CHAPTER 1

INTRODUCTION

The Security of a computer system from viruses, worms, attacks, etc. is a very difficult task. Lots of Security tools are commercially available for Computer Security. But the question arises in the mind of computer users, whether these tools are sufficient to protect the computer system. Various methodologies and technologies were proposed by various computer scientists for Computer Security. In this chapter these methodologies and technology are discussed and explained how Machine Learning and Soft Computing approaches can be used in Computer Security.

1.1 Overview

Nature, and particularly biology, has long presented a source of inspiration for new computational aptitude paradigms. Analogous to the way the nervous system functioning encouraged the development of Artificial Neural Networks (ANN). The Darwinian theory of evolution provoked the development of Evolutionary Algorithms (EA), the cram of the Biological Immune System (BIS) led to the emergence of Artificial Immune Systems (AIS). Since the early 1990s, these new promising Soft Computing techniques have already been functional in various domains such as data analysis, anomaly detection, fault diagnosis, pattern recognition and optimization.

Self awareness is one of the fundamental characteristics of life. Unusual program behavior often leads to program crashes, data sleaze and Security violations, in spite of these problems existing computer systems have no general purpose mechanism for detecting and responding to such anomalies. Natural Immune Systems protect living things from treacherous pathogens, as well as bacteria, viruses, parasites and toxins. Their function in the body is comparable to that of the Computer Security systems in computing. Although there are numerous differences between living organisms and computers, the similarities are
enthralling and could point the way to enhanced Computer Security. The anticipated elucidation of this is a computer that is intelligent to distinguish Self and Non-Self.

A novel computational intelligence technique, motivated by immunology, has emerged, called Artificial Immune Systems (AIS). AIS are fairly young promising techniques, which discover, develop and apply diverse biologically motivated immune mechanisms, intended at computational problem solving. Numerous concepts from the immune have been extracted and apply in solution to real world science and engineering problems. AIS is also used to provide Security for a computer system.

The aim of the research is to establish the sense in a computer system that could distinguish a Self process from the Non-Self process. Self process is the process that is not harmful to the system and do not use the CPU resource inappropriately. Self means part of the stable system and which does not harm the system. These processes are generally user or system processes. Non-Self processes are the processes that use CPU resources inappropriately and are harmful to the system. These processes have a few characteristic properties of making themselves a part of the computer system, hide their original identities and replicate themselves into a large number to extensively use system resources. Non-Self means which makes the system unstable and harm it.

The aim of this thesis is to describe a framework which identifies the Self and Non-Self process to provide the highest level of Security to a computer system. This thesis deals with identification of Self and Non-Self processes of a computer system. Various methods and techniques are being used for the definition and identification of the Self and isolation of Non-Self processes. These processes are being identified using the concept of Soft Computing and Machine Learning. Three approaches; Decision Tree, Artificial Neural Network and Genetic Algorithm are successfully implemented in the proposed framework.

A Decision Tree is constructed by using the concept of entropy and information gain. Iterative Dichotomiser 3 (ID3) algorithm is successfully implemented on the training data. ID3 uses the Information Gain for choice among the applicant attributes at each step while growing the tree. Primarily for the Decision Tree approach, five parameters of processes are used to recognize the Self and Non-Self processes.
Artificial Neural Network learning is vigorous in the training data and has been effectively applied to identify the Self and Non-Self processes. For the proposed framework, to fit an ANN, Gradient-Descent Algorithm and Backpropagation algorithm has been applied for creating Perceptron to identify the operating system processes as Self and Non-Self.

Genetic Algorithm is successfully implemented in the proposed framework to identify the Self and Non-Self operating system processes. The Detectors can employ (the Detectors will be according to process parameters and its pattern) by using Genetic Algorithms which can identify the processes as a Self and Non-Self. For the designing algorithm for generating a Detector, Genetic Algorithms concepts are used to ensure improvements in the functionality and should allow increasing its detection rates. The information that is conceded to future generations of Detectors should be improved so the new generated Detectors can effort more efficiently than the previous one. The Detectors will be generated randomly using the Genetic Algorithm for the first set of the population. The generated Detectors will be initial tartan that they do not equivalent with Self as they should match with the Non-Self only. If a match occurs, it will indicate an anomaly or an intrusion. The process will be stopped. For the next generation of Detectors the presented Detectors are used.

1.2 Security

Computer Security or Information Security rests on confidentiality, availability, and integrity. The interpretations of these three aspects vary, as do the contexts in which they occur. The interpretation of an aspect in a given environment is dictated by the requirements of the individuals, customs, and laws of the exacting organization. Confidentiality is the suppression of information or resources. The necessity for keeping information surreptitious arises from the use of computers in susceptible fields such as government and industry. Confidentiality also applies to the subsistence of data, which is occasionally more informative than the data itself.

Availability refers to the capability to use the information or resource desired. Availability is an vital feature of reliability and of system design because an unavailable system is at least as ghastly as no system at all. The characteristic of availability that is appropriate to Security is that someone may consciously arrange to contradict access to data or to a service by making it unavailable. System designs frequently imagine a statistical model to examine expected patterns of use, and mechanisms guarantee availability when that statistical model holds.
Integrity refers to the trustworthiness of data or resources, and it is usually phrased in conditions of preventing inappropriate or illicit change. Integrity includes data integrity (the content of the information) and origin integrity (the source of the data, frequently called validation). The source of the information may bear on its credibility and accuracy on the trust that people place on the information.

