An Intrusion Detection Scheme Utilizing Teiresias to Determine a Computer's DNA Sequence Responsible for Network Traffic

By

Samuel Hu

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELEOR OF APPLIED SCIENCE

In the School of

Engineering Science

Simon Fraser University

Approval

Name: Samuel Hu

Degree: Bachelor of Applied Science

Title of Thesis: *An Intrusion Detection Scheme Utilizing Teiresias to Determine a Computer's DNA Sequence Responsible for Network Traffic*

Dr. J. Jones
Director
School of Engineering Science
Simon Fraser University

Examining Committee:

Chair and Technical Supervisor:

Dr. B. Yu
Program Head, Degree and Advanced Programs
Computer Systems Technology
British Columbia Institute of Technology

Academic Supervisor:

Dr. M. Parameswaran
Professor
School of Engineering Science
Simon Fraser University

Committee Member:

Mr. Steve Whitmore
Senior Lecturer
School of Engineering Science
Simon Fraser University

Date Approved:

Abstract

The research and development of pattern discovery algorithms is of utmost importance in the field of computer science. Recently, the researchers at IBM's Zurich Research Laboratory made a significant breakthrough in this field. Dubbed *Teiresias*, this algorithm was designed to detect variable length patterns in long sequences. Initially, this algorithm was used to detect sequences in human DNA and assist researchers in decoding the human genome. However, wider applications of this algorithm have also been researched. One such application is in the field of computer intrusion detection and security. In this thesis, I use the *Teiresias* algorithm to recognize patterns in network traffic in order to determine the presence of a computer intrusion.

Acknowledgements

I would like to thank all the members of my committee for their help and support: Dr. Yu for his ideas and technical advice, Mr. Whitmore for his helpful suggestions and corrections, and Dr. Parameswaran for his instruction and advice during my time at Simon Fraser University.

I would also like to thank my parents, brother and friends – without them, none of this would be possible.

Table of Contents

List of Figures

List of Tables

1. Introduction

With the increasing use of personal computers in business and home, the topic of computer security must be continually addressed. Only with adequate computer security can users be certain that their computers are not infected by malicious programs or being used for malicious purposes. In order for a computer to have a dangerous program installed on it or for it to be used inappropriately, it must first be intruded upon. Computer intrusion may occur via an e-mail attachment, a download from a website, physically via a disc, or by unauthorized access.

Currently, defence techniques against computer intrusion may take one of two general forms. First, an intrusion prevention scheme may be utilized in which viruses, other malicious software, and unauthorized users are prevented from entering one's computer. Protection of this nature can be accomplished with the use of password controlled access, software signature recognition, and firewalls. However, inevitably the best intrusion prevention systems will fail due to the continuous evolution of malicious software and the persistence of unauthorized users. Therefore, the use of a second defence technique known as intrusion detection is necessary. In this thesis, I define a unique method of determining if one's computer is being intruded upon or attacked. In particular, the focus is on computer intrusions and attacks that occur over the network to which the computer is attached. This method has been dubbed *Computer DNA Characterization*. Finally, this thesis illustrates preliminary results of *Computer DNA Characterization* and its ability to detect computer attacks. Although not entirely conclusive, the preliminary results provide a solid foundation on which to conduct further research.

2. Motivation

It has been shown that human beings contain genetic material, commonly referred to as Deoxyribonucleic Acid (DNA). Humans have 23 strand pairs of DNA which are composed of four fundamental building blocks known as *base pairs*. Variable length combinations of these base pairs generate *genes*. When genes manifest themselves by creating proteins, unique physical characteristics such as sex, hair colour, and height are realized. Recent research has shown that certain genes and mutations to other genes are responsible for generating certain conditions detrimental to human health. Although not entirely defined, it has been shown that various external factors can lead to mutations of human genes resulting in diseases.

The goal of the Human Genome Project is to discover the genes (sequences of base pairs) responsible for generating both positive and negative human characteristics. The premise behind the Human Genome Project was to examine the DNA sequences of various humans possessing the same characteristics and look for patterns (genes) that are common to all those possessing the characteristic. The benefits of decoding the human genome are widespread. Doctors and researchers will have a better understanding of diseases and will be able to better develop drugs and techniques that will treat the disease. Thus, early detection of diseases and those susceptible to diseases will be possible.

