliable detection of both placement and presence of attacks that modify the system (e.g., Trojan horses);

- Because its footprints and intrusiveness are low, this approach can be useful for monitoring systems with modest processing or communications bandwidth;

- This approach is effective for determining which files need to be replaced in order to recover a system, rather than reinstalling everything from the original source or backup, as is often done.

Disadvantages:

- Depending on the number of files, system objects and object attributes for which checksums are computed, this approach may still impose an appreciable processing load on low-end systems;
- The approach is not well suited to real-time detection processes, as it monitors for the outcome of attacks, not for the attacks themselves while they are in progress.

Integrated approaches

Some intrusion detection products combine application, host, and network-based sensors.

Advantages:

- As agents at applications, host, and network levels are used, the system can target activity at any or all levels.
- It is easier to see patterns of attacks over time and across the network space; this is of value in damage assessment and system recovery; it also aids in investigating the incident and pursuing legal remedies (e.g. criminal prosecutions).

Disadvantages:

- There are no industry standards with regards to interoperability of intrusion detection components; therefore it is difficult or impossible to integrate components from different vendors.
- Integrated systems are more difficult to manage and deploy.

2.3.2.2 Timing of Information Collection and Analysis

Timing refers to the elapsed time between the events that are monitored and the analysis of those events.

Interval-Based (Batch Mode)

In interval-based IDSs, the information flow from monitoring points to analysis engines is not continuous.

Many early host-based IDSs used this timing scheme, as they relied on operating system audit trails, which were generated as files. Interval based IDSs are not able to perform active responses.

Advantages:

- They are well suited to environments in which threat levels are low and single-attack loss potentials high (e.g., financial institutions). In these environments, users are often more interested in establishing accountability for problems than immediately responding to suspected incidents. In this situation, batch-oriented analysis will likely be combined with other investigative process in order to identify the person responsible for the incident and support criminal prosecution for the incident.

- Batch mode analysis schemes impose less processing load on systems than real-time analysis, especially when collection intervals are short and data volumes are therefore low.

- Batch-oriented collection and analysis of information are particularly well suited to organizations in which system and personnel resources are limited. Organizations that have no full-time security personnel may find that real-time alarms generated by intrusion detection systems are seldom used. In such circumstances, it makes little sense to tolerate the processing load associated with real-time analysis and alarms.

- Attacks on computer systems often involve repetitive attacks on the same targets. For example, an attacker may enter a system via a password-grabbing attack, then install a Trojan horse "back door" in order to return later and continue the attack. Batch-mode analysis can usually recognize such attack signatures.

- Many current legal practices relating to computer evidence were established with batch-mode collection and manual analysis in mind. Therefore, it may be easier to submit system logs collected and processed in batch mode as evidence.

Disadvantages:

- Users will seldom see incidents before they are complete.
- Therefore, there is virtually no possibility of actively countering incidents as they happen in an attempt to minimize damage.
- The aggregation of information for batch-mode analysis consumes more disk storage on the analysis system. This can result in huge amounts of data for enterprise networks.

Real-Time (Continuous)

Real-time IDSs operate on continuous information feeds from information sources. This is the predominant timing scheme for network based IDSs, which gather information from network traffic streams. Detection performed by a "real-time" IDS yields results quickly enough to allow the IDS to take action that affects the progress of the detected attack.

Advantages:

- Depending on the speed of the analysis, attacks may be detected quickly enough to allow system administrators to interrupt them;
- Depending on the speed and sensitivity of the analysis, system administrators may be able to perform incident handling (leading to recovery of system operations) more quickly;
- In cases that occur on systems where legal remedies are available, system administrators may be able to collect information that allows more effective identification and prosecution of intruders.

Disadvantages:

- They tend to consume more memory and processing resource on the analysis system than post facto systems;
- Configuration of real-time systems is critical; a badly formed signature can generate so many false alarms that a real attack goes unnoticed.

2.3.2.3 Location of Analysis

As in sensors, analysis functions can reside at host level, at network-level, or both. Performing analysis strictly at the host level has the advantage of minimizing network load. However, it has the disadvantage of not allowing the detection of

broad scale attacks targeting a network of machines (for instance, an attacker sequentially hopping through a network performing brute force password guessing against each host).

Combining raw data and performing analysis strictly at the network level (in the case of systems with sensors at both host and network levels) offer the capability to detect attacks that involve more than one host on the network. The disadvantage to this approach is that the network load associated with transferring raw host-level information to the analysis engine can bring a heavy load to the network.

As in sensor placement, the optimal strategy for performing analysis of logs is one in which analysis is done at both host and network levels. The analysis done at the host level can be simple or extensive depending on the nature of the sensor information generated in that host or the signature against which the information is matched. The network-level analysis can take the results from the host-level analysis and use it to detect signs of network-wide attack or suspicious behavior without incurring as heavy a network load. Furthermore, in larger networks, this sort of approach can be applied hierarchically. That is, groups of hosts can report to a network analysis engine, which in turn reports its results to another analysis engine that collects results from a number of other network analysis engines and so on. This hierarchical structure lets intrusion-detection products succeed even in larger organizations.

Host-Target Co-location

In early days of IDSs, most IDSs ran on the systems they protected. This was due to the fact that most systems were mainframe systems, and the cost of computers made a separate IDS system a costly expense. This presented a problem from a security point of view, as any attacker that successfully attacked the target system could simply disable the IDS as an integral portion of the attack.

Host-Target Separation

With the advent of workstations and personal computers, most IDS architects moved towards running the IDS control and analysis systems on a separate system, hence separating the IDS host and target systems. This improved the security of the IDS as this made it much easier to hide the existence of the IDS from attackers.

2.3.2.4 Types of Analysis

Signature analysis

Signatures are patterns corresponding to known attacks or misuses of systems. They may be simple (character string matching looking for a single term or command) or complex (security state transition written as a formal mathematical expression). In general a signature can be concerned with a process (the execution of a particular command) or an outcome (the acquisition of a root shell.)

Signature analysis is pattern matching of system settings and user activities against a database of known attacks. Most commercial intrusion detection products perform signature analysis against a vendor-supplied database of known attacks. Additional signatures specified by the customer can also be added as part of the intrusion detection system configuration process. Most vendors also include periodic updates of signature databases as part of software maintenance agreements.

One advantage of signature analysis is that it allows sensors to collect a more tightly targeted set of system data, thereby reducing system overhead. Unless signature databases are unusually large (say hundreds of thousands or millions of complex signatures), signature analysis is usually more efficient than statistical analysis due to the absence of floating point computations.

Statistical analysis

Statistical analysis finds deviations from normal patterns of behavior. This feature, common in research settings, is found in few commercial intrusion detection products. Statistical profiles are created for system objects (e.g., users, files, directories, devices, etc.) by measuring various attributes of normal use (e.g., number of accesses, number of times an operation fails, time of day, etc.). Mean frequencies and measures of variability are calculated for each type of normal usage. Possible intrusions are signaled when observed values fall outside the normal range. For example, statistical analysis might signal an unusual event if an accountant who had never previously logged into the network outside the hours of 8 AM to 6 PM were to access the system at 2 AM.

The advantages of statistical analysis are:

- The system may detect heretofore-unknown attacks;
- Statistical methods may allow one to detect more complex attacks, such as those that occur over extended periods.

Disadvantages of statistical analysis (at this time) are:

- It is relatively easy for an attacker to trick the detector into accepting attack activity as normal by gradually varying behavior over time;
- The possibility of false alarms is much greater in statistical detectors;
- Statistical detectors do not deal well with changes in user activities. This rigidity can be a problem in organizations where change is frequent. This can result in both false alarms and false negatives (missed attacks).

Integrity analysis

Integrity analysis focuses on whether some aspect of a file or object has been altered. This often includes file and directory attributes, content and data streams. Integrity analysis often utilizes strong cryptographic mechanisms, called message digest (or hash) algorithms, which can recognize even subtle changes.

Advantages:

- Any successful attack where files were altered, network packet grabbers were left behind, or rootkits were deployed will be detected regardless of whether or not the attack was detected by signature or statistical analysis.

Disadvantages:

- Because current implementations tend to work in batch mode, they are not conducive to real-time response.

2.3.2.5 Responses to Detection of Misuse or Attack

Once IDSs have obtained event information and analyzed it to find symptoms of attacks, they generate responses. Some of these responses involve reporting results and findings to a pre-specified location. Others involve more active automated responses. Though researchers are tempted to underrate the importance of good response functions in IDSs, they are actually very important. Commercial IDSs support a wide range of response options, often categorized as active responses, passive responses, or some mixture of the two.

Active Responses

Active IDS responses are automated actions taken when certain types of intrusions are detected. There are three categories of active responses.

Collect Additional Information

The most harmless, but at times most productive, active response is to collect additional information about a suspected attack.

In the IDS case, this might involve increasing the level of sensitivity of information sources (for instance, turning up the number of events logged by an operating system audit trail, or increasing the sensitivity of a network monitor to capture all packets, not just those targeting a particular port or target system.) Collecting additional information is helpful for several reasons. The additional information collected can help resolve the detection of the attack (assisting the system in diagnosing whether an attack did or did not take place). This option also allows the organization to gather information that can be used to support investigation and capturing of the attacker, and to support criminal and civil legal remedies.

Change the Environment

Another active response is to halt an attack in progress and then block subsequent access by the attacker. Typically, IDSs do not have the ability to block a specific person's access, but instead block Internet Protocol (IP) addresses from which the attacker appears to be coming. It is very difficult to block a determined and knowledgeable attacker, but IDSs can often deter expert attackers or stop novice attackers by taking the following actions:

- Injecting TCP reset packets into the attacker's connection to the victim system, thereby terminating the connection
- Reconfiguring routers and firewalls to block packets from the attacker's apparent location (IP address or site),
- Reconfiguring routers and firewalls to block the network ports, protocols, or services being used by an attacker, and
- In extreme situations, reconfiguring routers and firewalls to cut off all connections that use certain network interfaces.

Passive Responses

Passive IDS responses provide information to system users, relying on humans to take subsequent action based on that information. Many commercial IDSs rely solely on passive responses.

Alarms and Notifications

Alarms and notifications are generated by IDSs to inform users when attacks are detected. Most commercial IDSs allow users a great deal of latitude in determining how and when alarms are generated and to whom they are displayed.

The most common form of alarm is an onscreen alert or popup window. This is displayed on the IDS console or on other systems as specified by the user during the configuration of the IDS. The information provided in the alarm message varies widely, ranging from a notification that an intrusion has taken place to extremely detailed messages outlining the IP addresses of the source and target of the attack, the specific attack tool used to gain access, and the outcome of the attack.

Another set of options that are of utility to large or distributed organizations are those involving remote notification of alarms or alerts. These allow organizations to configure the IDS so that it sends alerts to cellular phones and pagers carried by incident response teams or system security personnel.

Some products also offer email as another notification channel. This is ill advised, as attackers often routinely monitor email and might even block the message.

SNMP Traps and Plug-ins

Some commercial IDSs are designed to generate alarms and alerts, reporting them to a network management system. These use SNMP traps and messages to post alarms and alerts to central network management consoles, where they can be serviced by network operations personnel. Several benefits are associated with this reporting scheme, including the ability to adapt the entire network infrastructure to respond to a detected attack, the ability to shift the processing load associated with an active response to a system other than the one being targeted by the attack, and the ability to use common communications channels.

2.3.2.6 Management Functions and Deployment Issues

People need flexibility in adapting intrusion detection systems to their own environments. The following features help to tailor these products to specific needs.

Configuration

No two organizations are the same. Each has a different set of security and management concerns driving security policy, a different set of hardware and software platforms included in their systems environment, a different set of users or a

different set of operational policies. Therefore, the first issue facing a customer who acquires an intrusion detection system is the installation and setup of the system.

Many products, especially those designed for Windows NT environments, are shipped with clear, concise directions and installation scripts included. However, configuring these products is still an involved process. Information that customers must enter range from the IP addresses of the systems protected by the product to the sorts of security violations or system activities that the products are to detect and report. This is when a clear, current set of site security policy, procedures, and practices pays off handsomely.

Audit Subsystem Management

Products that include host-level agents typically use operating-system audit mechanisms. These products offer improved user interfaces to the operating-system audit controls, allowing users to specify what information is collected and how it is collected.

Reporting

One of the benefits of intrusion detection systems is the demonstration of care in system security management practice. A key to demonstrating this carefulness (e.g., to upper management, internal auditors and regulatory personnel) is to document the findings of intrusion-detection products over a particular time interval. Most intrusion-detection products have the ability to easily generate reports; many offer the capability to export report data to databases for subsequent analysis and archiving. Many offer multiple report formats (e.g., hard copy, screen, and HTML), with features allowing the user to report different layers of detail depending on the intended recipient of the report.

Control

Once the intrusion detection product is configured to the system environment, the next issue is actually running the system. Basic controls include starting and stopping the system, establishing the schedule at which certain activities should take place, and specifying how alarms should be handled. In the control function, another critical issue in intrusion detection products is addressed: the security and reliability of the intrusion detection system itself. One-way of addressing this is to require authentication before the system responds to control or configuration commands.

This reduces the risk of an adversary gaining access to the system and shutting it down.

Proof of Validity

In some cases, intrusion-detection systems are used to ensure the operation of other parts of the security infrastructure (e.g., firewalls). In this proof of validity, the intrusion-detection system analyses information from both inside and outside the coverage area of the security mechanism in question then compares results. The mechanism is proven valid when the intrusion-detection system isolates evidence that an attack (sensed on the outside) is blocked by the mechanism (therefore not sensed on the inside).

3. OVERVIEW OF IMMUNE SYTEMS

3.1 Natural Immune System

The biological immune system (IS) is the main defense mechanism of the human body. It protects the body from harmful micro-organisms called pathogens. It provides this protection by recognizing foreign cells or molecules and eliminating them from the body. There are very large number of pathogens such as bacteria or viruses. Therefore, the IS has a very complex structure to be able to recognize all of them. It is one of the most complicated systems of the body and sometimes its complexity is compared to the complexity of the brain [3] or the nervous system [4]. This IS does it is job of detecting and eliminating infections very well because the humans can still survive despite that many pathogens.

The IS is very important and even vital for the humans. If we did not have an IS we would die within weeks as in the case of some immune system infections. The immune system provides this defense mechanism with a multi-layered architecture. There are many different defenses on several levels of the IS (see Figure 3.1). The first level of the IS is the skin. It provides the first barrier to infections. Second level is the physiological condition in the body. For example, pH levels and body temperature eliminate some pathogens because of not being appropriate for them. Apart from these levels there are two subsystems of the IS: the *innate* immune system and the acquired or *adaptive* immune system. Both of these systems consist of a large amount of cells and molecules that work together to detect and eliminate pathogens. Both detection and elimination depends on chemical bonding. Surfaces of immune system cells are covered with various receptors. Some of these receptors chemically bind to pathogens and detect them. Others bind to other immune system cells or molecules to signal this detections and enable the IS to raise the immune response.

The "*innate immune system*" is the part of the immune system that exists when we are born and it does not change or adapt to specific pathogens (unlike the adaptive immune system). The innate immune system provides the initial response to the pathogens. This response is quick and it helps to keep early infections under control. Innate immune system has two chemical subsystems, which are complement system,

and the endocytic and phagocytic systems. These systems have some special cells wandering in the body and hunting pathogens. For example, cells called macrophages detect and swallow foreign molecules, and clean the system from both pathogens and their remainders.

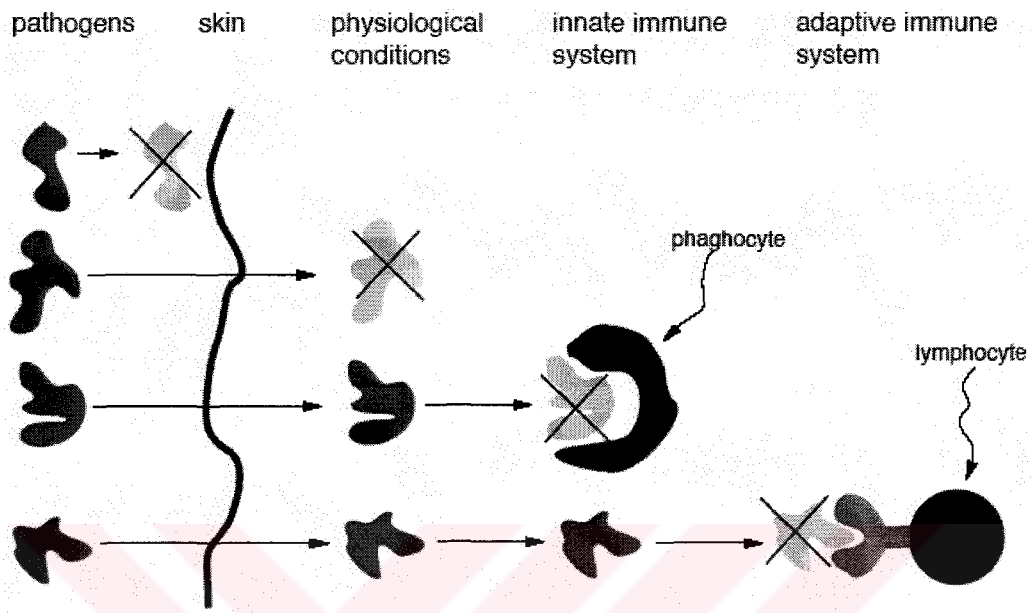

Figure 3.1 Layers of the Immune System

Unlike the innate immune system, the "*adaptive immune system*" adapts or "learns" to recognize specific kinds of pathogens, and keeps a "memory" of them. This memory then makes future responses quicker. The adaptive immune system learns pathogens during the process of the primary response which is described above. As mentioned, the primary response is slow and takes a long time to clear the infection but after the primary response clears the infection, the adaptive immune system keeps a memory of that pathogen. If the body is infected again by the same kind of pathogen, the IS will remember the pathogens, because it "remembers" their specific appearance. Because of this it can raise a much more quick and efficient secondary response. The secondary response is usually very quick. Sometimes even no clinical indications of a re-infection can be seen. This memory of the IS keeps the things it learned until the end of the life-time (as in the case of measles). The adaptive immune system is also called the acquired immune response system, because it is acquired during the life-time of the human.

The main components of the adaptive immune system are lymphocytes. Lymphocytes are special white blood cells that can detect pathogens. They can also help the elimination of these pathogens, but their major task is to circulate in the

body and detect foreign molecules. There are millions of lymphocytes in the body. They circulate in the blood and the lymph systems and detect pathogens independent from each other. They are not controlled by a central mechanism. This can be seen as a distributed system.

The main characteristics of the adaptive immune system [5] can be summarized as the following:

- **Antigenic specificity.** There are a huge amount of detector cells which can detect many different pathogens. Not all the detectors are unique to a specific pathogen but still most of them are different. This allows the immune system to distinguish different antigens.

- **Diversity.** The adaptive immune system has sufficiently many different detector molecules so it can recognize different structures of foreign antigens.

- **Immunologic memory.** Since the adaptive immune system keeps a memory of the seen antigens, it can give a quick response in reoccurrence of pathogens.

- **Self/nonself recognition.** As it was mentioned before, the immune system can distinguish self cells that belong to the body from foreign antigens. Thus, it responds only to the nonself molecules.

The innate immune system and the adaptive immune systems are described above as the subsystems of the human immune system, but they are not totally independent of each other. They work together to eliminate pathogens.

3.1.1 Recognition in the Immune System

The IS detects or recognizes pathogens as a result of the chemical bonding established between receptors on the surface of an immune cell, and epitopes, which are locations on the surface of a pathogen or protein fragment (a peptide). The surfaces of both receptors and epitopes have complicated three-dimensional structures that are electrically charged. The more matching the structure and charge of the receptor and the epitope, the more likely it is that binding will occur (see Figure 3.2). The strength of the bond between a receptor and an epitope is called the *affinity*. Each receptor on the surface of an immune cell can bind to only to a few similar epitope structures or patterns. Also all the receptor on one immune cell are identical. This provides the antigenic specificity (also called monospecificity) mentioned above. In contrast pathogens often have many different epitopes on their

surfaces. As a result several different lymphocytes may bind to a single kind of pathogen.