Computer Security or Information Security is the fortification of the items of assessment, called the possessions of a computer system. There are numerous types of assets, involving data, software, hardware, processes, people or a combination of these. To ascertain what to protect, first organize what has value and to whom. A computer device (together with hardware, added components, and accessories) is indeed an asset. Because most computer hardware is appealing useless exclusive of programs, the software is also an asset.

The software includes the operating system, device handlers and utilities; applications such as media players, word processing, or email handlers; and even programs that may have written by the user. Many software and hardware is off-the-shelf, implication that it is commercially accessible (not custom-made for user purpose) and that user can get a replacement without difficulty. The object that makes computer distinctive and significant to the computer user and is users content: projects, papers, email messages, photos, calendar information, ebooks (with users annotations), the programs created by the user, contact information and the most
similar to. The objective of Computer Security is shielding valuable assets. Figure 1.1 shows the component of information Security. To study different ways of protection, a framework is used, that describes how assets may be harmed and how to counter or mitigate that harm.

A vulnerability is a limitation in the computer system, i.e in procedures, system design, or implementation, that might be demoralized to grounds loss or harm. For example, a scrupulous system may be susceptible to unauthorized data exploitation because the system does not authenticate a user’s distinctiveness before allowing data access.

A threat to a computing system is a set of situation that has the prospective to reason of loss or harm. There are numerous threats to a computer system, together with human-initiated and computer-initiated. The user has all veteran the outcome of unintended hardware design flaws, software failures and human errors. But natural disasters are also treated as threats, as they can take a system failure when the computer room is flooded due to rain or the data center collapses when an earthquake comes, for example.

A human who exploits vulnerability perpetrates an attack on the computer system. An attack on a computer system can also be done by another computer system, as when one computer system sends an irresistible flood of messages to another, practically shutting down the second system’s capability to job. Unfortunately, it have been seen this type of attack commonly, as denial-of-service attack cascade servers with more messages than they can handle.

Computer viruses have been an emergent concern since the early 1980s. To fight this problem, a variety of antivirus programs have been formed; however, the encumber of detecting and destroying viruses is still brutal. Two trends are inhibiting the usefulness of existing antiviral techniques: the growing birth rate of new viruses and the increase in interconnectivity and interoperability between computers. Existing techniques are imprudent, labor intensive for virus researchers, have a time-consuming response from time of innovation until the cure is prescribed, and oblige user involvement to bring up to date the virus signature database. Enhance contemporary antiviral techniques can fight these problems. Based on numerous properties of the human immune system, computer scientists anticipate novel techniques to fight computer viruses will come into sight. They think that the human immune system has a number of useful characteristics for detecting computer viruses.
These properties provide a robust, lithe and scalable system durable to attack and a spanking new perspective on computer viruses and other Security's troubles.

Currently, nearly all virus protection tools for computers is implemented using signature recognition consequent after analyzing the known viruses. Although this method has been somewhat triumphant to date, computer era is quickly forthcoming a time when such methods will be insufficient. As viruses are continually mutating and tweaked to shirk detection, the signature list becomes bigger and bigger perhaps approaching seven figures. Another impenetrability is that viruses are only detected after they have been firstly analyzed, revealed and signatures have been distributed. This procedure can be time-consuming and wastes valuable time while a fast replicating virus rapidly renders a networked computing system useless.

1.2.1 Threats

A threat is a prospective desecration of Security. The desecration need not really happen for there to be a threat. The fact that the desecration might happen, means that those procedures that could cause it to occur must be protected against (or primed for). Those trials are called attacks. Those, who execute such events or grounds them to be executed, are called attackers. The three Security services: confidentiality, integrity, and availability - oppose threats to the Security of a system. Shirey (1994) divides threats into four extensive module revelation, or unauthorized access to information; trickery, or approval of false data; distraction, or interruption or anticipation of correct operation; and usurpation, or unauthorized manage of some part of a system. These four broad classes include many common threats. Because the threats are ubiquitous, a preliminary discussion of each one will present issues that persist throughout the lessons of Computer Security. Figure 1.2 shows the different kinds of threats.

Denial of service, a long-standing bashfulness of service, is a form of usurpation, although it is often used with other methods to swindle. The attacker tries to stop a server from providing a service. The denial may take consign at the source (by preventing the server from obtaining the resources required to execute its function), at the destination (by blocking the communications from the server), or along the intermediary path (by throwing away messages from either the client or the server, or both sides). Denial of service poses the similar threat as an unlimited delay. Availability mechanisms, contradict this threat.
Denial of service or delay may result from direct attacks or from non-Security associated troubles. From users spot of view, the reason and consequence are important; the intention underlying them is not. If a delay or denial of service compromises system Security, or are part of a sequence of events leading to the conciliation of a system, then it is viewed as a attempt to violate system Security. But the attempt may not be premeditated; undeniably, it may be the merchandise of environmental distinctiveness rather than specific actions of an attacker.

1.2.2 Attackers

Who are the attackers? As it has been seen, their motivations range from a possibility to a definite target. Putting aside attacks from natural and gentle causes, it can be explore who are the attackers and what motivates them. Most studies of attackers essentially analyze computer criminals, that is, people who have essentially been convicted of a crime, chiefly because that group is easy to classify and study. The ones who got away or who carry off an attack without being detected may have characteristics specially from those of the criminals who have been trapped. Worse, by studying only the criminals that are trapped, it may not be learn how to grab attackers who know how to mistreatment the system without being seized. Some are school or university students. Others are middle-aged industry executives. Some are
mentally unbalanced, overtly hostile, or tremendously dedicated to a cause, and they attack computers as an emblem. Others are common people tempted by individual revenge, challenge, advancement, profit, or job security-like perpetrators of some offense, by means of a computer or not. Computer researchers have tried to find the psychological personality that discriminates attackers. Originally, computer attackers were individuals, performing with motives of challenge, fun, or revenge. Early attackers such as Robert Jr., the graduate student of Cornell University, who brought down the Internet in 1988, and Kevin Mitnick, the man who broke into and stole data from dozens of computers including the San Diego Supercomputer Center acted alone.