It has been proposed that a computer system can be given a *DNA Characterization* (Yu *et al*., 2001) that manifests itself in the way important characteristics of a computer system appear. Such computer characteristics include network traffic, modification of system files, and modification of data files. Each of these characteristics is determined by one of the computer's DNA sequences. In order to determine these DNA sequences, it is proposed that patterns in a computer's characteristics can be detected and thus shed light on the computer's DNA.

As with humans, the initial DNA sequences are predetermined on inception and ideally do not cause the system any harm. However, mutations to the DNA are possible, resulting in abnormal behaviour. It is proposed that malicious software or users that intrude upon a

computer system can have the capability of mutating a computer's normal characteristics thereby causing various degrees of damage. However, by recognizing intrusions early enough, the damage can be limited and quicker recovery is possible. Thus, the development of an effective intrusion detection system can serve as a deterrent to intrusions as well as prevent un-repairable and extensive damage.

3. Background Information

3.1 Computer Attacks

According to a 2001 report by the BBC, the number of computer attacks could reach 46,000 attacks in a year. This figure is consistent with the trends shown in Table 1.

Reported Computer Attacks (Year) (	Number of Attacks
1998	3734
1999	9859
2000	21756
January-September 2001	34754

Table 1 - Trends in Reported Computer Attacks[1]

However, even more disturbing are the costs associated with these attacks. According to a 2000 report by E-Commerce Times, "hacker attacks will cost the world economy a whopping $1.6 trillion (US$) this year."[2] The costs of computer attacks can be attributed to the loss of information, sales, data, and infrastructure. Needless to say, the prevention of computer attacks and intrusions is paramount.

Computers attached to a network can be intruded upon or attacked by various methods. For the purposes of this thesis, I refer to anyone who attempts to attack or intrude upon a computer as a *hacker*.

- **Denial of Service Attacks (DoS)** – Attacks of this nature are primarily used to *crash* a web site and remove it from the Internet. An example of such an attack occurred in February 2000 when a teenager using very simple DoS tools managed to cripple the web sites of large companies like Yahoo and Amazon.[3]

[1] http://news.bbc.co.uk/hi/english/sci/tech/newsid_1602000/1602493.stm
[2] http://www.ecommercetimes.com/perl/story/3741.html
[3] http://www.cnn.com/2000/TECH/computing/02/09/cyber.attacks.01/index.html

- **E-mail Viruses** – Attacks via electronic mail, such as the *MyParty* virus that recently circulated through the SFU engineering student maillist, generally act by attacking an e-mail program causing it to propagate the virus to all the people listed in the victim's address folder.
- **Backdoor/Trojan Attacks** – Named after the instrument of war used by the Greeks to gain access to the city of Troy, this type of attack is facilitated by installing a program on a compromised target system in order to allow re-entry into the system at another time. Hackers could access a system through backdoors, which have historically been included by system developers as a secret way into a system. An example of such an attack was the use of the *Notepad Worm* that is believed to have given hackers access to confidential files at Microsoft.
- **Password Cracking** – Although this method of attack is accomplished primarily by brute force, is can prove to be most dangerous if a hacker gains access to a system by using a legitimate password. This type of attack is virtually undetectable and can give the hacker access to all the system's files and resources.

3.2 Computer Intrusion and Attack Defence Methods

As mentioned previously, computer intrusion and attack defence methods can be broadly classified into two groups.

3.2.1 Computer Intrusion Prevention

Defence mechanisms of this nature are generally passive methods and have been used with various degrees of success for quite some time. Examples of such methods are the use of firewalls, virus scanners, and passwords.

- **Firewalls** – Generally, firewalls are computer network devices that protect one network from other less trusted networks. Essentially, a firewall permits or denies different types of traffic traveling into and out of an organization's network. According to the U.S. National Institute of Standards and Technology, "a well configured firewall will stop the majority of publicly available computer attacks."

- **Virus Scanners** – These widely available software packages attempt to recognize existing viruses and remove them to prevent them from spreading. Examining known viruses and then writing code to look for telltale characteristics of the viruses can accomplish this. Two major developers of anti-virus software are Norton and McAfee.
- **Passwords** – Although quite basic, this method of computer protection can be effective if the passwords are sufficiently encrypted and complex enough to prevent unauthorized users from gaining access to the passwords and, in turn, the systems that the passwords aim to protect.