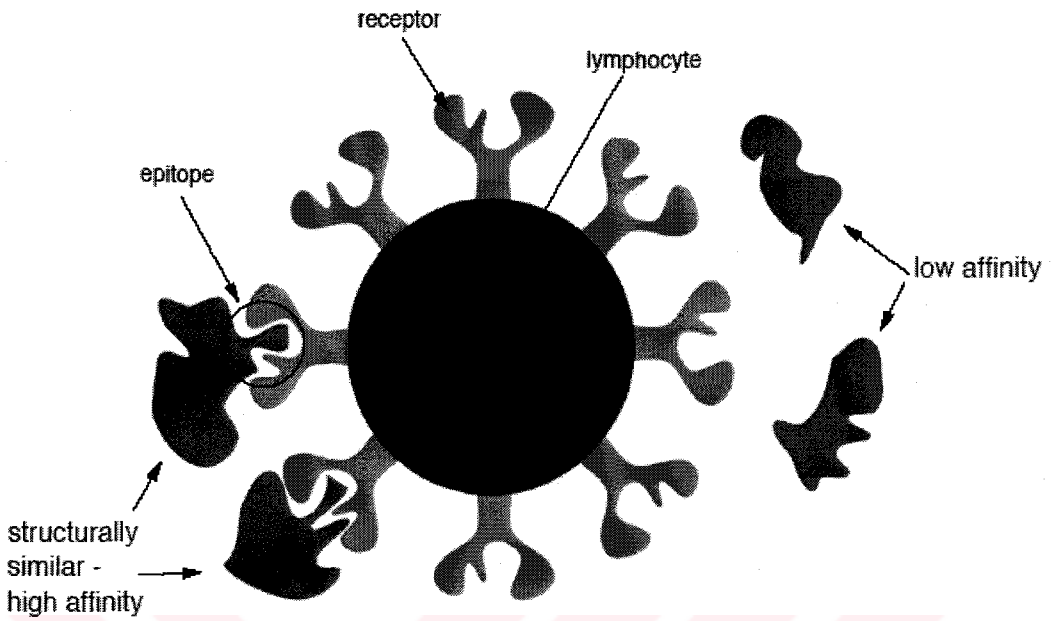

Figure 3.2 Detection in the Immune System.

There are a huge amount of receptors on the surfaces of each lymphocyte. As a result of this a lymphocyte can "estimate" the affinities of its receptors to epitopes by looking at the number of receptors binding to this kind of epitopes. It can also estimate the number of epitopes in its neighborhood, because the more receptors bound, the more pathogens in the neighborhood. The monospecificity mentioned above is very important, because if lymphocytes were not monospecific and could bind to different pathogens. The reaction to one kind of pathogen would cause response to other one.

The affinities are very important for the lymphocytes. When the number of receptors bound exceeds some threshold, the lymphocyte is activated. Therefore, lymphocytes are only activated when their receptors have sufficiently high affinities to epitopes on the pathogens and also there are sufficient number of pathogens around that lymphocytes. It is known that [6] certain kinds of lymphocytes (memory cells) have lower activation thresholds than other lymphocytes, and so need to bind fewer receptors to become activated.

These activation thresholds provide a generalization to the recognition process. Since each lymphocyte can detect structurally similar kinds of epitopes, more than the total number of lymphocytes can be detected. Hence, there does not have to be a different

lymphocyte for every epitope pattern to cover the space of all possible epitope patterns.

3.1.2 Receptor Diversity

Since IS detects nonself as a result of binding between lymphocytes and the pathogens, there must be sufficient diversity of lymphocyte receptors to ensure that at least some lymphocytes can bind to any given pathogen. However, to be called sufficient there must be a very huge amount of lymphocytes. This is a major problem because the human body can not produce so many different proteins. It is said that there are 10^6 different proteins in the IS while there are 10^{16} different proteins or patterns to be recognized [7]. To solve this problem the IS randomly recombines the DNA of the lymphocytes so different lymphocyte are produced. Similarly [8] states that there are at most 10^8 different receptors. Therefore, there are not sufficient number of different lymphocyte receptors to bind every single possible pathogen. This means that pathogens can evolve so that they cannot be detected, but IS prevents this by changing the lymphocytes continuously to provide a dynamic protection. It is reported that everyday approximately 10^7 new lymphocytes are generated [9]. If we assume that there are 10^8 different lymphocytes at any given time, and these are changed at a rate of 10^7 per day, there will be a totally different lymphocyte population 10 days later. As the time goes on, this change of lymphocytes and the immune memory property (see section 3.1.4) increases the protection of the IS.

3.1.3 Adaptation

The IS must detect and eliminate pathogens very quickly, because pathogens can replicate exponentially. If the lymphocytes are more generalized they will be slower at detecting specific pathogens, and less efficient at eliminating them. Thus, IS has some mechanisms that enable lymphocytes to "learn" or adapt to the structures of specific foreign proteins, and to "remember" these structures. This helps to provide faster responses in the future. This property of the IS is provided by special type of lymphocytes called B-cells. They are named as B-cells, because they mature in the bone marrow.

When a B-cell is activated (i.e. its affinity threshold is exceeded), it produces copies of itself through cell division. The mutation rates are very high in this copying process. Because of this, this process is called somatic hypermutation. As a result of high mutation rates this process can produce new B-cells that have different

receptors from the original cell. Different receptors mean different affinities to the pathogen. These new B-cells can also bind to pathogens. If they bind they will clone themselves too. The probability of a B-cell copying itself will be higher if it has high affinity for the pathogens present, so the overall affinity to the pathogens will increase. This is called *affinity maturation*. The process of copying only the cells with high affinities is called *clonal selection*. It resembles the survival of the fittest in the Darwin theory. In this process B-cells with the highest affinity will be the "fittest", and produce more clones. As a consequence of affinity maturation B-cells adapt to the specific pathogens present, because the number of B-cells with higher affinity to the pathogen will increase and this will result in a more efficient response to that pathogen so the infection will be eliminated more quickly. If a pathogen is replicating there will be a race between pathogen reproduction and B-cell reproduction.

A successful immune response results in the increase of B-cells that have high affinities for the foreign pathogens that caused the response. IS keeps these B-cells to form the "memory" of it. If the same pathogens are encountered again in future, the group of B-cells that previously learned that pathogen can provide a response that is quicker than the previous response.

Because of this learning and memory in the IS, it gives two different types of responses to the pathogens: primary response and secondary response. Primary response is given to pathogens that are not seen before. Secondary response is given to pathogens that have been encountered before. Normally, primary response takes a longer time. During this time B-cells learns the specific pathogens. Later, secondary response will be much shorter because the IS will remember those specific pathogens (see Figure 3.3). A secondary response is not only caused by the reoccurrence of the same specific pathogen. The occurrence of new pathogens that are similar to previously seen pathogens can also cause secondary response. This resembles the associative memory in the computers [10]. This idea is inspiration point of immunization. In the immunization process harmless forms of a pathogen is given to the human body which results in a primary response. After this primary response a memory of the pathogen is kept. Later, if similar but dangerous forms of the same pathogen occurs in the body, the IS will raise a quicker secondary response to that pathogen (see Figure 3.4).

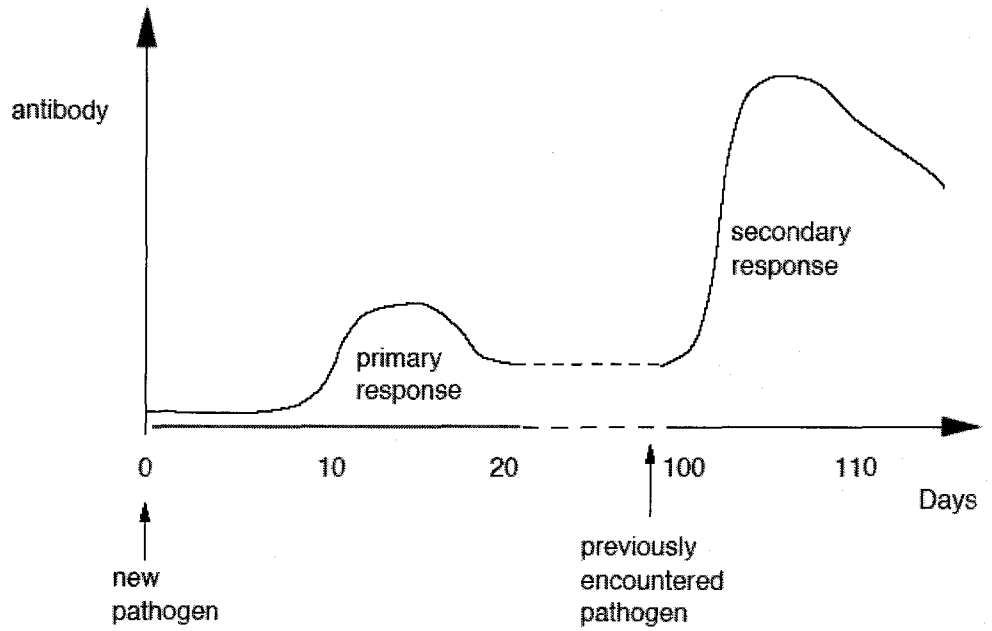

Figure 3.3 Responses in immune memory.

Figure 3.4 shows the associative memory property of the immune system. At time zero, the cowpox pathogen is introduced. It is harmless to the human body, but still it is recognized as foreign, so the IS raises a primary response to it. It clears this infection, and also keeps a memory of the cowpox. Smallpox is very similar to cowpox. Therefore, when smallpox enters in the body the memory detectors generated by the cowpox reacts to it and eliminate the smallpox in a more efficient secondary response.

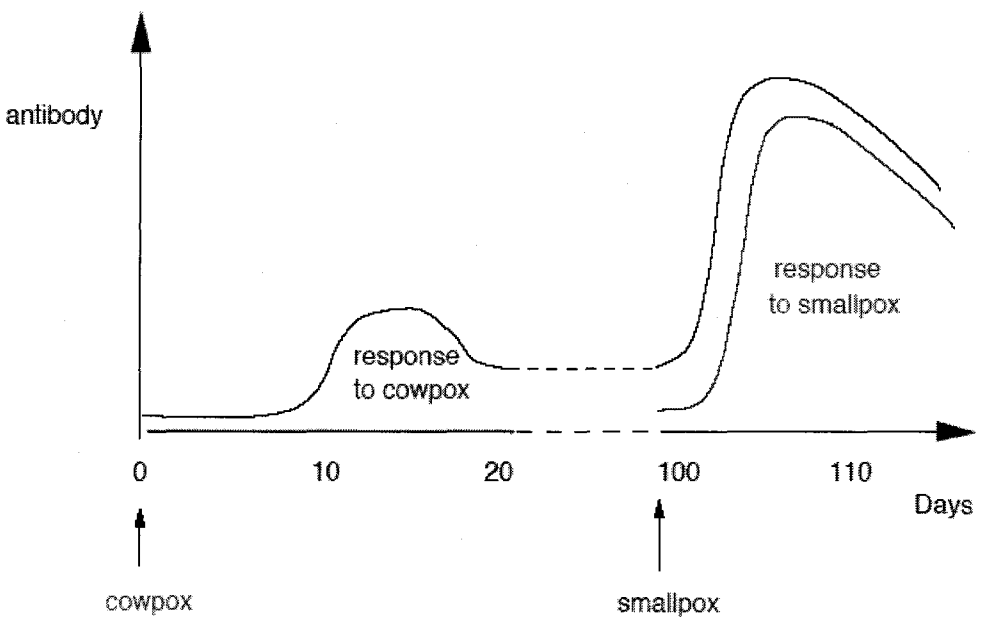

Figure 3.4 Associative memory in the immune system.

3.1.4 Tolerance

Up to now it is said that the receptors are randomly generated and randomly changed by somatic hypermutation. Since both of these are random processes the resulting receptors can also bind to some self cells and cause autoimmune responses. An autoimmune response occurs when the IS attacks the body, which is a major problem. However, this problem is not very common, in the humans [11]. Generally the IS is tolerant of self, that is, it does not attack self. Tolerance is the responsibility of another class of lymphocytes, called T helper cells (Th-cells). They are called Th-cells, because they mature in the thymus, and "help" the B-cells. The thymus is an organ located behind the breastbone and most self epitopes in the body are present in there. Therefore, during maturation Th-cells can come across most of the self epitopes. If an immature Th-cell is activated by binding with a self epitope, it will be killed in a process called clonal deletion or negative selection. Th-cells that survive the maturation process and leave the thymus will be tolerant of most self epitopes. This is called Central Tolerization (CT), because the immature Th-cells are tolerized in a single location (the thymus).

B-cells are tolerized in the bone-marrow, but this is not sufficient to prevent the production of B-cells that are reactive to self. Also, B-cells hypermutate during affinity maturation and this can result in previously tolerant B-cells producing autoimmune clones. Affinity maturation occurs in the germinal centers of the lymph nodes. Hundreds of these places are distributed throughout the body. Tolerance across these distributed locations is provided by a process called Distributed Tolerization (DT). Th-cells provide this through a method known as costimulation. To be activated, a B-cell must receive costimulation in the form of two different signals: signal I occurs when the number of pathogens binding to its receptors exceeds the affinity threshold (as described in section 3.1.1), and signal II is provided by Th-cells. If a B-cell receives signal I but not the signal II it dies.

Th-cells provide signal II to a B-cell, after they verify the epitopes detected by the B-cell. This verification process is very complex. B-cells swallow up pathogen peptides they detect as nonself. Therefore, Th-cells need to see the inside of B-cells to verify the detected epitopes. This is accomplished by using molecules of the Major Histocompatibility Complex (MHC). B-cells present the peptides they detected on their surfaces using these special molecules. These MHC molecules on the surface of the B-cell show the Th-cells what is inside the B-cell, that is, what the B-cell has detected. If a Th-cell binds to a MHC/peptide complex presented on the surface of a B-cell, it will provide signal II to that B-cell, and the B-cell will be activated. Since

Th-cells go through centralized tolerization in the thymus, most mature Th-cells are tolerant to self, so they will not costimulate B-cells that recognize self. The Th-cell verifies whether the detection carried out by the B-cell is correct or incorrect (autoimmune).

However, some of the self proteins are never presented in the thymus, so some mature Th-cells may still be reactive to the self proteins that are not present in the thymus. Therefore, self-tolerance in Th-cells is also assured through costimulation. In this case, signal I is again provided by exceeding the affinity threshold, but signal II is provided by cells of the innate IS. These innate system cells are thought to give out signal II in the presence of tissue damage. This may still allow autoreactive T-cells to survive, but when an autoreactive Th-cell leaves the region of tissue damage, it will receive signal I in the absence of signal II and die. This is called frequency-based tolerization because an autoreactive Th-cell should encounter self in the absence of tissue damage with higher frequency than self in the presence of tissue damage, because healthy self is generally more frequent than nonself. The frequency tolerization is important because the loss of the thymus does not result in fatal autoimmunity. However, if Th-cells were only tolerized centrally and not also by frequency tolerization this result should be unavoidable.

The different kinds of lymphocytes enable the IS to provide a faster adaptive response that is not self-reactive. Th-cells are responsible for self-tolerance, thus B-cells can hypermutate and adapt to a specific pathogen. Th-cells are general, nonspecific detectors, and so are not efficient at detecting specific pathogens. In contrast, B-cells can adapt to become more specific, and thus more effective at detecting particular pathogens.

3.2 Artificial Immune Systems

The natural immune system has been successful at protecting the human body against a huge amount of different foreign pathogens as mentioned in the previous section. This inspired many computer scientists to study the success of this natural mechanism and propose artificial immune models for solving various problems [12,13]. Different approaches to build AIS have been tried mainly by implementing a small subset of overall human immune mechanisms [12,13]. The most commonly used mechanisms of human immune systems and their application areas are as follows.

3.2.1 Negative Selection Algorithm

Negative selection is the tolerization process in the thymus as stated in section 3.1.4. In this process immature T-cells, which bind to self cells present in the thymus are eliminated. Therefore, only the T-cells which do not bind to any self cell are released from the thymus and distributed throughout the whole human body to monitor other living cells. This phase of the human immune system is important to make sure that the generated lymphocytes do not attack self cells [14,15].

Forrest et al. [16,17] proposed a negative selection algorithm inspired from this process. They think that the negative selection process of the human immune system is a complicated anomaly detection process. This process does not define specific harmful cells to be detected and thus allows the human immune system to be able to detect previously unseen harmful cells. This algorithm has three steps: defining self, generating detectors and monitoring the occurrence of anomalies. In the first step, it defines "self" by identifying normal behavior patterns of a monitored system like the other anomaly detection approaches. In the second step, it generates a number of random patterns that are compared to each self pattern defined in the first phase. If any randomly generated pattern matches a self pattern, this pattern fails to become a detector and thus it is removed. Otherwise, it becomes a "detector" pattern and monitors subsequent profiled patterns of the monitored system. In the third step, if a detector pattern matches any newly profiled pattern, it is then considered that a new anomaly must have occurred in the monitored system.

This negative selection algorithm has been successfully applied to detect computer viruses [16,18], tool breakage detection and time-series anomaly detection [19,20]. In addition to these practical results, D'haeseleer et al. [21] theoretically analyzed several advantages of negative selection as a new distributed anomaly detection approach.

3.2.2 Clonal Selection Algorithm

The human immune system learns dynamically changing antigens by a process called clonal selection. As stated in section 3.1.3 clonal selection is the process of copying the immune cells with high affinities to antigens. In this process activated B-cells are divided into a number of clones that have the same antigen-binding properties as parent B-cells, or mutated antigen-binding properties. On the other hand, if a B-cell is not activated within a limited time, it dies off. Therefore, only the fittest B-cells

survive. Since antigens continuously change, the efficiency of detection depends on the evolution of B-cells via clonal selection [15].

De Castro and Von Zuben [22] propose an AIS approach named CLONALG, which is inspired from clonal selection algorithm. CLONALG has an antigen and an antibody population. The antigen population stores the training data set and the antibody population keeps the candidate solutions. At every generation, CLONALG clones *n* selected antibodies proportionally to their fitness: the higher the antibody fitness, the higher number of clones. The cloned antibodies are mutated and the mutation rate is also determined depending on antibody fitness: the higher the fitness, the lower the mutation rate. From the mutated clones, CLONALG selects *m* mutants having highest fitness and these mutants are compared with selected *n* antibodies. Only k winners remain in the antibody population and l antibodies having lowest fitness are replaced by random antibodies. This algorithm is very similar to evolutionary algorithms (EA) because it uses a fitness based selection and applies the mutation operator which is also used in EA. The difference of this algorithm from EA is the method of determining the optimal antibody cloning rate and a mutation rate.

An improved genetic algorithm (GA) approach is proposed by [23], which also applies the clonal selection idea to AIS. Forrest et al. analyzes the strategy of the clonal selection process. The strategy of the clonal selection is learning the pathogens present, and thus learning the environment. As already mentioned this makes the immune system more efficient. Therefore, they investigated the effect of learning the environment and used this strategy to keep the generality and diversity of antibodies used for GA. They explored whether this strategy of clonal selection is able to i) detect common patterns of randomly presented antigens and ii) to maintain the diverse antigen population. In their model, they used the GA to evolve the antibody population under a constant antigen population. Like the strategy of the human immune system, their modified GA selects an arbitrary size of random sample from the antibody population, for each generation, and a single random antigen from the antigen population. After each antibody in the sample is matched against a selected antigen, the fitness score of the one highest-scoring antibody is increased while the fitness scores of the others stay the same. They compared this strategy of the artificial immune system with the fitness sharing algorithm. From this comparison, they reported that as the result of antibody sampling mechanism, the strategy of the AIS controls its generality by using the antibody sample size.

3.2.3 Gene Library Evolution

As mentioned in the previous section the human immune system learns dynamically changing antigens by a process called clonal selection. As the result of clonal selection, the surviving antibodies (memory cells) are able to detect various nonself antigens they have encountered before. However, it is known that the genes that define surviving antibodies and memory cells cannot be directly written back on the DNA (which is a gene library) of an egg or sperm cell. Therefore, the genes that define surviving antibodies are not passed on to the next generation of the immune system. However, it is also known that the learning results of the clonal selection during a lifetime indirectly results in the evolution of the gene library in the human immune system over generations. Although the genes of useful antibody mutants are not directly inherited, individuals that had more useful mutants are more likely to survive against various types of pathogens. Thus, the gene libraries of these individuals are passed over generations and offspring having these inherited gene libraries are more likely to have an immune system with a good capability of producing useful mutants. This effect was firstly proposed by Mark Baldwin in 1896 and named as the Baldwin effect.