More topical attacks have implicated groups of people. An attack adjacent to the government of the country of Estonia is thought to have been an ungraceful flare-up from a loose confederacy of attackers from around the world. Attackers’ goals include extortion, fraud, money laundering, and drug trafficking, areas in which planned crime has a well-established incidence. Evidence is growing that planned crime groups are engaging in computer crime. In fact, conventional criminals are recruiting hackers to join the money-spinning world of cyber crime. Planned crime may use computer crime (such as pilfering credit card numbers or bank account information) to finance other aspects of crime.

1.2.3 Security Policy and Mechanism

A Security policy is a declaration of what is, and what is not, permitted. A Security mechanism is a technique, tool, or a method for enforcing a Security policy. Mechanisms can be nontechnical, such as requiring evidence of identity before varying a password; in fact, policies often necessitate some ceremonial mechanisms that technology cannot implement. Policies may be accessible mathematically, as a list of allowable (secure) and prohibited (nonsecure) states. For users’ purposes, it will presume that any given policy provides a self-evident explanation of secure states and insecure states. In observing, policies are hardly ever so specific; they normally explain in English what users and staff are permitted to do. The indistinctness innate in such a explanation leads to states that are not confidential as “allowed” or “disallowed”.

In the real world, a Security policy describes how people may access documents or other information. In order for the policy to be reflected in a computer environment, it must rewrite it using terms such as subjects and objects that are meaningful to the computer. Strictly
speaking, the computer obeys Security properties, while people obey a Security policy. However, loosely talk about the computer’s Security properties as if they were a policy of the computer system. In cases where the distinction between Security policy and Security properties is especially important to use the more precise terminology.

The computer’s version of the policy consists of a precise set of rules for determining authorization as a basis for making access control decisions. Authorization depends on the Security attributes of users and information, unique IDs, and perhaps other information about the current state of the system. While all systems have Security properties, the properties are not always explicit, and the policy on which they are based may be difficult to deduce. Often the policy is a hodgepodge of ethic rules that have evolved over the years and are inconsistently enforced. Lack of a clear policy and not programming errors is a major reason why the Security controls of many systems are flawed.

1.3 Evolutionary Systems

All biological systems result from an evolutionary process. The sophistication, robustness, and adaptability of biological systems represent a powerful motivation for replicating the mechanisms of natural evolution in the attempt to generate software and hardware systems with characteristics comparable to those of biological systems. More than 40 years ago, computer scientists and engineers began developing algorithms inspired by natural evolution [Rechenberg (1965), Fogel et al. (1966) and Holland (1975)] to generate solutions to problems that were too difficult to tackle with other analytical methods. Evolutionary computation rapidly became a major field of Machine Learning and system optimization.

Biology is making continuous progress in the description of the components that make up living organisms and of the ways in which those components work together. However, the ultimate explanation is to be found in the theory of natural evolution. As Dobzhansky (1973) put it, “nothing in biology makes sense except in the light of evolution.” A bewildering number of books and articles have been written on the theory of natural evolution, but its foundations are rather simple and elegant.

The theory of natural evolution rests on four pillars: population, diversity, heredity, and selection. The premise of evolution is the existence of a population, which here it will loosely define as a pool of two or more individuals. In other words, it cannot be about the evolution
of a single organism. Diversity means that the individuals of the population vary from one another to some extent. Individual diversity, both within and between species, has been observed and described for thousands of years. Heredity indicates that individual characters can be transmitted to offspring through reproduction.

The notion that individual characters are hereditary was suggested in the eighteenth century by Maupertuis (1753). Selection indicates that only part of the population is capable of reproducing and transmitting its characters to future generations. Natural selection, put forward by Darwin (1859) and Wallace (1870) in the nineteenth century, is based on the premise that individuals tend to make several offspring and that not all of them may reproduce.

The selection of individuals that can reproduce is not completely random, but regulated by environmental constraints. For example, if an environment contains too many individuals for the available food, those individuals that are better or faster at gathering food will have a higher chance of survival and reproduction.

1.3.1 The Genotype and the phenotype

The genetic material of an individual is known as the genotype, whereas its manifestation as an organism is known as the phenotype. Natural selection operates solely on the phenotype, but the genotype is the ultimate vehicle of inheritance. The extent to which are determined by genotype or phenotype and the relationship between these two aspects of individuality is a complex and much debated issue [Gould (1977) and West-Eberhard (2003)].

1.3.2 Artificial Evolution

Artificial advancement includes a broad set of algorithms that take motivation from the ethics of natural evolution and molecular genetics in order to automatically find solutions to hard optimization problems, design electronic circuits, ascertain novel computer programs, get better object shapes, and investigate several other areas that are frequently addressed by human design. Most artificial advancement is based on the very same four pillars of natural evolution: (1) maintenance of a population; (2) conception of diversity; (3) a selection mechanism; and (4) a process of genetic inheritance.
In artificial advancement, the phenotype of an individual is the elucidation to a problem and undergoes a selection process. The genotype in its place is a generic illustration of that solution and is transmitted through generations and manipulated by genetic operators. The mapping between the genetic representation (genotype) and the problem description (phenotype) can take a variety of degrees of complication ranging from a direct, one-to-one association all the way to complicated models of gene idiom.