Although very important, computer intrusion prevention does not adequately protect the integrity of a computer system for the reasons listed below. Further, most experts agree that computer intrusion prevention schemes are most effective when implemented concurrently with computer intrusion detection.

3.2.2 Computer Intrusion Detection

Compared to computer intrusion prevention, the field of detection is very much in its infancy. Generally, intrusion detection schemes can be divided into two categories: statistic-based tests and rule-based schemes.

- **Statistic-Based** – This type of intrusion detection involves the collection and monitoring of user characteristics on a system for a period of time. From this information, sophisticated profiles of the users are generated. Characteristics that can be monitored include applications that are spawned, network protocols that are utilized, and file creation, modification, and deletion. After a user profile is completed, subsequent characteristics of the system are recorded and compared to the original profile.
- **Rule-Based** – Rule-based intrusion detection is predicated on the assumption that any attempt to intrude upon a computer can be characterized by sequences of user activities that lead to compromised system states. Generally, the rules that define possible computer intrusion are based on known security flaws in the system and on past attacks. Once an adequate list of rules is compiled, rule-based intrusion detection schemes typically look at a computer system's audit trail for violations of the rules.

3.3 Current Implementations

Although in its infancy, many attempts have been made by researchers to develop intrusion detection based either on either statistics or rules.

Possibly the first attempt at defining a rule-based intrusion detection scheme was made by Dorothy Denning in 1987. In her paper, Denning proposes that for this model to be effective, one must first generate system profiles of the users. These profiles contain information such as system logins, command and program executions, file modifications, and device access. The basic idea is to then monitor the operations of the user on a target system looking only for deviations in usage. Denning also suggests several models of analyzing the data generated by the audit trails to determine that presence of anomalous behaviour. An operational model would simply compare the number of observations of a particular action against a pre-determined limit. Other possible models that Denning suggests include the *Mean and Standard Deviation Model* and the *Markov Process Model*. Although Denning does not offer an implementation of the system she describes, this paper provides a good framework for developing an intrusion detection system.

Another intrusion detection scheme was developed by researchers at the University of New Mexico which they believe tries to mimic the human immune system (Forrest *et al*., 1996.) This particular implementation relies on the assumption that system calls made by a computer occur in predefined sequences that can be recorded. For example, executing a typical UNIX process will result in the following system calls: *open, read, mmap, mmap, open, getrlimit, mmap, close*. The process can then be executed a sufficient number of times until a database of sequences of a fixed length is generated. Such a database for a sequence length of three (3) is shown in Table 2.

Call	Position 1	Position 2	Position 3
open	read getrlimit	mmap	mmap close
read	mmap	mmap	open
mmap	mmap open close	open getrlimit	mmap
getrlimit	mmap	close	
close			

Table 2 - Sequences of UNIX Commands by a Typical Process

For example, after an *open* command, either *read* or *getrlimit* will be called, followed by *mmap* and then *mmap* or *close*. The probability of each sequence occurring can be determined using various mathematical and statistical models described in the paper. As well as describing the implementation details of this intrusion detection scheme, Forrest *et al*., also present some preliminary results of testing to demonstrate this particular method's effectiveness.

Finally, an implementation of a computer intrusion detection system was developed by researchers at IBM using Teiresias (Wepsi *et al*., 1999). This algorithm is the same as the one utilized in this thesis. The basis of this scheme is similar to the one described above. However, one key distinction is that the sequences generated by Teiresias are of varying lengths, thereby generating fewer but most accurate sequences. A comparison of the two described schemes is shown in Table 3.

Set	Method	Mean Pattern Length	Number of Pattern	Uncovered Events (out of 26000)	False Positives	Attacks Detect (out of 10)
1	Fixed Length	3	114	185	0	8
2	Fixed Length	4	152	437	4	9
3	Variable Length	10	71	47	0	8

Table 3 - Comparison of Intrusion Detection Schemes

Two of main benefits of using variable length patterns can be seen in the above results. First, the variable length patterns are longer than fixed-length patterns. This point is important because longer sequences imply that fewer patterns are required to cover the set of sequences that can be called by the given process. Second, the number of uncovered events is greatly reduced by using the variable length patterns. The significance of the second point is that fewer 'unknown' and potentially harmful events will slip past the intrusion detection scheme.