Since the Baldwin effect was proposed, many researchers have attempted to understand the relation between learning and evolution in nature. Hightower et al. [24] have reported that "learning accelerates evolution" by simulating the gene library evolution of the human immune system. They used GA to let the gene library of a binary immune system evolve. Other work by Hightower et al. [25] investigated what determines the gene library evolution's direction. This work reports that the binary antibodies of a binary immune system evolve toward equilibrium between maximum coverage of antigen space and least overlapping coverage of antibody space.

Oprea and Forrest [26,27] improved Hightower et al.'s work [25] and studied the required diversity of a gene library in the human immune system and the role of gene library evolution. This work reported that the gene library tends to evolve towards the point of learning antigens that have experienced before. Later study by the same author [28] investigated the role of hypermutation by investigating its mutating targets. The given experiment results show that hypermutation usually attempts to mutate the antigen-binding regions of a gene library towards better antigen-binding parts.

There are two methods employed by the currently available AIS's in order to make their gene libraries evolve. The first approach directs gene library evolution through

a Baldwin effect and the second approach makes use of direct feedback from learning results to a gene library. These two different approaches have been implemented in various ways depending on the adopted AIS model.

3.2.4 Immune Network Theory

Immune network theory, which was first proposed by [29], describes that the immune system is able to provide a memory as a result of the binding between B-cells. This binding between the immune cells results in the creation of a network structure between them. This network structure helps the B-cell clones to support each other as well as suppressing each other when needed. The suppression is needed to keep the B-cell cloning under control and stabilize the kept memory. When this network finally reaches an equilibrium state between suppression and stimulation, it provides the overall immune memory.

This theory shows the parallel and distributed character of immune systems. It claims that there is a high interaction between immune cells themselves as well as the interaction between the immune cells and the antigens in parallel. These interactions determine the appropriate level of immune response. In also argues that the overall immune response is regulated not only by the interactions between lymphocytes and antigens but also the interactions among the lymphocytes themselves, even when there is no antigens. This theory tries to model how the overall immune response can be self-regulating. However, this model is not proven yet.

Despite the debates on the theory, various computer scientists have examined Jerne's Immune Network theory to understand the dynamics of human immune systems and propose a new learning algorithm. Farmer, Packard and Pearson [30] inspected the theory as a new computational intelligence approach. They implemented Jerne's model using binary strings to represent antibodies and antigens. They formalized the concentration level of each antibody, determining the number of antibodies produced in cloning. The formula that determined the concentration level of each antibody has four factors: i) the stimulation by existing matching antigens, ii) the stimulation by neighboring antibodies, iii) the suppression by neighboring antibodies and iv) the tendency of cell to die due to a finite life span. As stated in ii and iii the neighboring antibodies could both stimulate and suppress an antibody. This is shown in Figure 3.5, each lymphocyte has two different types of protein structures: paratopes and idiotopes [31].

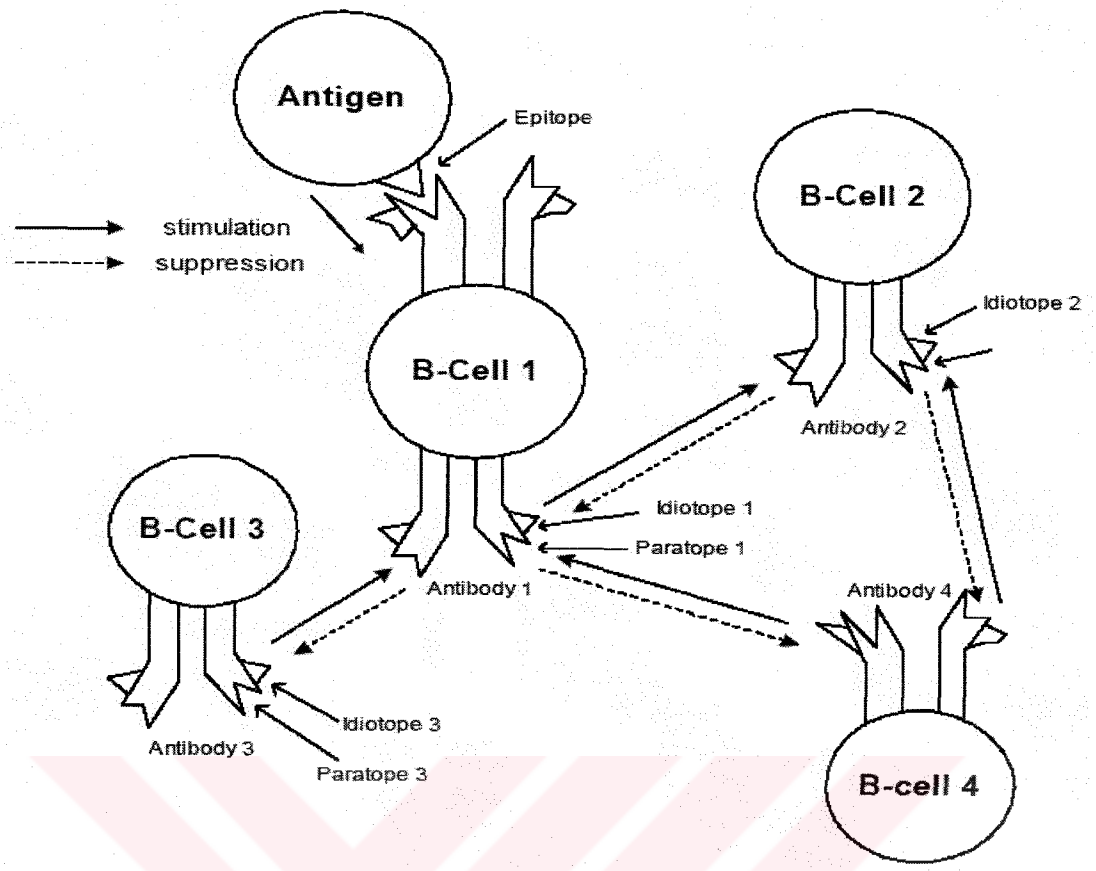

Figure 3.5 Jerne's Immune Network

Antibodies treat protein structures on the surfaces of cells that bind to their partatopes as antigens. In contrast, they treat proteins binding to their idiotopes as other antibodies. As a result, the concentration level of a given B-cell is increased by bindings with its paratopes and reduced by bindings with its idiotopes. Thus, if paratopes of a current antibody bind paratopes of other neighboring antibodies, the concentration level of a given antibody is increased. Similarly, it decreases if paratopes of a current antibody bind the idiotopes of other neighboring antibodies.

This work also model immune network theory as classifier system, which was already well-known as a computational intelligence approach. This resulted in many computer scientists' investigating the use of artificial immune model for computational intelligence purposes. Varela et al. [4] described that the immune system is cognitive because it has four important features. They are i) recognition of other molecules, ii) memory of encountered molecules, iii) determination between self and nonself molecules and iv) associative memory to recognize molecules that have not been encountered.

3.2.5 Immune Memory

Immune memory is defined as the ability of the immune system to fully protect the body from attack of pathogens that have previously been detected [32]. When B-cells are activated, they produce memory cells to protect against the reoccurrence of the same antigens. Memory cells enable the immune system's response to previously encountered antigens (known as the secondary response), which is known to be more efficient and faster than non-memory cells' response to new antigens [15]. The memory cells have long life times. They often last for many years or even for the lifetime of the human [15].

AIS's have implemented the immune memory of the human immune system in various ways. Most of these the implementations use the immune network theory to provide the memory property [31]. The researches on immune network theory show that the network structure formed by immune recognition forms the memory of the immune system. The new network formed by surviving B-cells can provide a new solution to a newly appearing environment, without losing the solutions to the previous environment. This is possible because the AIS selects surviving B-cells in the network not only by their antigen stimulation level but also by antibody suppression level. Although some antibody emitting B-cells did not receive a sufficient degree of stimulation from new antigens, they would not be deleted as long as they did not also receive a high degree of suppression from other antibodies. The B-cells having these antibodies remain and act as memory cells in the AIS.

Another type of immune memory employed for AIS's is Sparse Distributed Memory (SDM) [10]. Smith et al. [10] state that the approximate recalling mechanism of SDM is equivalent to the primary and secondary responses of memory cells. Consequently, Smith et al. claimed that immune memory is robust and associative, as is SDM, since both employ a similar approximate addressing mechanism.

In contrast to above approaches, Hofmeyr's AIS [33] adopts a separate memory cell population that is isolated from other antibody populations. This system is the only AIS to implement immune memory by directly representing memory cells of the human immune system. It has explicit antibodies to keep memory of previously detected antigens and these antibodies are treated differently from other antibodies. They remain in a population for a longer period than other antibodies and they detect antigens much more quickly than other antibodies.

3.2.6 Artificial Immune System Applications

This section briefly introduces application areas where AIS have been applied. Various application areas utilize AIS's, some of hese areas are as follows:

- Computer Security
- Computer System Maintenance
- Concept Learning
- Robot Control
- Fault Detection
- Optimization
- Scheduling
- Aircraft Control
- Production Line Management
- Pattern Recognition
- Spectra Recognition
- Vaccine Design
- Multi-Agent Systems
- Open Web Server Policy Management
- Protein Structure Prediction
- Fault Tolerant System Design

As seen in the previous sections, existing AIS's have various forms and each different form is based on different parts of the human immune system. This allows a quite large number of application areas to use AIS for different purposes.

3.3 Artificial Immune Systems for Computer Security

Computer security is one of the application areas that AIS have been most frequently applied to. The similarity between the goal of a computer security system and the human immune system naturally leads researchers to examine similarities between the two. This section summarizes two of the most studied different computer security related applications that implements human immune features.

3.3.1 Virus Detection

After the computer viruses have been identified as a destructive form of artificial life, one of the first ideas that come into the mind of researchers to solve this problem was using AIS. It is very natural, because the human immune system, which is the inspiration point of AIS, is also accomplishing a similar task by protecting the human body against harmful biological viruses. Stephanie Forrest at University of New Mexico was the first researcher in the study of the application of human immune system techniques to protection of computer systems from destructive artificial life. Forrest et al. [16] view virus detection as a "self-nonself discrimination problem" in a computer. They regard monitoring targets (such as legal user activities, legal application usage activities, uncorrupted data, etc.) as self and expect the AIS to discriminate them from nonself (such as illegal user activities, illegal application usage activities, virus infected data, etc.). This work models the negative selection process of human immune systems. In their work, Forrest et al. generates detectors from a standard binary executable .com file and then tests whether generated detectors can detect a virus infected .com file. The experimental results show that their AIS obtained 100% detection rate with 125 detectors when an infected file is encoded by 655 binary strings each string having 32 bits. However, their work has not been further tested against other viruses.

Another approach called Computer Virus Immune System (CVIS) [34,35], by the US Air Force Institute of Technology team, attempts to apply the latest AIS of Stephanie Forrest's team at University of New Mexico. The Forrest team's latest AIS [33,36] is developed to detect a certain set of network intrusions and is not tested for virus detection. CVIS uses the new negative selection algorithm that was used in [33,36]. The new algorithm has some new ideas such as life span, activation threshold and costimulation. As new features, CVIS adds detected virus analysis, repair of infected files and distribution of analysis results to other local systems. In addition, CVIS is designed to operate under a distributed environment using autonomous agents.

Although they design the AIS under a distributed environment, the reported test results of CVIS are limited to evaluation of whether CVIS detects various viruses on a local host. The tested viruses are the "TIMID" virus, which infects .COM files only within a local directory, and viruses included in the standard test file provided by the European Institute for Computer Anti-Virus Research. The test reports show the sensitivity of detection and error results depending on different matching thresholds. By setting appropriate parameters, their system is able to show a detection rate of up to 89%. One serious problem found from these tests is scalability. It requires about one year for generated antibodies to scan an 8GB hard disk drive.

Okamoto and Ishida [37] propose a multi-agent based AIS which can be used as a virus detection system that can operate under a distributed and heterogeneous environment. This system consists of four different types of agents: antibody, killer, copy and control agents. Antibody agents detect viruses from local hosts and killer agents deactivate viruses by overwriting self information on infected files. Copy agents copy uninfected files, which are equivalent to ones infected, from different hosts and control agents help communication among other agents. This system is clearly limited by their repair method since there are only limited numbers of files that have equivalent ones from other hosts.

A different approach to using AIS for virus detection is undertaken by [38-41] at the IBM research centre. While above approaches model a particular sub-mechanism of human immune systems, the AIS developed by [38-40] identifies some features of the human immune system that make it attractive for virus detection purposes and implements them using well-known algorithms. Thus, Kephart et al. design their AIS with the following five stages, each inspired by the human immune system: i) detect a previously unknown virus on a user's computer, ii) capture a sample of the detected virus and send it to a central computer, iii) analyze the virus sample automatically and derive an instruction for detecting and removing it from any computer host, iv) send the derived instruction to the user's computer, adding it into the anti-virus data files used by the anti-virus software, and run the anti-virus product to detect and remove all occurrences of the virus, and v) broadcast the derived instruction to other computers in the user's locale and to the rest of the world. Kephart et al. [38-40] claim each of these stages represents different sub-procedures of the human immune system. However, their implementation is completed using other well-known algorithms such as neural networks.

3.3.2 Intrusion Detection

The first attempt to use an AIS for intrusion detection is the host-based IDS [17,18,42]. A host-based IDS using AIS profiles the normal behaviors of privileged system processes and compares them to currently observed behaviors. This first computer immunology approach for intrusion detection does not make use of many features of human immune systems, such as distributed anomaly detection. Instead, it tries to use the basic concept of the human immune system: detection of nonself patterns. This is not very different from the usual anomaly detection mechanisms. An interesting point of their work is that it shows that system call sequences of privileged process can be used to describe the self of a monitored host. More importantly, their work shows that the profiling of system calls can result in a considerable decrease in the amount of audit data and thus this approach can help to build a light-weight IDS. However, they do not report whether this level of audit is enough to replace other more extensive audit levels, such as auditing all application programs activities, auditing all typed user commands and auditing system resources and network traffic activities.

Later the same research team extended their AIS to a network intrusion detection system. Forrest et al. [17] proposes the application of computer immune systems to computer security for the first time. From their work, they argue that the analogy between human immune systems and artificial immune systems works best for network intrusion detection problems. More recent work by [33,36] develops an AIS for network intrusion detection, called LISYS. LISYS implements the AIS proposed in [17]. It uses various features of the human immune system such as activation threshold, life span, memory detectors, costimulation, etc., in order for it to monitor dynamic self and nonself behaviors. In addition, LISYS operates under a distributed environment and it is tested to monitor 50 hosts in a local area network (LAN). The system considers the typical network traffic paths between two hosts as self and detects abnormal network packet paths. In order to perform this, LISYS extracts a "datapath triple", which is a source host IP address, a destination host IP address and a TCP service (port) number from TCIP/IP packet headers. This "datapath" is used as input data to build self-profiles. LISYS is tested over seven intrusions and shows promising results. However, this quite limited input data requires further research to evaluate whether LISYS is able to detect more diverse intrusions than those used in [33,36]. This is the main goal of the work in this thesis. This work implements the features of LISYS as mentioned by [33,36] and analyzes it using the well-known DARPA dataset. This analysis will be presented in the next chapter in more detail.

The AIS studied in this thesis shares many features of LISYS and provides new understanding of these features that were not reported by [33,36].

Different work inspired by LISYS has recently been reported in [43]. Williams et al. improve Computer Virus Immune Systems (CVIS) reported at [34,35] to detect network intrusions. This extended system, called Computer Defense Immune System (CDIS), also uses the negative selection algorithm with some new ideas that were introduced in LISYS such as life span, activation threshold, and costimulation. In contrast to LISYS, CDIS chooses 28 features of TCP packet headers and 16 features of UDP and ICMP packet headers as its input self data. For antibody string generation, CDIS randomly selects the network protocol and chooses between two and seven features of that selected protocol. For the selected protocol features, CDIS generates the values of these features randomly. As a new way of reducing the computing time to generate antibody strings, CDIS uses an affinity maturation process. This process aims to optimize each antibody to cover the nonself space as much as possible without matching any self string. In order to perform this, CDIS uses a genetic algorithm and its fitness function is defined as the growth rate of nonself space coverage by each antibody. The performance of CDIS is tested over two data sets: the first set with 20K self packets and the second set with 1.3M self packets and both data sets are collected during one day. For a nonself data set, Williams et al. simulate various intrusions on attack-free network traffic data. The test results show that CDIS is able to detect simulated intrusions without serious self detection errors. It also verifies that the costimulation and affinity maturation help CDIS to reduce both false positive and false negative error rates. However, the affinity maturation requires too much computation time to be applied to the second, larger, data set. In addition, the authors also point out that the high detection rates with low error rates may possibly be obtained because the simulated intrusions are limited.

Dasgupta et al. [44] proposes a general framework for Immunity-Based IDS. This framework applies a multi-agent architecture to provide an AIS model for intrusion detection. This immunity-based IDS framework simply follows the multi-level detection feature of human immune systems instead of using more complicated features of human immune systems, such as distributed detection, clonal selection, negative selection, etc. The multi-agents in this model monitor systems at various levels in a hierarchical manner, activate warning signals, exchange their local warning signals and make decisions based on collected local warning signals. In order for the proposed IDS to reduce false warnings, this model emphasizes the importance of collective decisions. It monitors anomalies of users' system usage

patterns, system resource usage patterns, process activity patterns, and network traffic patterns at the same time and produces final warning signals only when relationship among local warnings seem to be intrusion-related. The ART-2 NN is used to detect anomalies on all monitoring levels and fuzzy logic was used to combine four different levels of warnings into a final threat warning [45]. The different work [46] instead uses the classifier system to draw final collective decisions. The fitness function used for the classifier system is defined by three factors: "How well each rule follows security expert's heuristic rules", "the diversity of a rule set", and "the generality of a rule".

Since the negative selection algorithm was proposed by Forrest et al. [16] as a new anomaly detection algorithm, many researchers have been inspired by this revolutionary work. However, it is later found out that this algorithm has some natural problems. Dasgupta and Gonzalez [47] have examined whether the idea of negative selection is advantageous for the network intrusion detection problem. They have compared a negative characterization approach to a positive characterization one. The positive characterization approach is one of conventional anomaly detection algorithms that usually define normal profiles and detect anomalies by monitoring a significantly different event. A self set is a collection of vectors that present one event observed at time t and these vectors are grouped into a number of sets. When a new data is given, this approach searches for a nearest self vector to the given data point and measures the Euclidean distance between these two points. If this distance is larger than a pre-defined threshold value, the given data is considered anomalous. On the other hand, their negative characterization approach employed a genetic algorithm in order to generate detector rules covering holes of a nonself space. While the negative selection algorithm generates detectors covering the same radiuses of clusters in a nonself space, Dasgupta and Gonzalez [47] uses the GA to let detectors evolve to cover generalized holes that have various radiuses. In order for detector rules to evolve, the fitness function was defined by the volume of nonself space covered by detector rules after a penalty having been applied according to the number of matching self examples.

The experimental results in their work show that the negative characterization approach is more efficient than the positive characterization one in terms of space required. It also states that positive characterization approach is better in terms of its detection rate. It is also known that negative selection algorithm has a problem with respect to computing time [48] but this is not investigated in this work. In addition, distributed detection using a negative characterization approach is not investigated either, even though this is viewed as a potential advantage of negative.

Stillerman et al., [49] introduce an immunity-based intrusion detection approach that is to Common Object Request Broker Architecture (CORBA) applications. CORBA is a popular common messaging middle-ware that enables the communication of distributed objects forming a distributed application. Their work focuses on detection of attacks on CORBA applications, called rogue client attacks. This attack is of the "legal user's misuse" type. Its approach to detecting this type of attack is closely related to the early AIS reported in [17,18,42], in that it is not much different from conventional anomaly detection. As in [17,18,42], a self database is built using a sliding time window to define a self pattern from collected sequences of requested methods. After sufficient self-patterns have been collected, new patterns that are not found in the self database are regarded as anomalous. To reduce false positive error rates in the system, it waits until a sufficient number of anomalies from the same client have been observed. The experimental results show that the system is able to detect anomalies caused by this attack without high false positive error rates. Although this report shows that their work is feasible when applied to the CORBA application level of attack, the report does not include exact false positive error rates or the training and test data collection periods in the experimental results. Furthermore, their work does not use any of the new ideas of AIS instead using a conventional anomaly detection approach called an immune-based system.