The problem-solving quality of artificial advancement is built into the selection process, which consists of two steps: (1) an evaluation of the phenotype that provides a quantitative gain, also known as the fitness value; and (2) a reproduction operator that makes a bulky number of copies of genotypes consequent to phenotypes with greater fitness values. Although this functional and goal-oriented interweave of evolution has been fruitfully applied to living organisms by breeders of plants and animals for hundreds of years, it is not the way in which natural evolution operates.

Evolutionary algorithms are frequently used on inflexible problems where other optimization methods don't succeed or are trapped in suboptimal solutions. Those problems classically include belongings that have numerous free parameters with multifarious and nonlinear exchanges, are characterized by non-continuous functions, have misplaced or corrupted data, or demonstrate several local optima. Evolutionary algorithms are appropriate for a large number of domains as long as an articulate genetic representation can be formulated.

Evolutionary algorithms can also be united with other balancing search methods to increase the superiority of the solutions. For example, a local gradient ascent technique could be functional to the phenotypes prior to fitness estimation so that the selection process could replicate individuals that are situated in the best areas of the search space. Evolutionary algorithms also allow interaction and association with human designers, for example by letting humans dominate the fitness function and manually select assured individuals for reproduction or interleave in the evolving population genotypes of individuals with preferred features. There are several types of evolutionary algorithms, which are often labeled in a different way for historical reasons. These algorithms put prominence on diverse mechanism, such as the type of mutation operator, or are customized for unambiguous types of problem, such as the advancement of computer programs. Instead of delving into the niceties of each type of evolutionary technique, an outline is provided with the main steps essential to
accumulate a “custom-made” evolutionary algorithm. These steps are: (1) choose a genetic representation; (2) build a population; (3) design a fitness function; (4) choose a selection operator; (5) choose a recombination operator; (6) choose a mutation operator; (7) devise a data analysis procedure.

1.3.3 Types of Evolutionary Algorithms

There are several types of evolutionary algorithms that differ mainly in the choice of genetic representation and operators [Eiben and Smith (2003)]. The optimal choice of algorithm, or

![Diagram of a simple evolutionary algorithm]

Figure 1.3: A simple evolutionary algorithm. The bit strings represent the genotypes of the individuals. The generational cycle illustrated in the four boxes is continued until a satisfactory genotype is found.

the assembly of a custom-made evolutionary algorithm, depends on the properties of the problem to be solved; in other words, there is no single algorithm that performs better on a majority of problems [Michalewicz (1996)].

Genetic Algorithms [Holland (1975)] operate on binary representations of the individuals and emphasizes the role of building blocks and crossover. Genetic programming [Koza (1992)] operates on tree-based representations of computer programs and circuits. Evolutionary programming [Fogel et al. (1966)] operates directly on the parameters that define the
phenotype by applying perturbations drawn from a zero-mean Gaussian distribution (small perturbations are more likely than large perturbations).

Evolutionary programming often relies on tournament-based selection with gradual population replacement and does not use crossover. Evolutionary strategies [Rechenberg (1973)] are similar to evolutionary programming, but the variance of the distribution used for mutation of the individual is genetically encoded and evolved along with the parameters that define the phenotype. Figure 1.3 shows a simple evolutionary algorithm. In this Figure 1.3 the bit strings represent the genotypes of the individuals and the generational cycle illustrated in the four boxes is continued until a satisfactory genotype is found.

1.4 Natural Immune System

Immunology is the study of the body’s resistance to invasion by other organisms. The immune system uses several layers of defense to protect the body against invaders, known as pathogens. Initial barriers to infection are the skin and physiological barriers such as pH and temperature. If the pathogens are able to get past these barriers, they must be dealt with by another layer of the immune system.

Computational Neuroscience, which finds its roots in the detailed mathematical model of neuronal membrane dynamics, first described by Hodgkin and Huxley (1952), attempts to understand the functioning of living brains. The main questions addressed by computational neuroscience include the type of communication used by neurons, the effects of chemicals on neuronal behavior, the dynamics of neuronal assemblies, and the theoretical capacity of neuronal computation, to mention a few. Neural engineering, which can be traced back to the logic-level description of a neuron given by McCulloch and Pitts (1943), instead aims at reproducing the functionalities of brains in order to engineer intelligent machines. Issues addressed by neural engineering include robust control of robotic systems, learning algorithms and high-level architectures that could reproduce cognitive abilities, and implementation of neural models in hardware.

1.5 Artificial Immune System

To survive and reproduce, living beings need suitable materials and energy and must find these resources in their environment. Since all known naturally evolved living beings are composed of the same basic building blocks, they are potentially a rich source of high-quality
matter and energy for each other. For this reason, living organisms must protect themselves from the attempt of other organisms to exploit their resources. For example, viruses, bacteria, fungi, protozoans, and some kinds of parasitic worms in the initial stage of their life cycle are much smaller than the typical vertebrate.

Many human-built systems face the same kind of problems of biological organisms targeted by pathogens. For example, computer systems represent computational resources and contain data that attract non-authorized users in the form of computer viruses and network intrusion attempts [Mukherjee et al. (1994) and Nachenberg (1997)]. Typically the non-authorized operations take place at a low level in the hierarchy of software levels on the computer system so that their effect is not immediately apparent at the scale of the computer user or network administrator interface.