3.4 Teiresias Algorithm

As mentioned, a research team at IBM in the Bioinformatics and Pattern Discovery Group first created the Teiresias algorithm (Rigoutsos *et al*., 1998). The algorithm was seen as a major breakthrough in assisting researchers concerned with finding patterns in human DNA. However, other applications of Teiresias have been found including applications in the field of computer security. The following briefly describes the algorithm and how it is utilized for the research conducted for this paper.

Although the source code for the algorithm is downloadable[4], for the purpose of this research, I will utilize the algorithm online via an IBM server. Unfortunately, using the downloadable version requires access to a UNIX machine that was not easily accessible. A screen capture of the User Interface is shown in Figure 1.

Teiresias can be used in one of two modes: *Exact Pattern Discovery* and *Homology-based Pattern Discovery*. For the research at hand, we are only concerned with Exact Pattern Discovery. To utilize the algorithm, one must first generate an input sequence of data values. For this particular application, the data is simply be a series of integers. More information about the data set is presented in Section 4. The three main parameters associated with Teiresias are shown in Table 4.

[4] http://cbcsrv.watson.ibm.com/Tipd.html

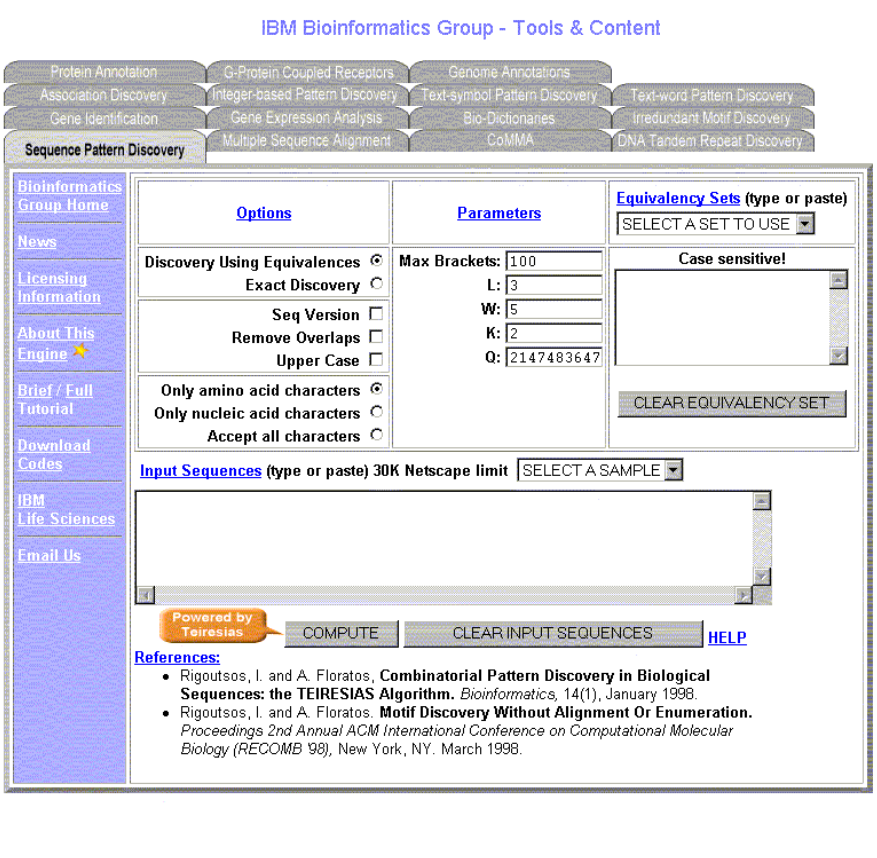

Figure 1 - User Interface of Teiresias Algorithm

L	Controls the minimum number of literals that must exist in a pattern
W	Controls the maximum extent spanned by any L consecutive literals in the reported patterns
K	Controls the support for a pattern (the minimum number of patterns that must exist)

Table 4 - Parameters Associated with Teiresias

The "K" parameter has two separate interpretations depending on the state of the SEQ_VERSION flag. If SEQ_VERSION is set, the problem being solved is "find all patterns that are included in at least K streams." Conversely, if SEQ_VERSION is not set, the problem being solved is "find all patterns that are included at least K times in the input stream." For the purposes of this research, the SEQ_VERSION flag is not set.