3.4 Summary

This chapter has briefly introduced the overall human immune mechanism and surveyed existing AIS's. Recently, a large number of computer scientists have studied this natural mechanism and proposed artificial immune models for solving various problems. However, due to the complexity of the overall human immune mechanisms, AIS's have mainly implemented a small subset of the overall human immune mechanisms. Among these AIS's, five of the most commonly used artificial immune algorithms are introduced. However, none of these systems can fully provide the significant features of the human immune system. This is also true in the computer security application area. Most of AIS's used for a computer security application [16,18,33,35,36,44,49] mainly use a small subset of overall human immune mechanisms and their performances have only been verified in very limited test cases. Other AIS's used for a computer security application [38,40,41,43] are commercially used or tested on a real environment. However, these AIS's only follow the basic concepts of the human immune mechanisms and actual implementation use available traditional algorithms.

4. ARCHITECTURE OF THE APPLIED AIS

As stated in the previous section the main goal of the work in this thesis is the implementation and analysis of the proposed artificial immune system for computer intrusion detection by [33,36]. Therefore, it is crucial to understand the architecture of the proposed system to understand and assess the results of this work.

This section describes the architecture of the artificial immune system proposed by [33,36] as a generalized system for anomaly detection using immune-based approaches. This system is named ARTIS and it is described independently of any particular problem domain. However, Hofmeyr and Forrest also mentioned utilization of ARTIS in a networked environment as an intrusion detection system called LISYS, which we will be looking at more closely in the next section.

Since ARTIS is closely modeled on the biological immune system, the following description will also describe the equivalent biological mechanisms that the model is based on.

4.1 Definition of the Problem

Since ARTIS is defined as a generalized system for anomaly detection using immune-based approaches it accomplishes a similar task as the IS. IS distinguishes self and nonself based on chemical bonding between protein chains, and ARTIS distinguishes self and nonself based on some match rule. The protein chains in the IS are modeled as binary strings of fixed length ℓ in ARTIS. The problem that ARTIS is proposed to solve is defined as follows: Suppose set of all strings of length ℓ forms a universe, U, which is partitioned into two disjoint subsets, which are called self, S, and nonself, N (i.e. $U = S \bigcup N, S \bigcap N = \varnothing$). The task of ARTIS can be summarized as: Given a random string from U, classify it as either normal (member of self) or anomalous (member of nonself).

As in the case of IS, ARTIS can also make errors in this classification task. There are two kinds of errors: false positive error and false negative error. A false positive error can be defined as the misclassification of a string belonging to self set and its identification as anomalous. Where, a false negative error is the misclassification of a string belonging to nonself set, and its identification as normal. Similarly, these

errors also occur in the IS. However, they are very harmful for the body. Therefore, IS has developed some mechanisms to minimize these errors. ARTIS is also designed to minimize both kinds of errors.

Above mentioned error types are shown in Figure 4.1. The figure represents the universe of strings in two-dimensions. Every point in that two-dimensional plane represents a different string which can belong to either self or nonself sets. The shaded area represents the self set. The task of ARTIS is to determine the boundary between the two sets by classifying strings as either normal (member of self) or anomalous (member of nonself).

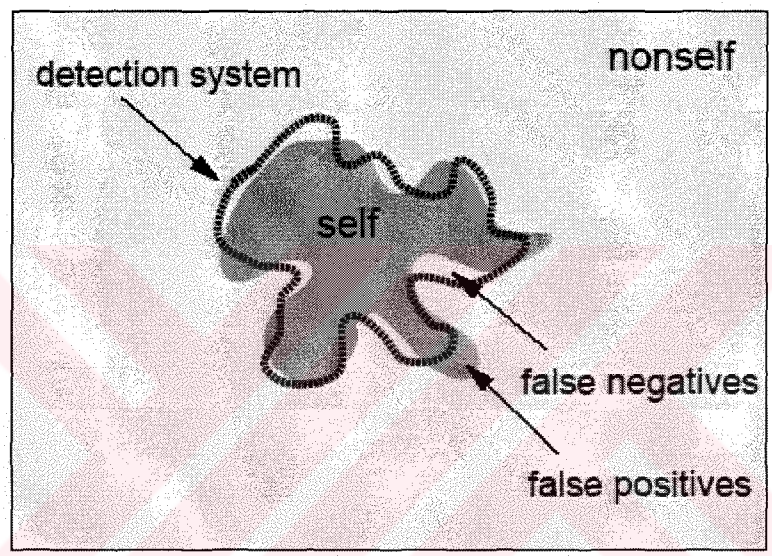

Figure 4.1 A two-dimensional representation of a universe of strings.

Sometimes some strings may belong to both self and nonself sets. In such a case, categorizing strings as either self or nonself can lead to errors, but ARTIS does not consider that case because it is designed as a general model. However, when using ARTIS, it is very important to choose the appropriate representation for the data. The representation should be chosen so that self and nonself can be distinguished reliably.

4.2 Detectors

There are many different kinds of cells and molecules in the IS which are used as detectors. However, ARTIS has only one type of detector which is modeled based on the immune cells called *lymphocytes*. This detector combines properties of B-cells, T-cells, and antibodies. Like the IS, ARTIS also has a large number of detectors, that circulate around the body. This property is modeled as a distributed environment, which is represented with a graph. If the graph is denoted as $G = (V, E)$ where V

represent vertices of the graph and E represent the edges, each vertex $v \in V$ represent a set of detectors at a local place called a *detection node.* The edges represent the connections between these vertices which are used to copy detectors from one vertex to the next. In this representation the detectors in one place only interact with each other.

As stated above IS distinguishes self and nonself based on chemical bonding between detector cells and pathogens. Lymphocytes have great number identical receptors on their surfaces. These receptors bind to regions (epitopes) on pathogens. Binding depends on chemical structure and charge, so receptors are likely to bind to a few similar kinds of epitopes. The greater the likelihood of a bond occurring, the higher the affinity between the receptor and epitope. In ARTIS, both epitopes and receptors are modeled as binary strings of fixed length ℓ, and chemical binding between them is modeled as approximate string matching. In fact, each detector is associated with a binary string, which represents its receptors.

Although, there are many different approximate matching rules such as hamming distance and edit distance, Hofmeyr and Forrest favored to use *r-contiguous* bits [14] rule because they think it is a more "immunologically reasonable". As can be implied from its name, the r-contiguous rule matches two strings if they have r contiguous bits in common. This is shown in Figure 4.2. In this example, the detector matches for $r = 3$, but not for $r = 4$. The value r is a threshold and determines the specificity of the detector, which determines the size of the subset of strings that the detector can match. For example, if $r = \ell$, the matching is totally specific, which means the detector will match only a single string(itself), but if $r = 0$, the matching is totally general, which means the detector will match every string of length ℓ.

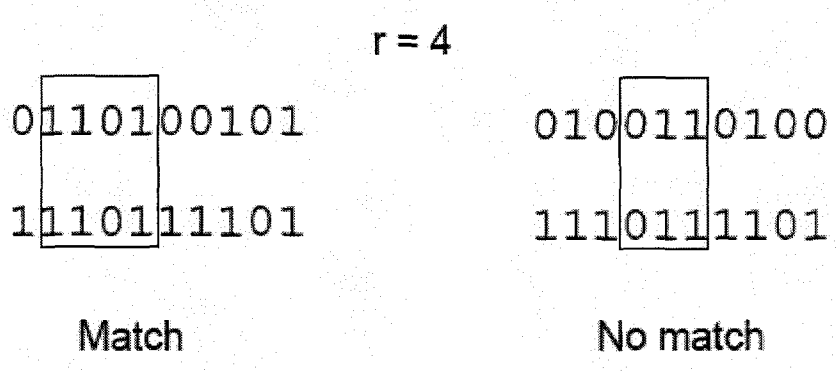

Figure 4.2 Matching under the contiguous bits match rule.

Since the *r-contiguous* bits rule is an approximate match rule, there is a trade-off between the number of detectors used, and their specificity. If the specificity of the detectors increases, the number of detectors must also increase to provide a certain level of detection. Therefore, the optimal r value must be chosen as the one which minimizes the number of detectors needed while providing good coverage.

When the number of receptors on a lymphocyte that bind to epitopes exceeds a threshold, the lymphocyte becomes *activated.* After the activation the lymphocyte gives some reactions to eliminate the pathogens. Since the binding occurs as a result of chemical bonding and the chemical bonds between receptors and epitopes are not long-lasting, a lymphocyte must bind enough receptors within a short period of time to become activated. ARTIS models this with *activation thresholds*: A detector must match at least τ strings within a given time period to be activated. This is implemented by keeping a match count for each detector. This match count is incremented on each match until it reaches the threshold. However, since these matches must occur within a given time period the match count is decremented over time. The match count is reduced by one with a probability of γ_{match} at each time step. Once a detector has been activated, its match count is reset to zero.

4.3 Training the Detection System

The training of the lymphocytes are accomplished against self so that they do not match self, but match only nonself. Because of this they are called *negative* detectors. This training provides a simple form of learning mechanism which is known as *tolerization.* This process is called tolerization because lymphocytes learn to be "tolerant of self".

In the training process, lymphocytes are created with randomly generated receptors. Therefore, they can bind to both self and nonself. The lymphocytes called T-cells, is tolerized in the thymus, which is an organ just behind the breastbone. Immature T-cells develop in the thymus, and if they are activated during development, they die through a mechanism called programmed cell death (*apoptosis*). Since most of the self proteins are present in the thymus, the T-cells that undertake this tolerization and leave the thymus will be tolerant of all those self proteins and do not match with them. This process is known as *negative selection*, because the T-cells that are not activated during the tolerization can leave the thymus, not the ones that are activated.

This training process teaches the lymphocytes to distinguish self and nonself, in other words normal and abnormal. Therefore, they learn how to perform *anomaly* detection. Since this training is done using proteins present in the thymus as the

training set, there will be some problems if nonself proteins are frequent in the thymus. The major problem is that the IS will also be tolerant to that nonself proteins in the future and they will not be eliminated by the IS. However, it is assumed that that the self proteins occur more frequently than the nonself proteins in the thymus. This assumption is the basis of most anomaly detection systems, which define normal as the most frequently occurring patterns or behaviors.

ARTIS uses the *negative selection algorithm*, which is based on negative selection in the IS (see Figure 4.3) [16]. Nevertheless, there are some differences in the ARTIS design. The main difference is that it has a distributed architecture. There are more than one locations keeping different local detectors. This can be implemented as different detection systems running on different computers. The tolerization is then realized in an asynchronous and distributed manner. Each detector is created with a randomly-generated bit string (analogous to a receptor), and remains immature for a time period T, called the tolerization period. During this time period, the detector is compared with the data strings in the environment (self and possibly nonself strings), and if it matches any bit string it is eliminated. If it does not match during the tolerization period, it becomes a mature detector (analogous to a naive B-cell). After a detector matures it is continued to compare it with the data strings and its match count is incremented after each match. If its match count exceeds the activation threshold it becomes activated. Unlike immature detectors mature detectors are not eliminated when they are activated. Oppositely, this is interpreted as a signal of an anomaly in the system. Here it is assumed that a mature detector can only match with nonself strings because if it matched with self strings it would have matched in the tolerization period and eliminated because of an immature match.

Figure 4.3 illustrates this negative selection algorithm. Candidate negative detectors (represented by dark circles) are generated randomly, and if they match any string in the self set (i.e. if any of the points covered by the detector are in the self set), they are eliminated and regenerated. This process is repeated until a set of valid negative detectors that do not match any self strings are generated.

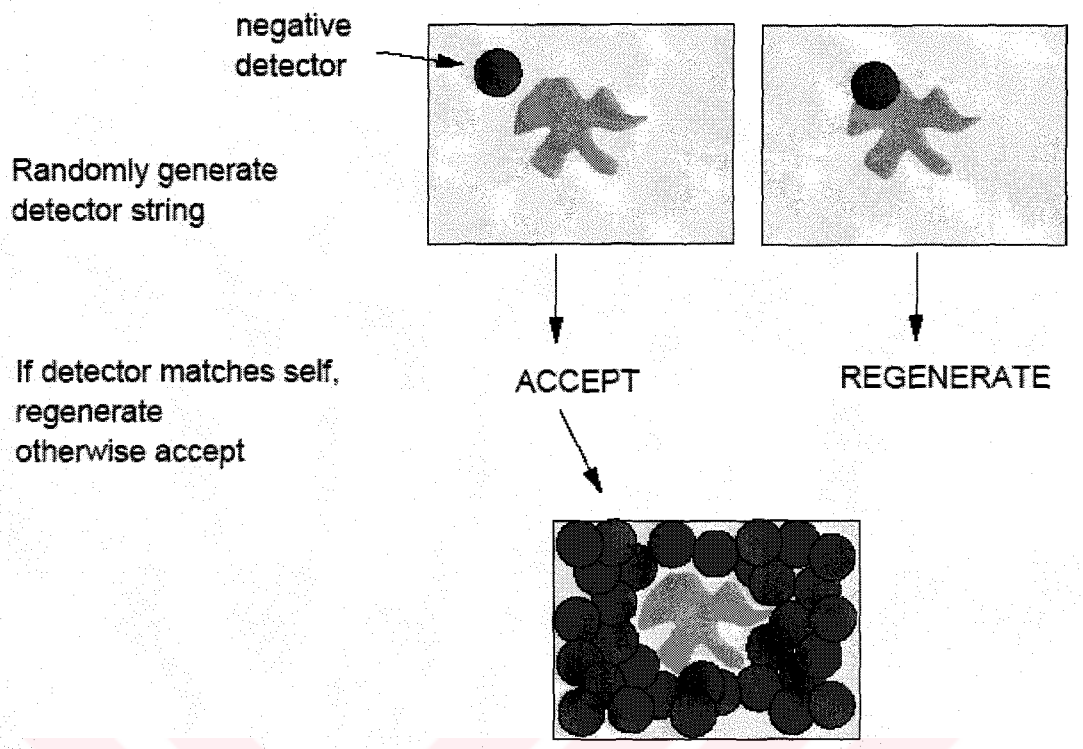

Figure 4.3 The negative selection algorithm.

4.4 Memory

As mentioned in section 3.1.3 the IS has an adaptive response mechanism. It learns protein structures that characterize pathogens it detected, and "remember" those structures so that future responses to the same pathogens will be quicker and more efficient. This is called memory-based detection, because the IS "remembers" the structures of known pathogens to help future responses. A memory-based detection system can be trained on a subset of nonself to detect elements of that subset if they occur again in the future. When the IS come across to a pathogen that it has not encountered before, it raises a *primary response*, which may take a long time to eliminate the infection. It learns that specific pathogen during this primary response and if it comes across to that pathogen again, it raises a *secondary response*. This response is generally more efficient than primary response and sometimes the re-infection can not even be realized. The secondary response shows how efficient a memory-based detection could be.

The memory property of the IS is known to be associative [10]. Which means it can detect new pathogens that are not previously encountered but structurally similar to the ones that are previously encountered. This idea is the basis of the immunization process. In the immunization process the body is inoculated with a harmless form of

pathogen such as an attenuated virus. This causes a primary response against that pathogen and a quantity of memory cells that are responsive to this pathogen are generated. This generated memory cells should also be responsive to the harmful forms of the same pathogen or to a totally different pathogen that is similar to the previous one. Therefore, if this second type of pathogens are encountered in the future the IS will raise a secondary response and eliminate them.

There are a huge amount of lymphocytes in the body. Although each of them is not unique, they all represent many different protein structures. Therefore, there may be very few lymphocytes to match a specific pathogen. Because of this, the primary response to a pathogen is generally slow and not very efficient. Since the efficiency in eliminating pathogens is important IS solves this problem by cloning activated lymphocytes. This result in an exponentially increasing population of lymphocytes which can detect the pathogens encountered. If the affinity between a lymphocyte's receptors and the pathogen epitopes is high it will be activated and cloned with a high probability. Hence, the lymphocytes that are cloned are those with the highest affinity for the pathogens present. During this time the pathogens are also replicating, so there is a race between pathogen replication and lymphocyte replication. The IS tries to improve its chances in this race through a class of lymphocytes called B-cells, which have high mutation rates, called somatic hypermutation, when cloning. Hypermutation combined with clonal expansion is an adaptive process known as *affinity maturation*. Once the infection is eliminated, the IS keeps a population of memory cells: long-lived lymphocytes which have a high affinity for the pathogen. This population of *memory* cells is sufficient to enable the very quick secondary response when a re-infection occurs. ARTIS uses a similar form of memory-based detection. When multiple detectors at a node are activated by the same nonself string, s, they enter a competition to become memory detectors. Those detectors that have the closest match (under r-contiguous bits rule) with s will be selected to become memory detectors. These memory detectors make copies of themselves, which then spread out to neighboring nodes. As a result, a representation of the string s is distributed throughout the graph; future occurrences of s will be detected very rapidly in any node because detectors that match s exist at every node. Additionally, memory detectors have lower activation thresholds (for example, $\tau = 1$), so that they will be activated far more quickly in future to re-occurrences of previously encountered nonself strings, i.e. they are much more sensitive to those strings. This imitates the fast second response seen in the IS.

4.5 Sensitivity

A detection event in the IS often results in the production of chemicals (cytokines) which signal other nearby IS cells. ARTIS models this using the locality in the graph defining the environment. Each detection node D_i (where i = 1,2,..., |V|) has a local sensitivity level, w_i which models the concentration of cytokines present in a physically local region in the body. The activation threshold of detectors at D_i is defined as $\tau - w_i$, i.e. the higher the local sensitivity, the lower the local activation threshold. Whenever the match count for a mature detector at node i goes from 0 to 1, the sensitivity level at D_i is increased by 1. The sensitivity level is not permanent: over time it decrements at a rate given by a parameter γ_w, which indicates the probability of w_i being reduced by 1. This mechanism ensures that dissimilar nonself strings will still be detected, providing they occur in a short period of time

4.6 Costimulation

As mentioned in section 3.1.4 some self proteins are not found in the thymus, and so lymphocytes that are tolerized centrally in the thymus may bind to these proteins and cause an autoimmune reaction. It is also mentioned that this is not very common because T-cells require *costimulation* to be activated: In addition to binding to proteins (called *signal one*), a T-cell must be costimulated by a *second signal*. This second signal is usually a chemical signal which occurs when the body is damaged in some way. The second signal can come either from cells of the IS or other cells of the body. When a T-cell receives signal one but not signal two, it dies. Hence, auto reactive T-cells (those that bind to self) will be eliminated. in healthy tissues.

Similarly, it is not assumed in ARTIS that a detector will encounter every string $s \in S$ during its tolerization period, so it is possible that some detectors will mature that match some strings in S. ARTIS implements a basic form of costimulation. The second signal is provided by a human operator. When a detector d is activated by a string s, it sends a signal to a human operator, who is given a time period T_s (called the costimulation delay) in which to decide if s is really nonself. If the operator decides that s is really nonself, a second signal is returned to d. If the operator decides that s is actually self, no signal is sent to d and d dies off and is replaced by a new, immature detector. Consequently, a human operator need make no response in the case of false positives; the system will automatically correct itself to prevent similar false positives in future.

4.7 The Lifecycle of a Detector

If detectors lived forever and only died off when they failed to receive costimulation, most detectors would only be immature once. Any nonself strings that occurred during the period of immaturity of these detectors would not be detected in future because all detectors would be tolerant of them and would stay tolerant. In the IS this is not a problem because lymphocytes are typically short-lived (a few days) and so new, immature lymphocytes are always present, i.e. the population of lymphocytes is dynamic. ARTIS establishes a similar mechanism: each detector has a probability p_{death} of dying once it has matured. When it dies, it is replaced by a new randomly-generated, immature detector. In the end, every detector dies sooner or later, unless it is a memory detector. Figure 4.4 presents the lifecycle of a detector. Now a nonself string will only be undetected if it is continually present to tolerize the continual change of new detectors, i.e. the only false negatives will occur when nonself is frequent.