The countermeasure consisting in the isolation of the computing system is seldom an option in times of widespread networking. The addition of built-in protections to the operating system does not always solve the problem, because the frequency of update that is reasonable for the operating system is quite low when compared with the speed with which the attack modality can change. Currently, the most common solution is the use of frequently updated antivirus and intrusion detection programs. However, the implementation and update of these protection programs require a substantial effort, and the effectiveness of the protection can be compromised if a communication failure or an oversight results in the omission of an update. A better solution would be a protection system capable of autonomously detecting and opposing the attempts at intrusion and exploitation.

Human-built systems must also be protected against malfunctioning and failures of their subsystems. As mentioned above, the strategies used to automatically fight exploitation attempts can also be used for the detection and cure of faults, that is, to obtain systems with built-in fault tolerance. Artificial immune systems (AISs) are the result of an effort to implement protection against external attacks and internal faults explicitly inspired by the workings of biological immune systems. More generally, an AIS is any artificial system that implements some of the processes that are found in biological immune systems. The applications of an AIS can thus go beyond system protection and fault tolerance to encompass other functions such as pattern recognition, noise reduction, function optimization, and biological modeling. To pave the way to the understanding of AIS, in the
structure and operation of biological immune systems must be described. Then, proceed to show how the concepts inspired by biological immune systems can be applied to the definition of AIS, and describe some examples of their application to computer and network protection and to fault detection in electronic systems.

![Diagram of the negative selection algorithm]

**Figure 1.4:** The steps of the negative selection algorithm

### 1.5.1 Negative Selection Algorithm

A negative selection algorithm was proposed [Forrest et al. (1994), Dasgupta (2007)] for the generation of the Detectors of an AIS. The negative selection algorithm assumes that there is a collection $P$ of fixed-length strings of symbols which must be protected from unauthorized change. For example, this collection could be an assemble of data and program files in the memory of a computer, or the control program of an electronic device, or the patterns of operation of a machine, or the patterns of connectivity and traffic of a networked computer. In the absence of unauthorized changes $P$ corresponds to a collection $S$ which is called the Self. The goal of the algorithm is to generate a set of Detectors that can signal the appearance in $P$ of any string that does not belong to $S$, that is, the appearance in $P$ of any Non-Self
string. Non-Self strings could be generated, for example, by the presence in the system of a virus or a network intrusion. To attain this goal the algorithm prescribes the following steps:

- Assign a comparison or matching function \( m(\cdot, \cdot) \) for pairs of strings, a detection threshold \( \theta_D \), a mechanism of generation of candidate receptor strings, and the maximum adequate probability \( P_f \) of detection failure.

- Estimate the number \( N_D \) of strings required to obtain the performance specified by \( P_f \) using the recognition regions specified by \( m(\cdot, \cdot) \) and \( \theta_D \), and the mechanism of generation of candidate receptor strings specified at step 1.

- Censoring phase (Figure 1.4): Generate a candidate receptor string \( r_c \). If \( r_c \) matches any Self-string, that is, if \( m(s, r_c) \geq \theta_D \) for any string \( s \) belonging to \( S \), discard the string; otherwise include \( r_c \) in the initially empty set of receptors \( R \). Repeat this step until the size \( |R| \) of \( R \) corresponds to \( N_D \).

- Monitoring phase (Figure 1.4): Choose (either deterministically or randomly) a string \( s \) in \( P \) and a string \( r \) in \( R \) and evaluate \( m(s, r) \). If \( m(s, r) \geq \theta_D \), signal the detection in \( P \) of a string not belonging to the legitimate Self \( S \); otherwise repeat this step with another pair of strings \( (s, r) \).

1.5.2 Clonal Selection Algorithm

The Detectors of the vertebrate immune system that survive the process of negative selection are distributed in the body and interact with the antigens. For some of these Detectors there exists a mechanism that improves their recognition performance. This mechanism of affinity maturation is based on an iterative process of production of clones, variation, and selection which resembles an evolutionary process. Eventually, the best performing Detectors, resulting from this clonal selection process are preserved as memory cells. A clonal selection algorithm based on the characteristics of this biological process has been proposed [Castro and Zuben (2002)] for pattern recognition and function optimization. In the case of function optimization problems the goal is to produce a population \( R \) of receptors that constitute a collection of candidate solutions to the problem. To achieve this end the clonal selection algorithm prescribes the following steps:
Assign the size of the population of receptors, the selection strategy, and the number of random receptors generated at each iteration. Assign a function that transforms the value of the optimizing function into a value of affinity. Assign a function that links the affinity to the rate of mutation to be applied. Assign the function that links the affinity to the number of clones to be produced.

Initialize a random population R of receptors.

Evaluate the affinity of the receptors in the population R.

Select the receptors with highest affinity obtaining a collection R_H.

Clone the elements of R_H, that is, for each element in R_H create a number of copies prescribed by the function assigned in step 1.

Mutate the clones with a mutation rate given by the function assigned in step 1, obtaining a collection R_M.

Generate randomly a collection R_R of new receptors.

Select the best receptors among R_H, R_M, R_R to form the new population R of receptors and return to step 3, unless some termination criterion is satisfied.

1.5.3 Self and Non-Self

Traditionally, the Self is defined as the internal cells and a molecule of the body, whereas Non-Self is measured as all elements that do not belong to “Self”, i.e. all foreign substances including viruses and bacteria. In the biological system, there exists, foreign organism Detectors known antigen. Thus, when bound to other cells, initiate an immune response to acknowledge lymphocytes to ruin the bounded foreign cells.