3.4.1 Example of Pattern Discovery Using Teiresias

As mentioned, a more detailed description of the data to be used for this research is presented in Section 4. However, an example of how Teiresias can be used to find patterns is included for clarification purposes. Also, the functionality of the parameters is demonstrated. A sample data set is shown in Figure 2.

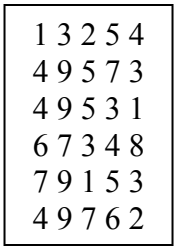

Figure 2 - Sample Data

Using Teiresias with various parameter settings, certain sequences or patterns were noted. A list of results is shown in Table 5 (note that not all parameter permutations were used.)

Parameter Values	Patterns Found
L = 2; W = 2; K = 3	[4 9] found 3 occurrences in 3 streams
L = 2; W = 2; K = 2	[4 9 5] found 2 occurrences in 2 streams [4 9] found in 3 occurrences in 3 streams [5 3] found in 2 occurrences in 2 streams [7 3] found in 2 occurrences in 2 streams
L = 2; W = 4; K =2	[4 9 5] found 2 occurrences in 2 streams [4 9] found in 3 occurrences in 3 streams [5 3] found in 2 occurrences in 2 streams [7 3] found in 2 occurrences in 2 streams [9. .3] found in 2 occurrences in 2 streams

Table 5 - Patterns Found after Applying Teiresias to Sample Data

From the above results, it can be seen that more stringent parameter setting leads to fewer patterns detected. The first row required that a pattern be found in at least three streams, and therefore, only one pattern was detected. Conversely, the last row parameters only require the

pattern to be found in two streams, and the final setting also allows up to two *wildcards*. Therefore, five separate patterns were detected using the least stringent parameter setting.

3.5 Basic Networking Concepts

In 1980, the International Organization for Standardization (ISO) proposed a model for computer communications referred to as the Open Systems Interconnection Reference Model (OSI.) OSI dictates that communications between two or more computers occur on seven separate *layers*. These layers are described briefly in Table 6.

OSI Model Layer	Description
7 – Application Layer	Layer acts as a window to the communications channel by interpreting information into a meaningful form that can be used by the application. Some common applications are web browsers, e-mail clients, and telnet clients.
6 – Presentation Layer	Layer deals with how data elements will be represented for transmission (i.e., order of data and format).
5 – Session Layer	Coordinates sessions between computers by initiating, maintaining, and managing communications.
4 – Transport Layer	Provides a reliable stream between communicating computers ensuring that information is delivered to the correct location on the destination computer. Also responsible for re-transmitting lost packets, ordering packets, and error checking.
3 – Network Layer	Responsible for moving data from one computer to another on a single network (i.e., an office LAN).
2 – Data Link layer	Used to join various networks together.
1 – Physical Layer	The physical medium used to transmit data (i.e., copper wire, fibre optics, or radio waves).

Table 6 - OSI Model of Computer Communications

3.5.1 Transport Layer Description

Although every layer of the OSI Model is vital for network communications, it can be argued that the "workhorse" layer is the transport layer. Within this layer, the main protocol utilized by thousands of applications is the Transmission Control Protocol (TCP.) TCP receives packages from various applications including web browsers and e-mail clients. TCP must then generate TCP packets by appending a TCP header to the front of the application-

generated package. TCP is also responsible for re-transmitting lost packets, correctly ordering packets, and error checking. Figure 3 illustrates the components of a TCP header. To better understand network communications and network attacks, several fields in the TCP header are examined in detail.