The memory detectors are not affected by this dying probability. In the IS, memory cells are long-lived so that the patterns that they represent are remembered over time. Similarly, memory detectors in ARTIS are long-lived. They can die only as a result of a lack of costimulation. In this case a second problem comes into mind. This problem is the becoming of all detectors memory detectors eventually, which would result in the loss of the advantages procided by dynamic detector populations. To solve this problem, ARTIS limits the number of memory detectors to some fraction m_d of the total detectors. If a new detector wins a competition and becomes a memory detector, and the fraction of memory detectors has reached the limit, then the least-recently-used (LRU) memory detector is changed back to an ordinary mature detector. The LRU detector is changed because the LRU detector is the one that has not been activated for the longest time period of any memory detector, and hence it is assumed to be the least useful memory detector.

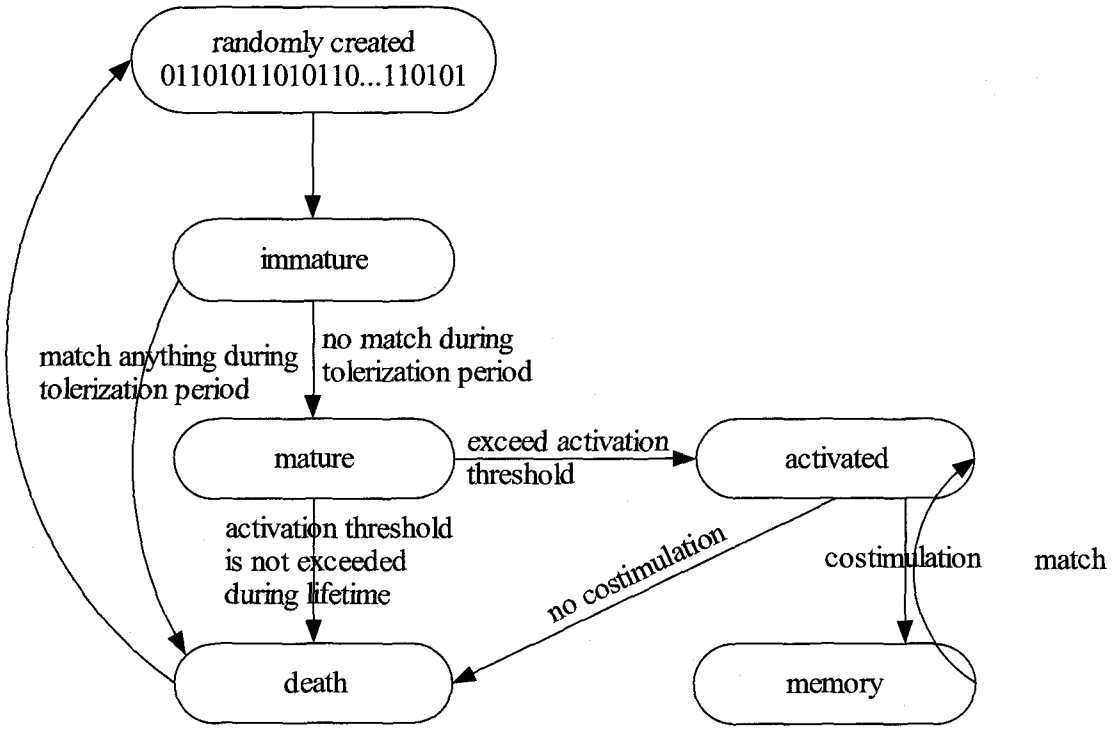

Figure 4.4 The lifecycle of a detector.

Figure 4.4 shows the lifecycle of a detector. A detector consists of a randomly created bit string that is immature for the tolerization period T. If it matches anything during that period it dies and is replaced by a new randomly generated detector. If it survives tolerization, it becomes a mature detector that lives for an expected $1/p_{death}$ time steps. If the detector accumulates enough matches to exceed the activation threshold τ, it is activated. If the activated detector does not receive costimulation, it dies. If it receives costimulation, it enters a competition to become a memory detector. When it becomes a memory detector, it lives forever, and only requires a single match for activation.

As a result of dynamic detector populations, the system can adapt to changing self sets. As the self set changes, it will tolerize new immature detectors, and mature detectors that were causing false positives will either die from lack of costimulation or from age. In the end all detectors will be tolerant of self, unless self does not change too quickly. If self changes very fast compared to the life span of a detector, mature detectors will continually die due to lack of costimulation and a large portion of detectors will be immature.

4.8 Representations

Molecules of the *major histocompatibility complex* (MHC) play an important role in the IS, because they carry peptides from the inner regions of a cell to the cell's surface. This enables IS cells to detect infections inside cells without entering in the cell membrane. There are many different types of MHC molecules, each of which binds a slightly different class of peptides. However, only a small set of these MHC types are present in each person. Therefore, individuals in a population are capable of recognizing different peptides, which provides population-level *diversity*.

MHC is guessed to plays an important role in protecting a population of individuals from holes in the detection coverage of nonself. A hole is a nonself string for which no valid detectors can be generated [50] (see Figure 4.5). Holes can exist for any approximate match rule with a constant probability of matching (such as the r-contiguous bits) [50].

The existence of holes is shown in Figure 4.5. There are strings in the nonself set that cannot be covered by valid negative detectors of a given specificity (match length r). The size of the dark circles representing detectors is a sign of the generality of those detectors. The detectors depicted in the example here are too general to match certain nonself strings without also matching self.

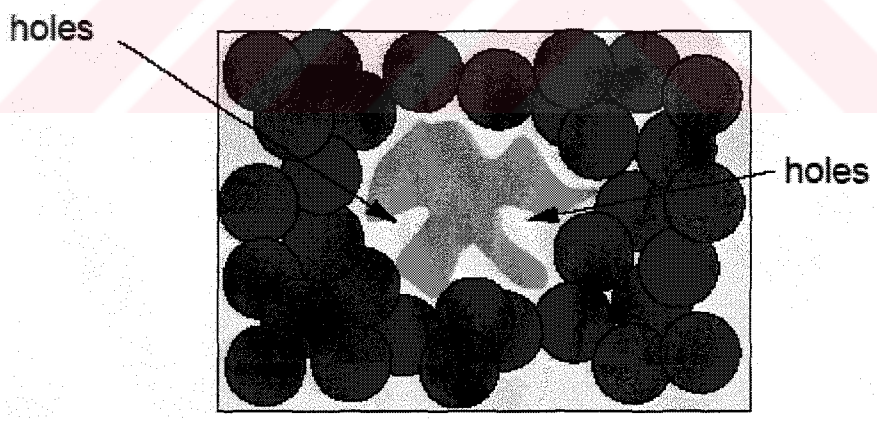

Figure 4.5 The existence of holes.

In the IS, each type of MHC can be regarded as a different way of representing a protein (depending on which peptides it presents); in effect, the IS uses multiple representations, or views, of proteins. Multiple representations can reduce the overall number of holes, because different representations will induce different holes. In ARTIS, each detection node uses a different representation: It filters incoming strings through a randomly-generated permutation mask. For example, given the

strings s_1 = 01101011; s_2 = 00010011, and a permutation, Λ, defined by the randomly-generated permutation mask 1-6-2-5-8-3-7-4, these strings become $\Lambda(s_1)$ = 00111110, and $\Lambda(s_2)$ = 00001011. Using the contiguous bits rule with r = 3, s_1 matches s_2, because the last 3 positions are the same, but under the new representation, $\Lambda(s_1)$ does not match $\Lambda(s_2)$. Having a different representation for each detection node is equivalent to changing the "shape" of the detectors, while keeping the "shape" of the self set constant (see Figure 4.6). Consequently, where one node fails to detect a nonself string, another node could succeed. Thus, different shaped detectors can cover different parts of the nonself space, for a global decrease in holes.

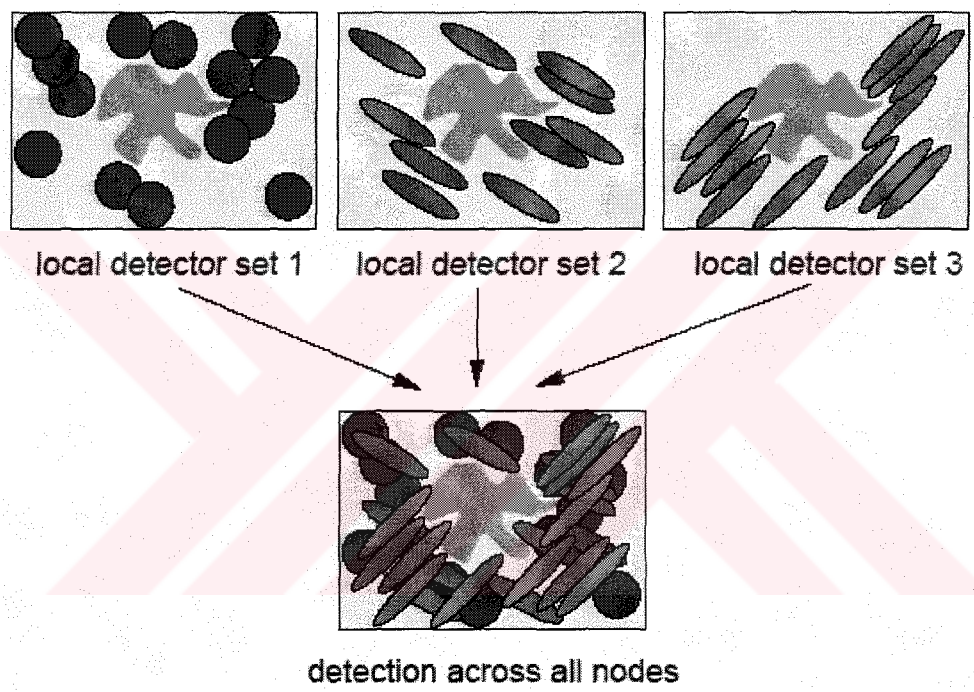

Figure 4.6 Using different representations to cover the holes.

4.9 Summary

Table 4.1 summarizes the differences and similarities between the IS and ARTIS. ARTIS does not use central tolerization and so have no equivalent of the thymus. The elements labeled "NONE" mean there is not a similar property in the ARTIS for the corresponding property in the IS.

Table 4.1 Comparison between the IS and ARTIS.

Immune System	ARTIS
peptide/protein/epitope	binary string
receptor	binary string
monoclonal lymphocyte	detector
(B-cell, T-cell)	
antibody variable region	detector string
antibody fixed region	bit string encoding response
memory cell	memory detector
pathogen	nonself binary string
binding	approximate string matching
locality	vertices in a graph
circulation	mobile detectors
central tolerization	NONE
thymus	NONE
MHC	representation parameters
cytokines	sensitivity level
peripheral tolerization	distributed negative selection
signal one	matches exceeds activation threshold
signal two(costimulation)	human operator
lymphocyte cloning	detector replication
pathogen detection	detection event
pathogen elimination	response
affinity maturation	memory detector competition

5. APPLICATION

The architecture of a generalized artificial immune system for anomaly detection (ARTIS) is described in the previous section. This architecture was proposed by Hofmeyr and Forrest, who also discussed the use of this system for network intrusion detection. The work, named LISYS, was designed to protect a local area network (LAN) from network-based attacks. LISYS was tested over seven intrusions and showed promising results. But, this quite limited input data made it necessary to evaluate whether LISYS is able to detect more varied intrusions than those used in [33,36]. This is stated as the main goal of the work in this thesis. Hence, this section provides an overview of the features of the LISYS and describes extensions and modifications to it, to evaluate it with a real-world dataset from DARPA.

A negative detection system consists of multiple negative detectors, each of which is represented by a string of length ℓ. These detectors are equivalent to lymphocytes in the immune system, and detection between a detector d and a string s is equivalent to receptor binding, which is modeled as partial string matching between s and the string representing d. In the described work Hofmeyr and Forrest used the contiguous bits rule. In this rule [14] two strings match if they have the same symbols in at least r contiguous positions (r is the match threshold). The detectors are called negative because they are generated so that they match nonself strings, and not self strings. Hence, in a negative detection system, self is defined in by defining nonself which is its complement.

The negative selection algorithm is used to generate the detectors, and is similar to negative selection in the thymus. During the training phase, candidate detectors are generated randomly and compared to all self strings. If a candidate detector matches any self strings, it is deleted and replaced by a new randomly generated detector, and the process is repeated. This results is a set of detectors that match nonself. Thus when a detector is activated, it is thought that a nonself string has been detected. In LISYS, the detectors can be kept on multiple computers and they do not need to communicate with each other. However, we do not need this property for evaluating the mechanism with a dataset because the evaluation will take place on a single computer.

5.1 Architecture

Section 4 has summarized an architectural model proposed by [33,36]. This model is also applied to network ID. In the network ID application datapath triples (source host IP address, destination host IP address and TCP port number) are used to represent the patterns. Then it is assumed that the set of all acceptable or allowed connections between computers both internally and externally is the self set, and everything else is nonself. The patterns or datapaths have been represented as binary strings of fixed length ℓ bits. This representation scheme is described in section 5.1.1.

As mentioned LISYS was designed to protect a LAN. To achieve this, the internal computers are mapped to n_L locations, and each location runs a local detection system. A local detection system, D_l, at location $l \in L$, consists of a set of detectors D_l, together with a function Λ_l implementing the secondary representation, and a match rule h_l, that is, $D_l = (D_l, \Lambda_l, h_l)$. All detector sets are the same size, that is, $|D_l| = |D_j| = n_d, \forall l, j = 1,2,...,n_l$. Furthermore, all detector sets use the same match rule, $h_l = h_j = h, \forall l, j = 1,2,...,n_l$. Each detector is a binary string of fixed length $\ell = 49$, drawn from U_R and tolerized using the negative selection algorithm. The detector generation phase is performed off-line, with all detectors being generated in a single location against a single training set. These detectors are then distributed across all locations. When the distributed system is monitoring, all local detection systems see the same traffic because the LAN is broadcast; thus, all locations have the same test set, $U_l = U_j, \forall l, j = 1,2,...,n_l$. Since all internal computers in the LAN run components of the detection system the system is distributed.

The above description of the system has many assumptions and generalizations some of which are inspired from the counterparts in the IS and some are specific to ID. For example, to imitate the distributed detection mechanism of the IS, this system is designed as a distributed system as mentioned above. One advantages of this approach is that a distributed network-based IDS delegates its responsibilities to a number of distributed components. This makes it more robust, extendible and scalable. This approach is also useful if the system is used to monitor different subnets of a network in real time, but in our work the evaluation is made offline on a previously captured dataset. Therefore, we do not need to implement it as a distributed system.

Since, we will not distribute the detection process to different locations we can not benefit from the use of different representations to cover the holes in the detection

space. So, the only solution to achieve this is to find the best detector specificity that minimizes the probability of holes while also minimizing the number of detectors required to provide a sufficient coverage in the nonself space.

LISYS is described as the implementation of the more general ARTIS system for the task of network intrusion detection. As stated in section 4.1 the first and most important step, when applying ARTIS to a domain, is to choose the equivalent of proteins. LISYS monitors network traffic, so its protein is a "datapath triple", which consists of a source internet protocol (IP) address, a destination IP address, and a TCP service (or port) by which two computers communicate. Since, this system is designed to monitor a broadcast subnet it is assumed that all internal computers will have IP addresses from the same class C network (IP addresses whose first 24 bits are equal which defines a total of 256 addresses). Because of this, LISYS represents IP addresses of the internal hosts with the last byte of their IP addresses as discussed in detail below. Hence, it uses detectors of length $l = 49$ but this is not suitable for the evaluation of DARPA dataset since there are internal computers that reside in different C class subnets. Therefore, an extension is required to the model, which resulted in the expansion of the detector string length ℓ to $\ell = 57$. This extension to the base representation is also described in section 5.1.1.

5.1.1 Base Representation

A SYN packet representing a datapath triple is mapped to a 49-bit binary string (see Figure 5.1) in the case of a C class broadcast network. The value of 49 bits was chosen as the minimum number needed to represent the relevant information. The composition of the string is as follows. Because at least one computer in the datapath must be on the LAN, the first 8 bits represent an internal computer, and are taken from the least significant byte of its IP address. The following 32 bits represent the other computer involved in the communication, which will require 32 bits for the IP address if that computer is external. If the other computer is internal, then only 8 bits are needed, but 32 bits (the full IP address) are still used to have a fixed length representation. If an external computer is involved, it is always represented in the second 32 bits, so an extra bit is used to indicate whether or not the first computer is the server; if the first is the server, the server bit is set to one, otherwise it is set to zero.

The final 8 bits represent the type of service. The service type is mapped to a number from 0 to 255 to reduce the length of representation. All non-assigned privileged ports are represented by a single number, and all nonassigned non-privileged ports

are represented by a different, single number. In the following list, ports are numbered in order, beginning with commonly assigned, privileged ports, followed by commonly assigned non-privileged ports, and finally with a single number for all other privileged ports and a single number for all other non-privileged ports.

- Commonly assigned privileged ports (numbered from 0 to 66): 1, 7, 9, 11, 13, 19, 20, 21, 22, 23, 25, 37, 38, 42, 43, 53, 68, 70, 79, 80, 87, 94, 95, 109, 110, 111, 113, 119, 123, 130, 131, 132, 137, 138, 139, 143, 156, 161, 162, 177, 178, 194, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 210, 213, 220, 372, 387, 396, 411, 443, 512, 513, 514, 515, 523, 540, 566
- All other privileged ports: (numbered 67) 1023
- All other non-privileged ports: (numbered 68) 1024
- Commonly Assigned Non-privileged ports (numbered 69): 6000–6063

There are only 69 numbered services, which means only 7 bits are required for their representation, not 8 bits as shown in Figure 5.1 but the LISYS keeps an extra bit to enable adding more services.

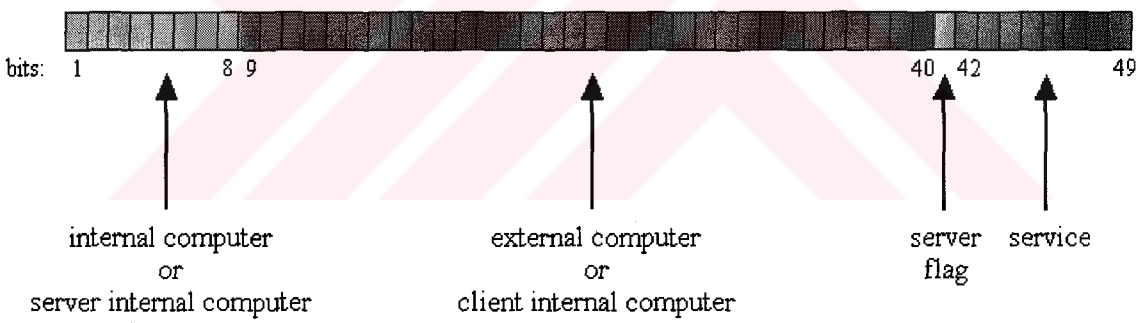

Figure 5.1 Base representation of a TCP SYN packet.

This simplification of the port numbers shortens the base representation, and thus reduces the probability of acceptable new datapaths occurring in the test set. This is important because if the probability of new acceptable patterns in the test set increases the false positive rate will also increase. It is claimed that, the simpler the data being monitored the better [33].

The DARPA dataset consists of internal computers that reside in different C class subnets of a B class network. Therefore, using such a representation scheme is not appropriate for it because there will be collisions on the first 8 bits of the strings representing different datapath triples. To avoid this problem, an extension is made

to the model, which expands the detector string length ℓ to $\ell = 57$. This new representation scheme is shown in Figure 5.2.