The human immune system is unique, meaning individual immunity is derived and adapted differently; this is a desirable and appropriate property of computer system, as well. The human system is extremely flexible and does not necessitate the absolute detection of every invader; instead, partial detection allows for quicker recognition of various invaders. In a computer system, this is similar to the use of byte pattern signatures as partial Detectors for locating an infected file on a computer system.
For human beings, detection of an unidentified object is not easy and sometimes causes
cognitive errors, but if the Self is known, a discrimination of the unknown object from the
Self becomes easier. Due to the known complexity of the Non-Self, the attribute set of the
Non-Self is unlimited in theory and is not an sufficient amount of the criteria for detecting
unknown Non-Self.

However, many Non-Selfs are detecting techniques such as virus detecting, abnormality
detecting and fault detecting are based on matching the features of the Non-Self, and the
probability of detecting the Non-Self is quite limited. In fact, any unknown Non-Self such as
viruses and faults may cause mortal lost in the application system, so that many problems
such as anti-virus Security, fault diagnosis and robust control, push the Non-Self detecting
techniques to improve judgment & methods.

1.6 Genetic Algorithms

Genetic Algorithms (Gas) will be used for Detector generation. GAs is an adaptive heuristic
search algorithm based on the evolutionary thoughts of natural selection and genetics.

- GAs is a ingredient of evolutionary computing, a quickly growing area of artificial
  intelligence.
- GAs is motivated by Darwin’s theory about evolution-“survival of the fittest”.
- GAs represents an intelligent development of random search used to optimize the
  dilemma.
- GAs, though randomized, utilize historical information to express the search into the
  province of better performance within the search space.
- In nature, effort among individual for inadequate resources results in the fittest
  individual dominating over the weaker one.

At the establishment of a run of a Genetic Algorithm a huge population of random
chromosome is produced. Each one, when decoded will embody a different solution to the
given problem. Let's say there are N chromosomes in the preliminary population. Then the
following steps are repetitive until a solution is found.

- Test each chromosome to see how excellent it is at solving the problem at hand and
  assign a fitness value consequently. The fitness value is a appraise of how excellent
  that chromosome is at solving the problem at hand.
Select two chromosomes of the present population. The probability of being selected is relative to the chromosome fitness value.

Depending on the crossover operator, crossover the bits from each selected chromosome.

Step in the course of the selected chromosomes bits and flip depending on the mutation operator.

Repeat step $2^{nd}, 3^{rd}, 4^{th}$ in anticipation of a new population of $N$ members has been created.

Genetic algorithm will be used as an alternative of unadventurous AI because-

- It is superior than unadventurous AI; it is more robust.
- Unlike older AI systems, the Genetic Algorithms do not shatter easily, even if the inputs changed to some extent, or in the presence of reasonable noise.
- While performing a search in multi-modal state-space, large state-space, or n-dimensional surface, a Genetic Algorithm present considerable benefits over many other usual search optimization techniques like –heuristic, linear programming, depth-first, breadth-first etc.
- "Genetic Algorithms are superior at taking large, potentially colossal search spaces and navigating them, looking for best possible combinations of belongings, the final outcome one might not or else find in a life span." – Salvatore Mangano(1995).

1.7 Artificial Neural Networks

Artificial Neural Networks (ANN) are computational models implemented in software or custom-made hardware devices that attempt to capture the behavior and adaptive features of biological nervous systems.

An ANN is composed of several interconnected units, or neurons. Some of these units receive information directly from the environment (input units), some have a direct effect on the environment (output units), and others communicate only with units within the network (internal, or hidden, units).

Each unit implements a simple operation that consists in becoming active if the total incoming signal is larger than its threshold. An active unit emits a signal that reaches all units to which it is connected. The connection, or synaptic point, operates like a filter that
multiplies the signal by a signed weight, also known as synaptic strength. Figure 1.5 shows the schematic representation of a biological and artificial neuron.

Figure 1.5: Schematic representation of a biological and artificial neuron.

Whereas biological neurons are either inhibitory or excitatory and have the same effect on all neurons which they send signals to, artificial neurons can emit both negative and positive signals and thus the same neuron can establish both negative and positive synaptic connections with other neurons. There are two reasons for this difference. The first is that artificial neurons are mathematical objects that are not constrained by the physiological properties of biological neurons in order to achieve the same functionality. The second is that an artificial neuron often models the average response of a population of biological neurons, which may include both excitatory and inhibitory neurons.

The response of an ANN to an input from the environment depends on its architecture and pattern of connection strengths. The knowledge of the network is distributed across its connections. The behavior of the network is given by the pattern of activations of the neurons, which in some models can self-sustain and change over time even in the absence of input from the environment.

**ANN Learning**

The field of Machine Learning can be divided into three broad categories: unsupervised, supervised, and reinforcement learning. With supervised learning, the ultimate goal is usually classification of a previously unseen data item based on the characteristics of the problem learned during the training phase of the algorithm. During training, both the feature vector and the vector’s classification are given.
The algorithm uses both pieces of information to learn a view of the world that will allow correct classification of future instances. At times the known class of the training instance will be used to assess the quality of the algorithm’s response; whereas, at other times this class information will be used for restructuring the memory representation of the system. In all cases of supervised learning, it is the combination of the feature vector and the class that help dictate the adaptation of the memory system.