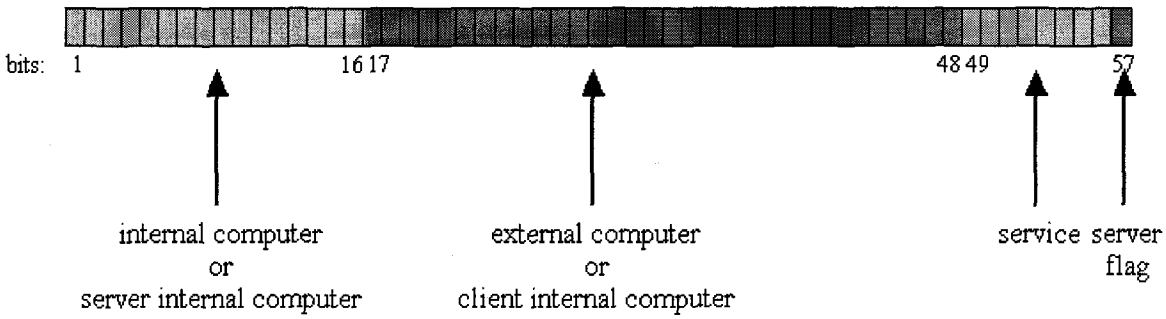

Figure 5.2 The new base representation of a TCP SYN packet.

5.1.2 Secondary Representations

Three types of secondary representation schemes were described by Hofmeyr and Forrest: pure permutation, substring hashing, and imperfect hashing. Each of these defines a parametric family of representations.

A representation Λ is a function mapping a pattern in U to a string of length ℓ in U_R, that is, $\Lambda : U \to U_R$. The size of U_R is then 2^ℓ. Since U_R is a representation of U, it could be that $|U_R| < |U|$. S will then be represented by the set of strings S_R and N will be represented by the set of strings N_R. Under the representation, $N_R \cap S_R = \varnothing$ and $N_R \cup S_R = U_R$, will hold if Λ is a one-to-one mapping and, $S \bigcup N = U$ and $S \bigcap N = \varnothing$.

The same family, Λ_{family} is used for all local detection systems, but the parameter sets Φ are randomly determined, therefore each local detection system has a different secondary representation, determined by $\Lambda_{family}(\Phi)$. The parameters for pure permutation scheme are $\Phi = \{x\}$, where $x = x_1 x_2 ... x_{49}$ is a permutation mask, that is, each x_i is an integer, $1 \le x_i \le 49$, representing the position to which the current bit value must be mapped. For example, if $x_3 = 10$, then the bit in position 3 must be mapped to position 10. x is randomly generated such that $x_i \neq x_j, \forall i, j = 1, 2, ..., 49$.

As mentioned previously holes can exist for any approximate match rule with constant matching probability. This results in a lower bound on the false negative probability. However, different matching rules generate different holes for the same self set, so it was suggested in [50] that using different matching rules for different detectors could reduce the overall number of holes. However, it is easier to use a

single base representation. Therefore, one match rule could be used and the representation changed. Hofmeyr investigated the use of two different match rules: Hamming distance match and r-contiguous match rules along with three secondary representation schemes: pure permutation, substring hashing, and imperfect hashing. According to this, using the r-contiguous match rule with the pure permutation representation schemes is more suitable for this application domain. Therefore the work in this thesis only implements these features of the originally proposed system. Moreover, since the system is not a distributed system any more, the use of different secondary representation schemes is not useful, but we can still use a single secondary representation scheme which is supposed to have no effect on the overall performance of the system. To provide the secondary representation feature of the LISYS as an add-on only the pure permutation scheme is implemented in this work.

5.1.3 Activation Thresholds and Sensitivity Levels

If the training set does not include all self strings this will result in false positives, which, cannot be tolerated if the system is required to be scalable. If a complete training set cannot be collected, then false positives must be minimized in some other way. Two mechanisms are introduced by LISYS to reduce false positives. These mechanisms are based on two assumptions:

1. An incident will generate multiple nonself strings.

2. When an incident occurs, it will generate nonself strings at a greater rate than the rate of occurrence of new self strings, over some given period of time.

These assumptions mean that nonself strings are more likely to occur in sequential clusters relative to new self strings. As the number of connections used by an attacker is reduced, so the difficulty of attacking increases, because the more connections there are, the more information an attacker can gain about the target network. Furthermore, the more the attacks are separated in time, the longer it will take to execute an attack, requiring more patience and possibly skill from an attacker. The two mechanisms for reducing false positives are:

Activation thresholds: This mechanism is based on the concept of affinity thresholds in the immune system: multiple receptors on a lymphocyte must bind to epitopes before that lymphocyte is activated. A similar mechanism is used in which each detector has to match a certain number of times before it is activated and signals an anomaly. Each detector $d \in D_l$ has a match count c_d, which is incremented every time d matches a string. When c_d exceeds the activation threshold, τ, an anomaly is

signaled and the counter for d is reset to 0, that is, $c_d = 0$. The activation threshold, τ, is a global parameter, that is, it is the same for all locations. The match count c_d is not permanent, it decays at a rate determined by a decay parameter, γ_{match}, which indicates the probability of the match count decreasing by one. The effect of this mechanism is that anomalies will be detected only if they are caused by groups of similar strings that occur frequently within a limited time period.

Sensitization: This mechanism is intended to improve detection of distributed attacks. Activation thresholds can not solve this problem, because they require that a single detector match repeatedly, and if the attacks are launched from different sources, a single detector may not match all the attacks. This additional mechanism is intended to "sensitize" the detection system so that an increase of connections from multiple different locations will generate anomalies. This is similar to the effect of cytokine levels in the IS. When suspicious activity is detected by some cells of the IS, cytokines are released which alert the rest of the IS. Each detection system D_l has a sensitivity level σ_l, which is a non-negative real. Every time the match count for a local detector $d \in D_l$ goes from 0 to 1, the sensitivity level is increased by a factor w, that is, $c_d(t) = 0$ and $c_d(t+1) = 1 \Rightarrow \sigma_l(t+1) = \sigma_l(t) + w$ for $t = 1, 2, \ldots$. This sensitivity level is not permanent too. The sensitivity σ_l decays over time with a decay parameter, γ_w, which is the probability that the sensitivity will decrease by 1. The sensitivity affects local detectors by reducing the number of matches needed for activation: detectors at location l will only need $\tau - \sigma_l$ matches to be activated. The sensitivity can never reduce the activation threshold below one, that is $0 \le \sigma_l < \tau, \forall l = 1, 2, \ldots n_L$. The effect of this mechanism is that anomalies occurring within a limited time period, even if they involve different strings, can still be detected. All of the above mentioned parameters of the system and their default values are given in Table 5.1.

Table 5.1 Parameters of the system

Parameter	Description	Default Value
n_L	number local detection systems	50
n_d	number detectors per location	100
ℓ	string length	49
r	match length	12
Λ	secondary representation	substring hashing
h	match rule	r-contiguous bits
τ	activation threshold	1
γ_{match}	match decay	0
w	sensitivity	1.5
γ_w	sensitivity decay	10

5.2 Dataset

Ideally an IDS should be evaluated on a real network and tested with real attacks. Unfortunately it is difficult to repeat such tests so that other researchers can replicate the evaluation. To do that, the network traffic would have to be captured and reused. This raises privacy concerns, because real traffic can contain sensitive information such as email messages and passwords.

The DARPA/Lincoln Laboratory IDS evaluation (IDEVAL) data sets [51,52] overcome this difficulty by using synthetic background traffic. This project had two goals. First, the goal was to test a wide variety of systems (host or network, signature or anomaly, four different operating systems) on a wide range of attacks. The second goal was to provide off-line data to help development of new systems and algorithms by publishing a standard benchmark so that researchers could compare systems and replicate results.

Evaluations were conducted in 1998 and 1999. The 1999 evaluation improved on the 1998 evaluation by simplifying the scoring procedure, providing attack-free data to train anomaly detection systems, adding many new attacks, and one new target (Windows NT) to the three UNIX based targets in 1998.

Figure 5.3 shows the configuration of the 1999 evaluation test bed. A local area network is set up to resemble a portion of a typical Air Force base network. There are four main "victim" machines, running SunOS, Solaris, Linux, and Windows NT. Traffic generators simulate hundreds of other hosts and users running various applications and an Internet connection [53]. The mix of protocols (HTTP, SMTP, telnet, etc.) and hourly variations in traffic volume are similar to traffic collected on a real Air Force network in 1998.

The evaluation data set is collected from the four victim machines and from two network sniffers, an “inside” sniffer between the router and the victims, and an “outside” sniffer between the router and the Internet. The host based data consists of audit logs and nightly file system dumps and directory listings, in addition to Solaris Basic Security Module (BSM) system call traces. This data is analyzed off-line to detect attacks. Attacks may originate from either the Internet, from compromised hosts on the local network, or from attackers who have physical access to the victims or the local network. Most attacks are against the four main victim machines, but

some are against the network, the Cisco router, or against simulated hosts on the local network.

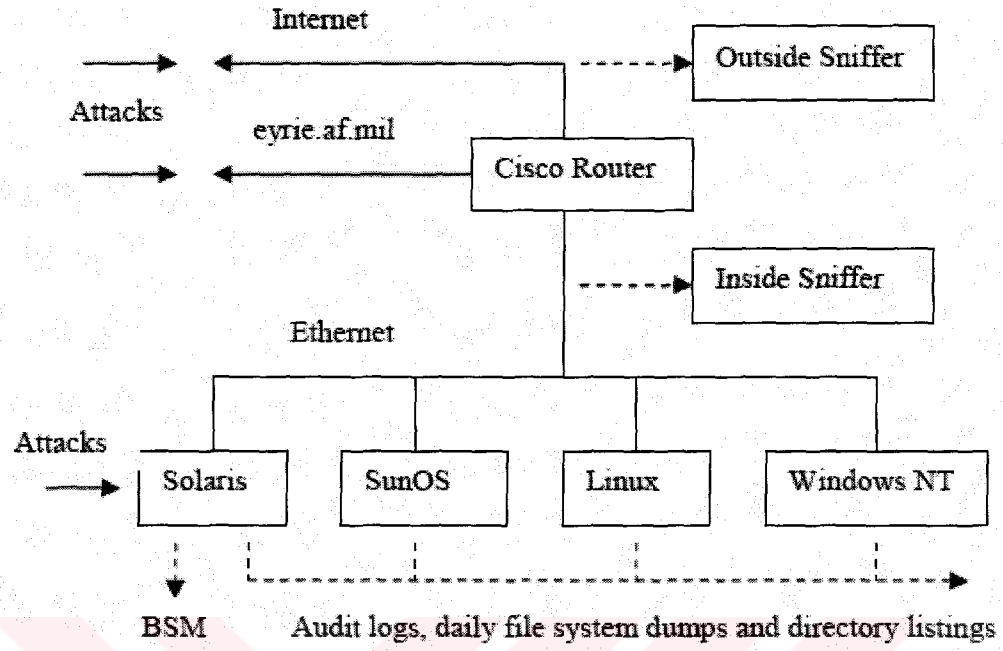

Figure 5.3 The 1999 DARPA/LL IDS Evaluation Test Configuration

The 1999 evaluation had two phases separated by about three months. During the first phase, participants were provided with three weeks of data (collected Monday through Friday, 8:00 AM to 6:00 AM local time each day). The second week of data contained 43 labeled instances of 18 attacks which were taken from the Bugtraq mailing list, from published sources such as www.rootshell.com, or which were developed for the evaluation. Attacks were sometimes made stealthy or hard to detect, for example, by slowing down a port scan or obfuscating suspicious shell commands. Attacks were labeled with the start time and the name of the victim. The first and third weeks contained no attacks, and could be used to train anomaly detection systems.

During the second phase, participants were provided with two weeks of test data (weeks 4 and 5) containing 201 unlabeled instances of 58 attacks, 40 of which were not in the training data. Participants were required to provide a list of alarms identifying the target address, time, and a numerical score representing a confidence level in the alarm, and optionally, the type of attack. Participants also provided a system specification which describes the types of attacks their system is designed to detect. Attacks are classified by category (probe, DOS, R2L, U2R), the type of data examined (inside sniffer, outside sniffer, BSM, audit logs, file system dumps, or directory listings), victim operating system (SunOS, Solaris, Linux, or NT), and

whether the attack is new (not in week 2) (see Appendix A for more details). Systems are evaluated by the fraction of attacks detected out of those they are designed to detect at various false alarm rates (say, 10 per day, or 100 total) by ranking the alarms by score and discarding those which fall below a threshold that would allow more false alarms. An attack is counted as detected if there is an alarm with a score above the threshold that identifies the victim address (or any address if there is more than one) and the time of any portion of the attack with 60 seconds tolerance before the start of the attack or after the end. Alarms that detect attacks that the system is not designed to detect (out of spec attacks), and duplicate alarms detecting the same attack are discarded without penalty. Any other alarm is considered a false alarm.

Eight organizations participated in the 1999 evaluation, submitting 18 systems. The top four systems reported by [52] are shown in Table 5.2. The best systems detected about half of the attacks they were designed to detect at a false alarm rate of 10 per day. Subsequent to the evaluation, the five weeks of training and test data were released to encourage research in intrusion detection.

Table 5.2 Top four systems in the DARPA 1999 evaluation

System	In-Spec Attacks	Detected at 10 false alarms per day
Expert 1 (EMERALD)	169	85 (50%)
Expert 2 (STAT)	173	81 (47%)
Dmine (ADAM)	102	41 (40%)
Forensics (DERBI)	27	15 (55%)

Of the 58 attack types, there were 21 that were classified as poorly detected. None of the 18 systems detected more than half of the instances of any of these attacks. Often these were missed because the systems did not monitor the relevant protocols or because they were new attacks not found in the labeled training data. The attacks are as follows:

- Stealthy *ipsweep* (probe, slow ECHO REQUEST scan for active IP addresses).
- Stealthy *portsweep* (probe, slow port scan, or using stealthy techniques such as FIN scanning).
- *ls_domain* (probe, obtains a list of hosts using a DNS zone transfer).
- *queso* (probe, operating system fingerprinting using malformed TCP packets).

- *resetscan* (probe, port scan using unsolicited RST packets).
- *arppoison* (DOS, disrupting local traffic by forging replies to ARP-WHO-HAS packets).
- *dosnuke* (DOS, crashes Windows by sending urgent data in a NetBIOS request).
- *selfping* (DOS, crashes SunOS by sending an ICMP ECHO REQUEST to self).
- *tcpreset* (DOS, disrupts local traffic by forging RST packets to close TCP connections).
- *warezclient* (DOS, downloading illegal software from a local anonymous FTP server).
- *ncftp* (R2L, FTP client bug exploit).
- *netbus* (R2L, a backdoor server).
- *netcat* (R2L, another backdoor, uses a stealthy DNS channel).
- *snmpget* (R2L, exploits an easily guessed router password).
- *sshtrojan* (R2L, fake SSH server upgrade with login backdoor).
- *loadmodule* (U2R, exploits a trojan shared library to gain root).
- *ntfsdos* (U2R, an attacker with console access copies the disk, bypassing file system protection).
- *perl* (U2R, exploits a bug in a setuid root Perl script).
- *sechole* (U2R, exploits a bug in Windows NT).
- *sqlattack* (U2R, a restricted user escapes from a database application to the UNIX shell).
- *xterm* (U2R, gains root using a buffer overflow).

Out of 201 attack instances, 77 are poorly detected. Only 15 instances were detected by any system.

5.3 Implementation Details

The system described above is implemented in C++ programming language using the object-oriented programming approach. The program consists of different classes each implementing a different function of the anomaly detection operation. The primary class of the program is the IDIS class (named after the acronym for Intrusion Detection Immune System). This class is the driver of the program. It handles the detectors, permutations, match operations and realizes the negative selection algorithm depicted in Figure 4.3. It also keeps the detection parameters such as the number of detectors (n_d), detector length (l), match length (r), etc. The properties of the IDIS class are shown in the Table 5.3.

Table 5.3 Properties of the IDIS class

Name	Type	Description
nNumDetectors	int	Number of detectors in the system
mMatchRule	ContiguousMatchesMatchRule	The class that implements the r-contiguous match operation.
nBipLength	int	Detector length
nRandomSeed	long	Seed to use with the random number generator
bUseMemory	bool	Whether to use memory detectors
nMaxMemoryDetectors	int	Maximum number of memory detectors
nTolerizationPeriod	int	Duration of the Tolerization period (as packet count)
nActivationThreshhold	int	Threshold value for the detectors to raise alarms after that many occurrences of an abnormal string encountered.
nCostimulationDelay	int	Duration before a detector dies from lack of costimulation.
dDeathProbability	double	Probability of a detector dying randomly.
dMatchDecay	double	Probability of a detectors match count being reduced.
nMinMatchLength	int	Minimum match length for the r-contiguous match rule.
nNumMemoryDetectors	int	Current number of memory detectors.
dSensitivityLevel	double	Threshold adjustment factor.
dSensitivityDecay	double	Sensitivity level decrement factor.
dSensitivityIncrement	double	Sensitivity level increment factor.
vDetectors	vector<Detector>	Vector of Detectors.
random	Random	Random number generator.
bif	PurePermutationBIF	Filter class that implements the pure permutation representation scheme.
nNumBipsReceived	long	Number of the strings received by IDIS.
nNumAnomalies	long	Number of anomalies detected.
nNumDetectorsKilled	long	Number of detectors killed for various reasons.

The Detector Class implements the functional properties of a detector in the negative selection system. Detector class keeps some local variables to represent the different states of a detector. For example, a detector can be in the states of immature, mature, activated, awaiting costimulation, memory, etc. Each of these states is represented with different Boolean variables. The properties of the Detector class are shown in the Table 5.4.

Table 5.4 Properties of the Detector class

Name	Type	Description
NONE	int	Static variables used to represent different events in member functions.
DIE	int	
ADJUST_SENSITIVITY	int	
ACTIVATED	int	
random	Random *	Reference to the random number generator of the program.
memory	bool	State variable representing whether the detector is a memory detector or not.
awaitingCostimulation	bool	State variable representing whether the detector is awaiting costimulation or not.
immature	bool	State variable representing whether the detector is still immature or not.
age	long	Age of the detector. (As packet count)
numberMatches	int	Number of strings that this detector has matched so far.
ageAtLastMatch	long	Age of that detector at the last match with a string.
localBif	PurePermutationBIF	Filter class that implements the pure permutation representation scheme.
localBip	TCPDump57BitBIP	The binary representation string of the detector.
localMatchRule	ContiguousMatchesMatchRule *	Reference to the class that implements the r-contiguous match operation.
activated	bool	State variable representing whether the detector is activated (matched) or not.
currentMatchLength	int	The length of the match between the detector and the data under the r-contiguous match rule.
_nActivationThreshhold	int	Threshold value for the detector to raise alarms after that many occurrences of an abnormal string encountered.
_nCostimulationDelay	int	Duration before the detector dies from lack of costimulation.
_nTolerizationPeriod	int	Duration of the Tolerization period that detector learns via negative selection algorithm. (as packet count)
_dDeathProbability	double	Probability of the detectors dying randomly.
_dMatchDecay	double	Probability of the detectors match count being reduced.
_nMinMatchLength	int	Minimum match length for the r-contiguous match rule.

TCPDump57BitBIP class is responsible for the string representation described in section 5.1.1. It is used for both representing each incoming packet and also for the binary string of each detector. While the detector strings are created randomly, the incoming packets are encoded using the class utility functions as shown in Figure 5.2. Table 5.5 provides the properties of the TCPDump57BitBIP class.

Table 5.5 Properties of the TCPDump57BitBIP class

Name	Type	Description
binaryString	BitSet<>	57 Bit binary string used to encode the information.
NUMBER_OF_BITS	int	Number of bits in the string (57)
NUM_PACKED_BYTES	int	Number of bytes used to encode the bit string (7)
localIPMask	char *	Initial two bytes of the local IP addresses (172.16)
packedBytes	char *	Temporary space used in the encoding phase.
serverFlag	bool	1 bit flag representing the direction of the connection.
tcpDumpString	string *	ASCII string representation of the information.
constructorString	string	Temporary variable to keep the incoming string information
datestamp	string	Date of the connection
timestamp	string	Time of the connection

ContiguousMatchesMatchRule class implements the r-contiguous match operation. It has a utility function *match*, which takes two arguments of the type TCPDump57BitBIP and compares them according to the r-contiguous match rule described in Section 4.2.

PurePermutationBIF class is used for changing the representation of the incoming binary strings by using a random permutation mask as specified in Section 4.8.

Random class constitutes the random number generator of the implemented system. Since the detector generation is done randomly, the random number generation is very important for the overall performance of the system. There are different methods to generate random numbers. One of these methods is the use of the standard random number generation function *rand* in the C++ library. Another method would be using another proprietary library for this task but both of these approaches are heavily dependent on the system. Therefore, the need for a standard and reliable random number generator arises. This task is accomplished by implementing the random number generation algorithm from Knuth [54] as detailed in [55].

As the system is tested offline on DARPA dataset and the attacks are previously known the costimulation mechanism is implemented as an internal module that looks up the generated alarms from an attack table and costimulates an alarm if it is in the known attacks table. This module is also used as an external module for the evaluation of the simulation results later.

6. EXPERIMENTAL RESULTS

Although previous work using a negative selection algorithm for anomaly detection [16,33] showed promising results, there had been little effort to apply this algorithm on vast amounts of data. One distinctive feature of a network intrusion detection problem is that the size of data, which defines "self" and "nonself", is enormous. In order for this algorithm to be adapted to a network-based IDS, it is important to understand whether this algorithm is capable of generating detectors in a reasonable computing time. In addition, it is essential to examine whether its tuning method, which derives an appropriate number of detectors to gain a good nonself detection rate, works when it is used on the huge size of real network data. Therefore, a series of experiments were performed to investigate these two significant features of the negative selection algorithm.

The following tests are performed on the preprocessed DARPA/Lincoln Laboratory IDS off-line evaluation (IDEVAL) data set. This dataset consists of two different network traffic captures for each weekday of a five weeks period. These are inside and outside traffic captures recorded by two sniffers located in and out of the test network. The tests are carried out on the internal traffic dumps. This traffic dumps are in the form of tcpdump output files and size in the range of 161MB to 1GB. Since this system is designed to monitor only the TCP connection initiation packets (aka. SYN packets) the dataset is preprocessed and the interested fields (connection date and time, source and destination IP addresses and TCP port numbers) of the TCP SYN packets are extracted. This preprocessing is done using the tcpdump utility freely available on UNIX systems. The resulting output files only consisted of the connection date and time, source and destination IP addresses and TCP port numbers. All the output files for the weeks 1, 3, 4 and 5 are concatenated to form a single input file to the program. The data for the weeks 1 and 3 is used as the training data. Since week 2 data has some attacks it is not used as the training data. Weeks 4, 5 are the original test periods of the DARPA IDS evaluation. Therefore, they are also used as the test data in this work.

The negative selection algorithm implemented in this work has many different parameters affecting its overall performance in anomaly detection. This work investigates the effect of these parameters on real world data and tries to derive

appropriate values for them. Therefore a number of different tests are conducted on different parameters.

6.1 Number of Detectors

One of the most important features of the negative selection algorithm is the number of detectors used to represent nonself data in the data space. As the detectors are generated during the training phase, not to match the self patterns, they represent possible nonself patterns that would be encountered in the test data. As a consequence of the use of an approximate match rule to provide generalized detection each detector represents a subset of the nonself patterns in the data space (see section 4 for more details). Thus, the more the number of detectors, the more nonself patterns are represented and detectable by the system. To investigate this effect the system is tested with different number of detectors in the range 100 to 8000. The test results are provided below as the number of detections in the test data.

The tables below provide the output of external evaluation module that processes the output files of the program and generates the tables showing detection rates for different types of attacks targeting different machines. The structure of the tables is as follows: Columns show different categories of attacks. Column 2 is the all attacks. Columns 3-7 are different attack categories (see Appendix A). Columns 8 and 9 show the new attacks that are not found in week 2 and the stealthy attacks that are crafted specially to avoid detection by intrusion detection systems. Rows, on the other hand shows the information summarized in Table 6.1.

Table 6.1 Rows of the eval output tables.

W45	Attacks in Weeks 4-5
IT	Attacks with evidence in the inside sniffer capture files
OT	Attacks with evidence in the outside sniffer capture files
BSM	Attacks with evidence in the Solaris BSM files
NT	Attacks with evidence in the NT audit logs
FS	Attacks with evidence in the File system dumps
pascal	Attacks with the Solaris target
hume	Attacks with the NT target
zeno	Attacks with the SunOS target
marx	Attacks with the Linux target
Poor	Attacks that are not detected by many of the systems participated in IDEVAL.
W2	Attacks in week two

Table 6.2 shows the results for 100 detectors with a match threshold of 12. As seen from the table the system detected 4 of the total 201 attacks, 177 of which has evidence in the inside sniffer traffic data that the system is tested on. However, not all the attacks are identifiable by just looking at the connection identifiers (addresses and port numbers) of the packets. Still, this result is not reasonable and efficient for a network intrusion detection system. Actually, the system produces a total of 41 alarms when the number of detectors is 100. 22 of which correspond to 4 different attacks and 19 are false alarms. This means a false alarm rate of 46% which is relatively low compared to other anomaly detection systems.

Table 6.2 Detection results for 100 detectors

	All	Probe	DOS	R2L	U2R	Data	New	Stealthy
W45	4/201	0/37	1/65	1/56	2/37	0/16	2/62	0/36
IT	4/177	0/34	1/60	1/54	2/27	0/7	2/52	0/30
OT	2/151	0/32	0/44	1/46	1/26	0/11	1/38	0/23
BSM	1/38	0/1	0/12	1/10	0/11	0/6	0/8	0/6
NT	2/33	0/3	0/7	0/10	2/12	0/4	2/26	0/0
FS	4/189	0/37	1/62	1/56	2/31	0/11	2/54	0/34
pascal	1/55	0/8	0/20	1/12	0/11	0/6	0/11	0/9
hume	3/48	0/7	1/15	0/12	2/13	0/5	2/31	0/2
zeno	0/22	0/7	0/9	0/3	0/3	0/1	0/2	0/6
marx	0/44	0/6	0/17	0/18	0/2	0/2	0/11	0/10
Poor	0/72	0/21	0/17	0/15	0/18	0/7	0/38	0/29
W2	0/43	0/9	0/13	0/6	0/12	0/3	0/0	0/0

The obtained results are very poor for 100 detectors. Since the aim of this experiment was to investigate the effects of the number of detectors, the system is tested with different number of detectors which the results are provided below.

Table 6.3 shows the results for 8000 detectors with a match threshold of 12. As seen from the table the system detected 40 of the total 177 attacks identifiable from the test data. The system produced a total of 1535 alarms. 1154 of which correspond to 40 different attacks and 381 are false alarms.

Table 6.3 Detection results for 8000 detectors

	All	Probe	DOS	R2L	U2R	Data	New	Stealthy
W45	40/201	5/37	6/65	23/56	4/37	2/16	14/62	7/36
IT	39/177	5/34	6/60	23/54	4/27	1/7	13/52	7/30
OT	31/151	5/32	3/44	19/46	3/26	1/11	9/38	5/23
BSM	9/38	0/1	1/12	5/10	2/11	1/6	1/8	2/6
NT	8/33	1/3	0/7	4/10	2/12	1/4	7/26	0/0
FS	39/189	5/37	6/62	23/56	4/31	1/11	13/54	7/34
pascal	11/55	0/8	2/20	6/12	2/11	1/6	2/11	2/9
hume	11/48	3/7	1/15	4/12	2/13	1/5	7/31	2/2
zeno	6/22	1/7	3/9	2/3	0/3	0/1	1/2	0/6
marx	9/44	1/6	0/17	8/18	0/2	0/2	4/11	3/10
Poor	11/72	2/21	2/17	6/15	0/18	1/7	9/38	5/29
W2	0/43	0/9	0/13	0/6	0/12	0/3	0/0	0/0

Increasing the number of detectors from 100 to 8000 resulted in an increase in the number of detections. The system has detected a total of 40 attacks with 8000 detectors. As expected the detection rate increases as we increase the number of detectors since we are covering the nonself space more accurately.

The effect of the number of detectors to the performance of the system is shown in Table 6.4. As we increase the number of detectors from 100 to 8000 we see an increase in the number of detections from 4 for 100 to 40 for 8000. This increase is also depicted in Figure 6.1. The number of detections rises quickly with the number of detectors and stabilizes after a certain point. Since the total number of attacks identifiable by this system is limited, the detection count should not increase more than a certain amount.

Table 6.4 provides the total, true and false alarm counts of the system as well. The more the number of detectors, the more alarms are produced by the system. However, the false positive rates are always lower than 46%. Despite this rate may seem high, the total false alarm counts are ranging between 19 and 381 at which means an average of 2 to 38 false alarms a day for the two weeks test period. If we use a false alarm filter to filter out the alarms of the system after the first 100 false alarms we still get reasonable results for 8000 detectors. In that case the system detects a total of 34 attacks producing 1023 true and 100 false alarms.

Table 6.4 Detection results and alarm counts for different number of detectors

Detectors	Detections	Alarms	True Alarms	False Alarms
100	4	41	22	19
200	9	92	58	34
400	17	254	175	70
600	20	290	191	99
800	24	838	725	113
1000	27	873	736	137
1200	29	891	742	149
1600	30	935	748	187
2000	34	1254	1039	215
2400	36	1349	1115	234
2800	36	1369	1126	243
3200	36	1383	1126	257
3600	36	1401	1126	275
4000	36	1413	1126	287
5000	36	1440	1127	313
6000	37	1475	1128	347
7000	38	1518	1152	366
8000	40	1535	1154	381

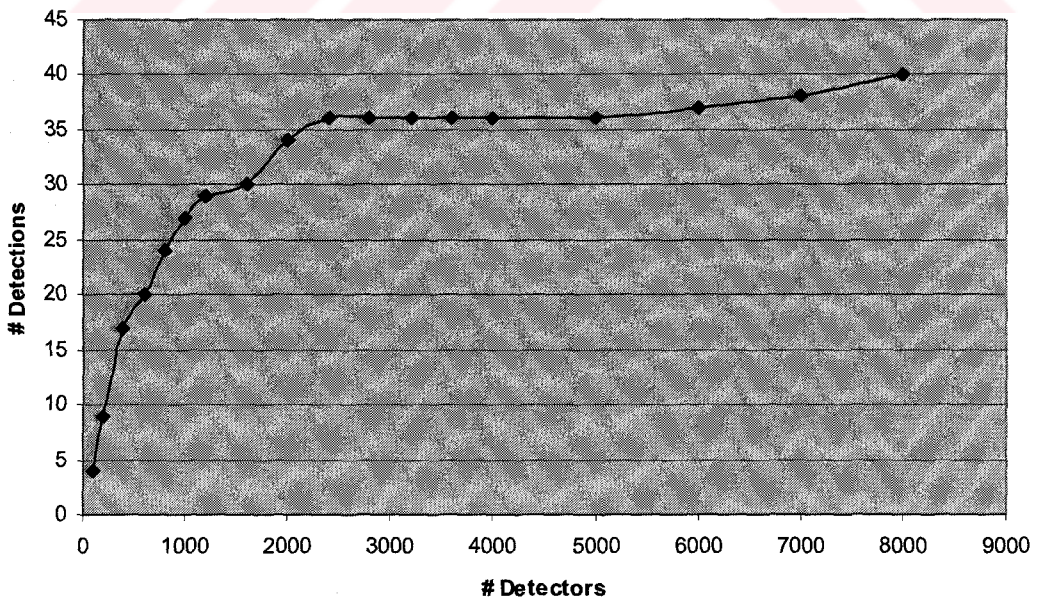

Figure 6.1 Number of detections against number of detectors

The matching probability, p_M, can be used to compute the relationship between the number of detectors and the probability of a false negative error, p_{ε^-}. Assuming that the detectors are drawn randomly from U, and the probability of one detector matching is independent of the other matching, then the probability that a random nonself string $s \in N_R$ is a false negative, ε^-, is [16]

$$P(\varepsilon^-) = p_{\varepsilon^-} = (1 - p_M)^{\eta} \tag{6.1}$$

For example, if $l = 57$ and $r = 12$, under the contiguous bits rule, $p_M = 0.006$, and a false negative probability of $p_{\varepsilon^-} = 0.032$ is achieved with $\eta = 600$ detectors. This number of detectors is small compared to the $2^{57} = 144115188075855872$ possible strings, most of which could be nonself. Combining detectors in this way exhibits a useful property: the probability of an error decreases exponentially with an increase in the number of detectors. In the previous example, if the number of detectors is doubled to $\eta = 1200$, the error probability drops by two orders of magnitude to $p_{\varepsilon^-} = 0.001$.

Equation 6.1 holds only if the probability of a random nonself string being matched by one detector is independent of the probability of matching by other detectors. This is approximately true if detectors are strings randomly drawn from U, and the self set is small in comparison to the universe, $S \ll U$, because then $N \approx U$ and hence the negative selection algorithm will not bias the detectors towards certain strings. Note that equation 6.1 is derived for the case where the nonself string is randomly drawn. If a test set contains a random subset of nonself strings, then this assumption is approximately true, but if the test set contains a biased subset of nonself strings, then this result will not hold.

A consequence of this implementation of generalization is that there is a computational trade-off between the number of detectors, η, in a set and the number of retries, ρ, needed to generate that set. As the match length r increases, the probability of a match p_M decreases (for both match rules) and hence the number of retries decreases, but there is a corresponding increase in the number of detectors required for a given false negative error probability. The number of detectors required can be given as (6.2) if we assume that $E(\rho)$ denotes the number of retries required to generate a valid detector.

$$\eta = -|s| \frac{\log p_{\varepsilon^-}}{\log E(\rho)} \tag{6.2}$$

The test results also show that trade-off. Table 6.5 shows total and average number of retries to generate different number of mature detectors. For example, if the system is run with 100 detectors 594 retries are made during this run and a total of 83 mature detectors are generated. However, if the system is run with 8000 detectors 51208 retries are made during this run and a total of 6395 mature detectors are generated. The average number of retries to generate a single mature detector ranges between 7.13 and 8.14. On the average, approximately 8 retries are required per mature detector.

Table 6.5 Number of retries to generate different numbers of mature detectors.

Detectors	Retries	Mature Detectors	Average Retries
100	594	83	7,16
200	1183	166	7,13
400	2407	331	7,27
600	3657	483	7,57
800	4920	653	7,53
1000	6380	806	7,92
1200	7591	967	7,85
1600	10287	1283	8,02
2000	12642	1599	7,91
2400	15490	1904	8,14
2800	18193	2251	8,08
3200	20802	2566	8,11
3600	23248	2893	8,04
4000	25788	3217	8,02
5000	31906	4029	7,92
6000	38287	4819	7,95
7000	44922	5636	7,97
8000	51208	6395	8,01

The number of retries is the main factor influencing the computational complexity; hence the run time of the program. High numbers of retries result in a severe increase in the run time of the system. The duration increases from six minutes to some eight and a half hours with an increase in the number of detectors from 100 to 8000, which is definitely an unacceptable result for an IDS system. It should only be used as an offline evaluation mechanism.

6.2 Match Length

The value r is a threshold which determines the specificity of detectors, in that it controls how many strings can potentially be matched by a single detector. For example, if $r = \ell$ (57 in our case), the match is maximally specific, and the detector can match only a single string-itself. As shown in [56], the number of strings a detector matches increases exponentially as the value of r decreases. The probability of two random binary strings a, b matching under the contiguous bits rule is [14]:

$$P(Match_{contg}(a,b)) = p_M \approx 2^{-r}\left(\frac{\ell - r}{2} + 1\right) \tag{6.3}$$

The time complexity of the negative selection algorithm is proportional to the number of times a candidate detector must be regenerated before it becomes valid. The number of retries required to generate a valid detector is a geometric random variable and the expected number of trials, ρ until success is [16]:

$$E(\rho) = \frac{1}{(1 - p_M)^{|S|}} \tag{6.4}$$

Hence the expected number of retries to generate a single valid detector is exponential in the size of the self set. This assumes that there are no similarities between self strings. The more similar the self strings, the fewer the number of retries.

More specific detectors match fewer strings and thus can distinguish more precisely between observed self and everything else. Additionally, as the specificity of detectors increases, the number of detectors required to achieve the same level of coverage increases (see [16] for probabilistic estimates). So in order to minimize the number of detectors, the trick is to find the smallest r which still provides reasonable discrimination. This specific value is clearly dependent upon the particular data being monitored.

Since the specificity of the detectors determines the number of the strings they match, it affects the detection rates of the system seriously. This effect is seen as two different consequences. On one hand low specificity of the detectors results in a big increase in the match probability and causes the match of every new generated detector with a self string in the training data. This prevents the generation of mature detectors during the training phase; thus the system can not detect any anomalies. On the other hand high specificity of the detectors results in low match probabilities and

this result in the detection of lower number of anomalies with a certain number of detectors.

Table 6.6 Detection results and alarm counts for different *r* values

r	Detections	Alarms	True Alarms	False Alarms
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	30	886	759	127
10	34	1014	837	177
11	35	1062	838	224
12	34	1254	1039	215
13	27	924	732	192
14	24	946	789	157
15	21	813	692	121
16	12	744	637	107
17	7	220	204	16
18	8	56	37	19
19	8	329	287	42
20	2	38	4	34

Table 6.6 provides the test results of using different *r* values with 2400 detectors. As seen from the table the system failed to generate a single mature detector when the *r* value is below 9.

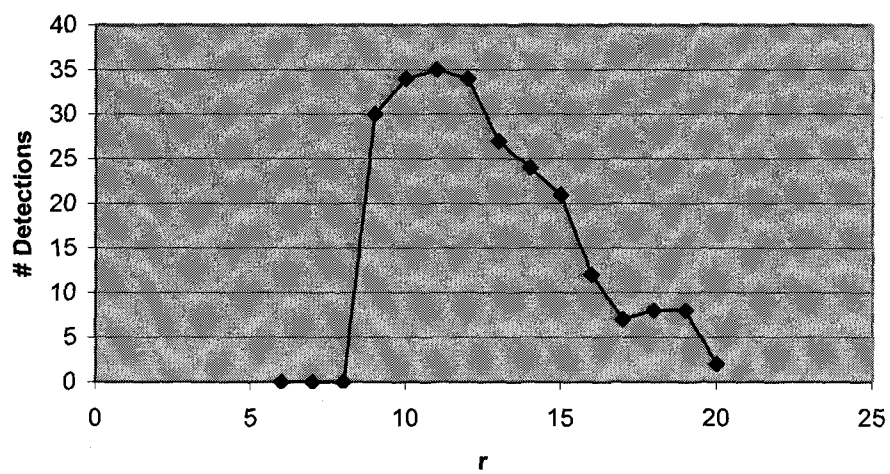

Figure 6.2 Number of detections against *r* value

The *r* values from 9 to 15 produced reasonable results, further increasing the r values results in a severe decrease in the number of detections. The effect of the match length is better shown in Figure 6.2. Applying a suitable value of *r* is crucial for the systems performance. These tests provided empiric results for that value and they are used for the other experiments.

7. CONCLUSION AND FUTURE WORK

This thesis investigated the implementation and application of negative selection algorithm described in [33,36] to network intrusion detection. The implementation is tested on the DARPA/Lincoln Laboratory 1999 IDS evaluation dataset, which is commonly used to test intrusion detection systems.

In this study, we have explored different properties of the negative selection algorithm and their effects to the detection rates. Different experimental results are provided above about the analysis of these properties or features.

The experimental results show that the negative selection algorithm is useful for network intrusion detection because it was able to detect a reasonable amount of attacks in the used dataset. However, it did not seem to be very efficiency because the tests took very long durations.

As far as we can see the main problem of the negative selection algorithm is the scaling problem. Unlike previous work using the negative selection algorithm for anomaly detection, a much larger "self" set was applied to the negative selection algorithm. This resulted in computational overhead and long processing time. Another problem was the representation of the data patterns. In the original work [33,36] "self" was represented by data-pathtriples defining the normal pair wise connections between computers. These include connections between two computers in the LAN and between one computer in the LAN and external computers. The data-pathtriple consisted of only the source IP address, the destination IP address and the port called for this connection. Moreover, the internal host addresses are only represented with their last bytes. This self definition was chosen based on the work by [57]. However, it is observed that this representation is very limited to encode different data. This limitation required the change described in section 5.1.1. This representation is also very limited to detect different types of network intrusions because not all the attacks are identifiable by looking only at the datapath triples representing this connection. There are many attacks which require looking at the other portions of the network packet to detect them. This is identified as a future work.