Unsupervised Learning

In unsupervised learning the neural network learns some properties of the input pattern distribution without any feedback from the environment or from the user. Learning typically consists in the extraction of information, such as the detection of common or distinctive features that allow the network to classify the input patterns. From a mathematical perspective, unsupervised learning performs statistical operations such as computation of correlation indices, estimation of parameters of the probability density function of the input patterns, and principal component analysis, to mention a few. In order to carry out those operations, the input pattern distribution must be redundant so as to allow the detection of structure [Barlow (1989)].

Supervised learning

Supervised learning is characterized by the presence of a teacher that provides the response required from the network for each training pattern. Within this framework, originally proposed by Rosenblatt (1962), the synaptic weights are modified so as to reduce the error between the desired response and the response given by the network.

Reinforcement Learning

Reinforcement learning focuses on goal-driven learning in which the actions that the learner takes has an impact on the external environment. However, this impact is possibly dimly perceived. Unlike supervised learning which presents training examples and solutions as connected pairs, in reinforcement learning systems the learner is given a reward for certain actions. The goal of the learner is to maximize its reward. One of the fundamental issues in this reinforcement learning environment is determining which sequence of actions resulted in the greatest reward. That is, since the reward may be provided only after several actions made by the learner, the learner is taxed with the task of appropriately assigning credit to the given
states it passed through when attaining the given reward. Despite their computational power, networks trained with the delta rule or with backpropagation can be used only in those situations where one knows the correct response for all input patterns in the training (and validation) set. This is not always the case for agents that operate in partially unknown environments where the feedback (if any) available from the environment is usually rare and generic.

Unsupervised learning does not require the specification of the correct response, but pays this flexibility with its inability to discriminate the statistical regularities that are potentially useful to the agent and are thus worth learning from those that are not worth learning. Agents that must operate in a partially unpredictable environment require therefore another kind of learning that is neither purely supervised nor purely unsupervised. The solution devised by evolution consists in equipping biological agents with a kind of learning that is linked to the consequences of the agent’s behavior.

The exact mechanism that implements this kind of learning in biological neural systems is the subject of much research and is not yet completely understood. The existing evidence points to the combined action of evolved value systems [Pfeifer and Scheier (1999)] and neuromodulatory effects [Fellous and Linster (1998), and Bailey et al. (2000)]

The value system has the task of discriminating the behaviors, according to their reinforcing, punishing, or negligible consequences. This leads to the production of neuromodulatory signals that can activate or inhibit synaptic learning mechanisms. Algorithms inspired by this kind of approach have been developed by the Machine-Learning community. For example, reinforcement learning is a class of learning algorithms that attempt to estimate, explicitly or implicitly, the value of the states experienced by the agents in order to favor the choice of those actions that maximize the amount of positive reinforcement received by the agent over time [Sutton and Barto (1998)].

1.8 Decision Tree Learning

A Decision Tree is a hierarchical data structure implementing the divide-and-conquer strategy. It is an efficient nonparametric method, which can be used for both classification and regression. Learning algorithms will be discussed that build the tree from a given labeled
training sample, as well as how the tree can be converted to a set of simple rules that are easy to understand. Another possibility is to learn a rule base directly.

A Decision Tree is a hierarchical model for supervised learning whereby the local region is identified in a sequence of recursive splits in a smaller amount of steps. A Decision Tree is tranquil of domestic decision nodes and terminal leaves. Decision Tree Learning approximates a target function which is represented as a Decision Tree. Each internal node in the Decision Tree is an attribute of the problem space, and each branch from a given internal node corresponds to a value the attribute can assume. The leaf nodes of the tree represent the values of the target attribute (the attribute that is being used for classification). As may be obvious from this description, Decision Tree Learning typically applies to problems whose feature vectors consist of discrete valued attributes.

Figure 1.6 shows a Decision Tree for the concept PlayTennis[Tom (1997)]. An example that classified by sorting it through the tree to the apposite leaf node, then inveterate the classification associated with this leaf (in this case, Yes or No). This tree classifies Saturday mornings according to whether or not they are suitable for playing tennis.

![Decision Tree Diagram]

**Figure 1.6 :** A Decision Tree for the concept *PlayTennis*.

Decision Tree Learning, then, is the process of learning the correct structure of this tree, i.e., which attributes get tested, at which level or along which path of the tree. This is typically achieved through finding the tree that correctly classifies the largest number of examples from a training data set. Once this tree is discovered, pruning, through various means, sometimes occurs to prevent the tree from overfitting the training data and, thus, not being
able to generalize to previously unseen data items. Finally, the learned tree is used to classify
instances of the problem space according to the target attribute.

1.9 Machine Learning

To solve a problem on a computer, an algorithm is required. An algorithm is a step of
instructions that should be conceded out to transform the input to output. Machine Learning
is training computers to optimize a performance measure using example data sets or
precedent experience. A model is definitely up for some parameters, and learning is the
implementation of a computer program to optimize the parameters of the model using the
training data set or precedent experience. The model may be predicted to make predictions of
the future, or troublesome to expand knowledge from data, or both. Machine Learning uses
the assumption of statistics in construction mathematical models, because the foundational
task is making inferences from a sample. The responsibility of computer science is dual:
First, in training, an efficient algorithm is required to solve the optimization problem, as well
as to accumulate and process the enormous amount of data generally have. Second, once a
model is learned, its illustration and algorithmic solution for presumption needs to be
proficient as well. In certain applications, the effectiveness of the learning or presumption
algorithm, namely, its space and time complexity, may be as significant as its predictive
accuracy.

The ability to improve with experience, to get better over time, to remember past decisions
and outcomes in order to make better choices in the future, similar situations, that is, the
ability to learn can be seen as a fundamental characteristic of human intelligence. This being
said, it is the slight disclosure that so much research in the field of Artificial Intelligence (AI)
could be labeled, in one way or another, as research in Machine Learning. According to Tom
Mitchell (1997), “The field of machine learning is concerned with the question of how to
construct computer programs that automatically improve with experience”. This description
will acknowledge as well as any other for now concerning this field. However, while we may
have a working definition of what Machine Learning is, we do not, necessarily, have a reason
as to why we would want such a thing. Why do we care to develop programs that can
“automatically improve with experience?”

Machine Learning furthermore helps to discover solutions to numerous problems in robotics,
vision, speech recognition, and many more. Application of Machine Learning methods to
huge databases is called data mining. Its application areas are profuse: In addition to trade, in finance, banks explore their past data to construct models to use in the stock market, credit applications, and the fraud detection. In the manufacturing sector, learning methods are used for control, optimization, and troubleshooting.

In medicine and healthcare, learning programs are used for medical diagnosis. In telecommunications, call pattern is analyzed for network optimization and maximizing the prominence of service. In science, large amounts of data in astronomy, physics, and biology can only be analyzed quick enough by computers. The World Wide Web is enormous; it is persistently growing, and searching for appropriate information that cannot be done manually.

But Machine Learning is not just a database problem; it is also an ingredient of Artificial Intelligence. To be intelligent, a method that is in a varying environment should have the capability to learn. If the system can learn and acclimatize to such changes, the system designer need doesn't predict and present solutions for all possible situations.

Figure 1.7 shows the research domain of the thesis. We have to design a computer immune system by using the concept of human immunology, computer immunology, virus detection and constructive induction.

![Figure 1.7: Research Domain](image-url)
1.10 Organization of the Thesis

From the study of the Security of the computer system by various approaches, it is observed that many researchers and anti-virus developers have contributed to provide the best approaches to a computer system. But a question arises here in the mind of computer users that “Are these approaches and anti-virus tools are 100% truthful?” This thesis presents the investigation of Machine Learning and AIS approaches to provide the Security to a computer system.

The work reported in this thesis is organized in eight chapters, as follows-

**Chapter-1: Introduction**

This chapter presents an introduction and the key areas of the thesis.

**Chapter-2: Literature Survey**

This chapter presents an in-depth literature survey of the Computer Security approaches and its issues. This chapter starts with the basic Security issues.

**Chapter-3: Framework for Identification of Non-Self Operating System Process**

In this chapter, the aim is to establish the sense in a computer system that could differentiate between the Self process (i.e. Processes that are not harmful to our computer system) and the Non-Self process (i.e. Processes that are harmful and dangerous to our computer system). A process coming into the system is identified whether the process is part of the stable system, i.e. Self process or is it a harmful process which can threaten a computer system i.e. Non-Self process. The identification of Non-Self process can be done with the help of the Machine Learning approaches.

A framework is proposed for identification of Non-Self operating system process. The proposed framework will work on operating system process parameter’s values. In chapter 4 Decision Tree, in chapter 5 Artificial Neural Network, and in chapter 6 genetic algorithm are successfully implemented within the framework. These techniques would be used to classify the processes at process level into Self (non-harmful) and Non-Self (harmful or dangerous). This would help the system to sense the processes before the harmful processes do any harm to the system.
Chapter-4: Filtering Non-Self Operating System Processes Using Decision Tree

Through this chapter a learning system is being projected to identify the operating system processes as Self and Non-Self, by means of the concepts of Decision Tree Learning. ID3 algorithm will be used to create a Decision Tree after conniving the Entropy and Information Gain. Firstly Decision Trees are constructed by using training examples and then these constructed Decision Trees are tested with test data sets. Further, it has been contingent through investigation results that the Decision Tree Learning approach will offer better Security through efficient identification of Self and Non-Self processes.

Chapter-5: Immune-Neuro Approach For System Security

Through this chapter a learning system is being proposed to provide Security by identifying the operating system process as Self and Non-Self. Concepts of Artificial Neural Network (ANN) Learning has been used for the identification of processes. Initially, an Artificial Neural Network is created by using process parameters with random weights. These weights are updated by using Gradient Descent Algorithm and Backpropagation Algorithm for various training examples, and then this Artificial Neural Network is tested with test data examples. It has been observed that the Artificial Neural Network Learning provides a better approach for identifying Self and Non-Self process and provides a better Security.

Chapter-6: Adaptive Evolutionary System For Operating System Process Identification

In this chapter Detector are generated with the help of Genetic Algorithm will be proposed which will identify the operating system processes as Self and Non-Self. A process in a computer system is identified whether the process is part of the stable system, i.e. Self process or is it a harmful process which can destabilize a system i.e. Non-Self process. This task will be performed with the help of the Detectors generated by the Genetic Algorithm.

The first Detector set is generated randomly. After calculating the fitness of these randomly generated Detectors, the best fitness valued Detectors are used as a parent for the next generation. To generate the next set of Detectors Genetic crossover operators are used. Mutation is performed on new generated Detectors to maintain the diversity. Training will be provided to selected Detectors to identify the operating system processes. Proposed technique may be used to classify the processes at process level into Self (non-harmful) and Non-Self (harmful or dangerous). This would help the system to sense the processes before the harmful
processes do any harm to the system. The concept of Self and Non-Self is inspired from biological immune system.

Chapter-7: Conclusions and Future Work

The final segment of the thesis concludes the effort and addresses the upcoming scope of the future enhancements.