The modification described in section 5.1.1 enabled the application of the system to the DARPA dataset, but since the detectors lengthened the appropriate number of detectors to achieve a good detection rate increased. Therefore, it required a very long computation time. The experiment results clearly show this problem.

The r-contiguous rule is used to check the match between a given detector and antigen. However, the r-contiguous matching rule is argued to be too simple to determine the matching between complicated patterns. It has been stated the rules to represent intrusion signatures must describe correlation among significant network connection events [58,59]. Since the r-contiguous bit matching only measures the contiguous bits of genotypes of given two strings, it can not catch this kind of correlation from given self and nonself patterns. This is examined in [60].

The immune system is very complicated and not all of its properties are currently understood. It is also uncertain whether understanding these mechanisms and applying them directly to the field of intrusion detection is practical. Yet, there are many works applying some of the immune mechanism to this field.

The work by Hofmeyr and Forrest was the basis of this study. Likewise, it was also the inspiration point for many of the recent work already mentioned above. Some of the shortcomings of this work have been identified in this thesis or other works, but the results also confirm that this approach would be useful to achieve better anomaly detection systems for intrusion detection.

To sum up, this thesis provides the results of implementation and test of an artificial immune system for network intrusion detection. Some of the outcomes are provided above which can be summarized as: the negative selection requires high computation time to process large data, the representation scheme offered is not very appropriate because some attacks can not be identified in this scheme and finally the immune systems can still be considered as a promising approach for intrusion detection.

REFERENCES

[1] **McHugh J., Christie A., and Allen J.,** 2000. Defending Yourself: The Role of Intrusion Detection Systems, *IEEE Software*, **17(5)**, 42-51.

[2] **Bace, R.,** 1999. An Introduction to Intrusion Detection & Assessment, *ICSA Technical Report*, Pennsylvania.

[3] **De Castro, L. N. and Timmis, J. I.,** 2002. Artificial Immune Systems: A Novel Paradigm to Pattern Recognition, in *Artificial Neural Networks in Pattern Recognition*, pp. 67-84, Eds. J. M. Corchado, L. Alonso, and C. Fyfe, SOCO-2002, University of Paisley, UK.

[4] **Varela, F., Coutinho A., Dupire B., and Vaz N.,** 1988. Cognitive networks: Immune, neural, and otherwise, in *Theoretical Immunology, Part II*, pp. 359-375, Ed. A. Perelson, Addison-Wesley, New Jersey.

[5] **Kuby J.,** 1997. Immunology, 3rd ed., W. H. Freeman and Co., New York.

[6] **Hofmeyr S. A.,** 2000. An Interpretative Introduction to the Immune System, in *Design Principles for the Immune System and other Distributed Autonomous Systems*, pp. 3-28, Eds. I. Cohen and L. Segel, Oxford University Press, UK.

[7] **Inman, J. K.,** 1978. The antibody combining region: Speculations on the hypothesis of general multispecificity. in *Theoretical Immunology*, pp. 243-278, Eds. G. I. Bell, A. S. Perelson, and Jr. G. H. Pimbley, Marcel Dekker, New York.

[8] **Tonegawa, S.,** 1983. Somatic generation of antibody diversity, *Nature*, **302**, 575–581.

[9] **Osmond, D. G.,** 1993. The turn-over of B-cell populations, *Immunology Today*, **14(1)**, 34–37.

[10] **Smith, D., Forrest, S., and Perelson, A. S.,** 1998. Immunological memory is associative, *Artificial Immune Systems and their Applications*, pp. 105-114, Ed. Dasgupta, D., Springer-Verlag, Berlin.

[11] **Steinman, L.,** 1993. Autoimmune disease, *Scientific American*, **269(3)**, 75–83.

[12] **Dasgupta, D.,** 1998. An Overview of Artificial Immune Systems and Their Applications, in *Artificial Immune Systems and Their Applications*, pp 3-21, Ed. Dasgupta, D., Springer-Verlag, Berlin.

[13] **De Castro, L. N., and Von Zuben, F. J,** 1999. Artificial Immune Systems: Part I - Basic Theory and Applications, *Technical Report*, **TR DCA 01/99**, FEEC/UNICAMP, Brazil.

[14] **Percus, J. K., Percus, O. E., and Perelson, A. S.,** 1993. Predicting the Size of the T-cell Receptor and Antibody Combining Region From consideration of Efficient Sef-Nonself Discrimination, *Proceedings of the National Academy of Sciences*, **90**, 1691-1695.

[15] **Tizard, I. R.,** 1995. Immunology: Introduction, 4th Ed, Saunders College Publishing, Ft Worth, TX.

[16] **Forrest, S., Perelson, A. S., Allen, L., and Cherukuri, R.,** 1994. Self-nonself Discrimination in a Computer, *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, Los Alamos, CA, May 16-18, 202-212.

[17] **Forrest, S., Hofmeyr, S., and Somayaji, A.,** 1997. Computer Immunology, *Communications of the ACM*, **40(10)**, 88-96.

[18] **Forrest, S., Hofmeyr, S. A., Somayaji, A., and Longstaff, T. A.,** 1996. A Sense of Self for Unix Processes, *Proceedings of 1996 IEEE Symposium on Computer Security and Privacy*, Los Alamos, CA, May 6-8, 120-128.

[19] **Dasgupta, D.,** 1996. Using Immunological Principles in Anomaly Detection, *Proceeding of the Artificial Neural Networks in Engineering*, St. Louis, MO, November 10-13, 443-448.

[20] **Dasgupta, D.,** 1998. An Artificial Immune System as a Multi-Agent Decision Support System, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, San Diego, October 11-14, 3816-3820.

[21] **D'haeseleer, P,** 1997. A Distributed Approach to Anomaly Detection, *ACM Transactions on Information System Security.*

[22] **De Castro, L. N., and Von Zuben, F. J.,** 2000. The Clonal Selection Algorithm with Engineering Applications, *Proceeding of Artificial Immune System Workshop, Genetic and Evolutionary Computation Conference*, Las Vegas, NV, July 8-12, 36-37.

[23] **Forrest, S., Javornik, B., Smith, R. E., and Perelso, A. S.,** 1993. Using Genetic Algorithms to Explore Pattern Recognition in the Immune System, *Evolutionary Computation*, **1(3)**, 191-211.

[24] **Hightower, R., Forrest, S., and Perelson, A. S.,** 1996. The Baldwin Effect in the Immune System: Learning by Somatic Hypermutation, in *Adaptive Individuals in Evolving Populations*, pp. 159- 167, Eds. R.K. Belew and M. Mitchell, Addison-Wesley, Reading, MA.

[25] **Hightower, R., Forrest, S., and Perelson, A. S.,** 1995. The Evolution of Emergent Organization in Immune System Gene Libraries, *Proceeding of the Sixth International Conference on Genetic Algorithms*, Pittsburgh, PA, July 15-19, 344-350.

[26] **Oprea, M. and Forrest, S.,** 1998. Simulated Evolution of Antibody Libraries Under Pathogen Selection, *IEEE International Conference on Systems, Man and Cybernetics*, San Diego, CA, October 11-14.

[27] **Oprea, M. and Forrest, S.,** 1999. How the Immune System Generates Diversity: Pathogen Space Coverage with Random and Evolved Antibody Libraries, *Genetic and Evolutionary Computation Conference,* Orlando, FL, July 13-17.

[28] **Oprea, M.,** 1999. Antibody Repertoires and Pathogen Recognition: The Role of Germline Diversity and Somatic Hypermutation, *PhD Thesis*, Department of Computer Science, The University of New Mexico.

[29] **Jerne, N. K.,** 1974. Towards a Network Theory of the Immune System, *Annual Immunology (Inst. Pasteur)*, **125(C)**, 373-389.

[30] **Farmer, J. D., Packard, N. H., and Perelson, A. S.,** 1986. The Immune System, Adaptation and Machine Learning, Physica D: Nonlinear Phenomena, **22**, 187-204.

[31] **Timmis, J.,** 2001. Artificial Immune Systems: a Novel Data Analysis Technique Inspired by the Immune Network Theory, *PhD Thesis*, Department of Computer Science, University of Wales, Aberystwyth.

[32] **Yates, A. and Callard, R.,** 2001. Cell Death and the Maintenance of Immunological Memory, *Discrete and Continuous Dynamical Systems B*, **1(1)**, 43-60.

[33] **Hofmeyr, S. A.,** 1999. An Immunological Model of Distributed Detection and Its Application to Computer Security, *PhD Thesis*, Dept of Computer Science, University of New Mexico, NM.

[34] **Lamont, G. B., Marmelstein, R. E., and Van Veldhuizen, D. A.,** 1999. A Distributed Architecture for a Self-Adaptive Computer Virus Immune System, in *New Ideas in Optimization, Advanced Topics in Computer Science Series*, pp 167- 183, Eds. D. Corne, M. Dorigo and F. Glover, McGraw-Hill, London.

[35] **Harmer, P. K., and Lamont, G. B.,** 2000. An Agent Based Architecture for a Computer Virus Immune System, *Proceeding of Workshop on Artificial Immune System, Genetic and Evolutionary Computation Conference*, Las Vegas, USA, July 8-12, 45-46.

[36] **Hofmeyr, S. A., and Forrest, S.,** 2000. Architecture for an Artificial Immune System, *Evolutionary Computation*, **8(4)**, 1289-1296.

[37] **Okamoto, T., and Ishida, Y.,** 1999. A Distributed Approach to Computer Virus Detection and Neutralization by Autonomous and Heterogeneous Agents, *Proceeding of the International Symposium on Autonomous Decentralized Systems*, Tokyo, Japan, March 20-23, 328-331.

[38] **Kephart, J. O.,** 1994. A Biologically Inspired Immune System for Computers, Artificial Life IV, *Proceeding of the Fourth International Workshop on the*

Synthesis and Simulation of Living Systems, MIT, Massachusetts, July 6-8, 130-139.

[39] **Kephart, J. O., Sorkin, G. B., Arnold, W. C., Chess, D. M., Tesauro, G. J., and White, S. R.,** 1998. Biologically Inspired Defenses against Computer Viruses, in *Machine Learning and Data Mining: Method and Applications*, pp. 313-334, Eds. Michalski, R. S., Bratko, I., and Kubat, M., John Wiley & Sons, West Sussex, England.

[40] **Kephart, J. O., Sorkin, G. B., Swimmer, M., White, S. R.,** 1998. Blueprint for a Computer Immune System, in *Artificial Immune Systems and Their Applications*, pp 241-261, Ed. Dasgupta, D., Springer-Verlag, Berlin.

[41] **White, S. R., Swimmer, M., Pring, E. J., Arnold, W. C., Chess, D. M., Morar, J. F.,** 1999. Anatomy of a Commercial-Grade Immune System, *International Virus Bulletin Conference*, Vancouver, Canada, September 28-30.

[42] **Hofmeyr, S. A., Somayaji, A., and Forrest, S.,** 1998. Intrusion Detection System Using Sequences of System Calls, *Journal of Computer Security*, **6**, 151-180.

[43] **Williams, P. D., Anchor, K. P., Bebo, J. L., Gunsch, G. H., and Lamont, G. D.,** 2001, CDIS: Towards a Computer Immune System for Detecting Network Intrusions, *Proceedings of the Fourth International Symposium on Recent Advances in Intrusion Detection*, Davis, CA, October 10-12, 117-133.

[44] **Dasgupta, D.,** 1999. Immunity-Based Intrusion Detection Systems: A General Framework, in *Proceedings of the 22nd National Information Systems Security Conference*, Arlington, Virginia, October 18-21.

[45] **Dasgupta, D., and Brian, H.,** 2001. Mobile Security Agents for Network Traffic Analysis, *Proceedings of DARPA Information Survivability Conference and Exposition II*, Anaheim, CA, June 12-14, 332-340.

[46] **Dasgupta D. and Gonzalez F.A.,** 2001. An Intelligent Decision Support System for Intrusion Detection and Response, *Proceedings of International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security*, St. Petersburg, Russia, May 21-23, 1-14.

[47] **Dasgupta, D. and Gonzalez, F. A.,** 2001. A New Approach to Intrusion Detection, *University of Memphis C. S. Technical Report*, **CS-01-011**, Memphis, USA.

[48] **Kim, J. and Bentley, P. J.,** 2001. Evaluating Negative Selection in an Artificial Immune System for Network Intrusion Detection, *Genetic and Evolutionary Computation Conference*, San Francisco, July 7-11, 1330 – 1337.

[49] **Stillerman, M., Marceau, C. and Stillman, M.,** 1999. Intrusion Detection for Distributed Applications, *Communications of the ACM*, **42(7)**, 62-69.

[50] **D'haeseleer, P.,** 1996. An immunological approach to change detection: Theoretical results, *Proceedings of the 9th IEEE Computer Security Foundations Workshop*, Los Alamitos, CA, June 10-12, 18-26.

[51] **Lippman, R., Haines, J. W., Fried, D. J., Korba, J., and Das, K.,** 2000. Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation, in *Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection*, Toulouse, France, October 2-4, 162-182.

[52] **Lippman, R., Haines, J. W., Fried, D. J., Korba, J., and Das, K.,** 2000. The 1999 DARPA Off-Line Intrusion Detection Evaluation, *Computer Networks*, **34(4)**, 579-595.

[53] **Haines, J. W., Lippman, R., Fried, D. J., Zissman, M. A., Tran, E. and Boswell, S. B.,** 2001. 1999 DARPA Intrusion Detection Evaluation: Design and Procedures, *MIT Lincoln Laboratory Technical Report*, **TR-1062**, Massachusetts, USA.

[54] **Knuth, D. E.,** 1981. The Art of Computer Programming Vol. 2: Seminumerical Algorithms. Addison-Wesley, Reading, MA.

[55] **Press, W.H., Flannery, B.P., Teukolsky, S.A, Vetterling, W.T.,** 1992. Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, Cambridge, UK.

[56] **Esponda, F., Forrest, S.,** 2002. Detector coverage under the r-contiguous bits matching rule, *University of New Mexico Technical Report*, **TR-CS-2002-03**, UNM, USA.

[57] **Heberlein, L., Dias, G., Levitte, K., Mukherjee, B., Wood, J. and Wolber, D.,** 1990. A network security monitor, in *Proceedings of the IEEE Symposium on Security and Privacy,* Oakland, CA, May 7-9, 296-305.

[58] **Lee, W., Park, C., and Stolfo, S. J.,** 1999. Automated Intrusion Detection Using NFR: Methods and Experiences, in *Proceeding of 1st USENIX Workshop on Intrusion Detection and Network Monitoring*, Santa Clara, CA, April 9-12, 63-72.

[59] **Porras, P. A. and Valdes, A.,** 1998. Live Traffic Analysis of TCP/IP Gateways, in *Proceeding of ISOC Symposium of Network and Distributed System Security*, San Diego, CA, March 11-13.

[60] **Balthrop, J., Forrest, S., and Glickman, M. R.,** 2002. Revisiting LISYS: Parameters and Normal Behaviour, in *Proceedings of the Congress on Evolutionary Computation*, Honolulu, May 12-17, 1045 - 1050.

APPENDIX A. 1999 DARPA IDS EVALUATION ATTACK CATEGORIES

In the 1999 DARPA/Lincoln Laboratory IDS evaluation, all attack components had specific goals that included obtaining information about a computer/network system with intent to use that information as part of an exploit, achieving some level of privilege on a computer system not specifically granted by a system administrator, or violation of some explicitly stated security policy. Five major attack categories were chosen to group attack components with regard to the actions and goal of the attacker.

- **Denial of service (DoS)** attacks have the goal of limiting or denying service provided by to a user, a computer, or network. A common example is the SYN-Flood attack, where the attacker floods the victim host with more TCP connection requests than it can handle, causing the host to be unable to respond to valid requests for service.

- **Probe** attacks have the goal of gaining knowledge of the existence or configuration of a computer system or network. A common example is the IPsweep attack where the attacker sweeps one or more IP addresses in a given range to determine which addresses correspond to live hosts.

- **Remote-to-local (R2L)** attacks have the goal of gaining local access to a computer or network to which the attacker previously only had remote access. Examples are attempting to guess a password to illegally login to a computer system and the "imap" buffer-overflow, where a specifically crafted binary string sent during an "imap" mail connection yields access to the Linux system, for a particular version of imapd.

- **User-to-root (U2R)** attacks have the goal of gaining root or super-user access on a particular computer or system on which the attacker previously had user-level access. Common examples of U2R's are buffer-overflow attacks on various operating system programs, including eject, ffbconfig, and fdformat on Solaris.

- **Data** attacks have the goal of gaining access to some piece of information (file or directory) to which the attacker is not allowed access according the

stated security policy for the 1999 Evaluation. Many U2R and R2L attacks had an end-goal of accessing a "secret file" given the new, elevated privilege. These attack instances were classified as either U2R or R2L and also considered Data attacks. Other data attack instances occur when valid users breaking the security policy by, for example, accessing a secret file from a remote, un-encrypted login or emailing a secret file off the host.

In addition to being categorized by attack goal, attacks were also categorized by platform or operating system of the attack victim computer. These categories allow intrusion detection systems that detect attacks only against some of the victim platforms to specify those attacks that they are trying to detect. Reliance on system-level auditing information makes it likely that a host-based intrusion detection system will only be designed to detect attacks against that particular platform. Platforms that attacks were run against in the 1999 evaluation were:

- Solaris - pascal.eyrie.af.mil, 172.16.112.50, was the Solaris 2.5 victim host.
- Windows NT - hume.eyrie.af.mil, 172.16.112.100, was the Windows NT 4.0 victim.
- SunOS - zeno.eyrie.af.mil, 172.16.113.50, was the SunOS 4.1.4 victim.
- Linux - calvin.eyrie.af.mil, 172.16.112.20, RedHat 5.0, and marx.eyrie.af.mil, 172.16.114.50, RedHat 4.2 were both victims of attacks in the evaluation.
- Cisco - loud.eyrie.af.mil, 172.16.0.1 and 192.168.0.1, was the Cisco IOS v.11.3 rev. 4 router which also served as a victim host.

APPENDIX B. ATTACKS RUN IN THE 1999 DARPA IDS EVALUATION

Fifty-eight types of attacks were run in the 1999 evaluation. Table B.1 illustrates the categories in which these attacks are placed for the 1999 evaluation and shows how the attack instances run in the evaluation fall into those categories. Attacks in italics are those that are considered "new" in the 1999 Off-Line Evaluation, while those in regular font are "old".

Table B.1 Categorization of the attacks run in the DARPA 1999 IDS evaluation

	DoS (65)	**Probe (37)**	**R2L (56)**	**U2R (37)**	**Data (13)**
Solaris	Neptune pod Processtable *Selfping* Smurf Syslogd *Tcpreset* warezclient	Portsweep *Queso*	Dict Ftpwrite Guest Httptunnel Xlock Xsnoop	Eject Fdformat Ftbconfig Ps	*secret*
WinNT	*Arppoison* *Crashiis* *Dosnuke* *Tcpreset*	*Ntinfoscan* portsweep	*Dict* *Framespoofer* *Netbus* *Anypw* *ppmacro*	*Casesen* *Netcat* *Sechole* *Yaga*	*ntfsdos*
SunOS	*Arppoison* Land Mailbomb Neptune Pod processtable	Portsweep *queso*	Dict Xsnoop	Loadmodule	
Linux	Apache2 Arppoison Back Mailbomb Neptune Pod Processtable Smurf *Tcpreset* Teardrop udpstorm	*Ls_domain* Mscan *Queso* satan	Dict Imap Named *Ncftp* Phf Sendmail *Sshtrojan* Xlock Xsnoop	Perl Sqlattack Xterm	*secret*
Cisco	Neptune	Portsweep *queso*	Snmpget		
All O/S		*Illegalsniffer* Ipsweep portsweep			

CURRICULUM VITAE

Orhan Bıyıklıoğlu was born in Kastamonu, on the 19th of August, 1980. He received his BSc degree from Istanbul Technical University Computer Engineering Department in 2002. He has been working at the same department as a research and teaching assistant since then. His areas of interest include but not limited to: intrusion detection and prevention as well as other computer and network security topics such as vulnerability analysis, cyber forensics, and architecting secure networks.