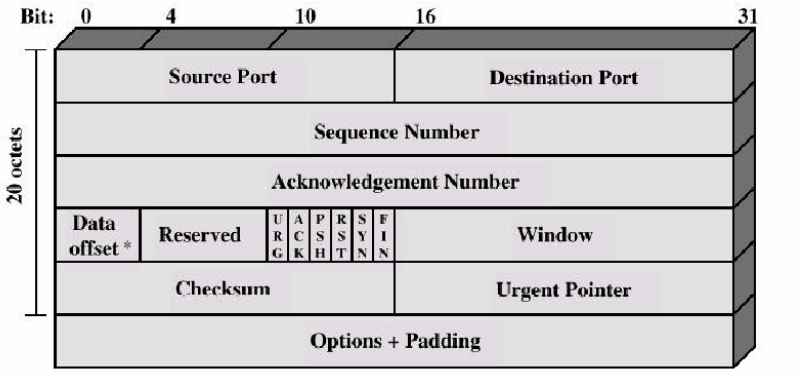

Figure 3 - TCP Header Structure

Within every TCP header are two port numbers denoting the source port and the destination port. A port can be seen as a door into a computer system and every machine has 65,535 ports. Applications that require network communications listen on a specific port for incoming TCP packets. Frequently used TCP port numbers include:

- TCP Port 80 – World Wide Web (HTTP)
- TCP Port 21 – File Transfer (FTP)
- TCP Port 23 – Telnet

For example, to request information from a web server such as www.sfu.ca, the requesting machine, via a web browsing application, would send out a packet to the SFU server with the destination port value of 80. Furthermore, TCP is responsible for dynamically generating a source port number (e.g., 1234.) When the SFU server responds with information, it will set its destination port number to 1234 and its source port number to 80.

Another key element of the TCP header is the control bits. These control or 'code' bits can be set independently and are critical in establishing legitimate communications. Of particular note are the ACK and SYN bits that are used to establish a connection via a "three-way

handshake." Figure 4 outlines the three steps that occur during the initialization stage of establishing network communications via TCP. The significance of TCP and how it relates to computer attacks is discussed in future sections.

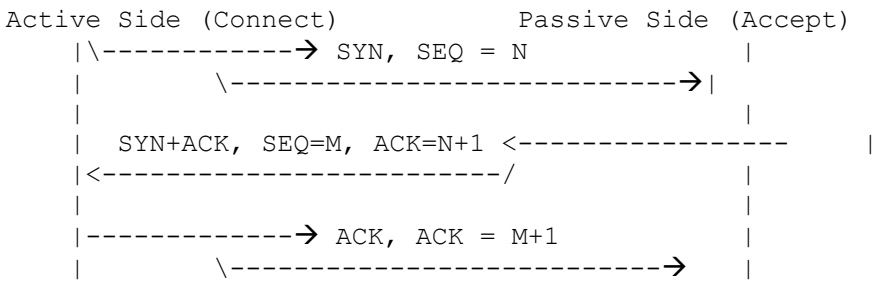

Figure 4 - TCP 3 Way Handshake

A close cousin to TCP is the User Datagram Protocol (UDP.) This protocol also operates at level four of the OSI Model but is generally more unreliable than TCP and is commonly used in many streaming audio and video applications as well as Domain Name Service (DNS) queries and responses. Figure 5 illustrates a UDP header and also demonstrates the simplicity of UDP versus TCP.

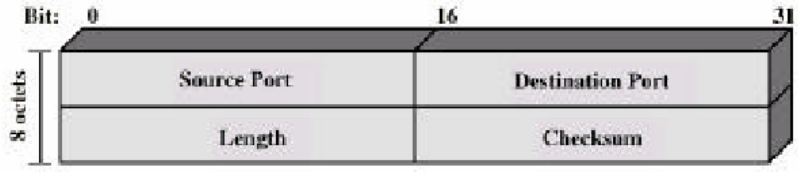

Figure 5 - UDP Header Structure

As well as utilizing a much simpler header, UDP does not require a 3-way handshake to establish communications. Therefore, applications that require simplicity and speed often use UDP. An example of such an application is *RealPlayer*. This application streams audio and video data from a server to a user's computer. UDP is used in this application to increase the speed of data transfer by eliminating the extra processing required by TCP communications. Further, human senses are able to compensate for the occasional lost packet that generally occurs when using UDP.

3.5.2 Internet Protocol and Internet Control Message Protocol

TCP, UDP, and other level four protocols establish communications between two systems. However, level three protocols, namely Internet Protocol (IP), ensure that packets leaving a system are carried to the appropriate destination system across a network. The task of transmitting a packet across a network uses a separate IP header that is appended to the level four (TCP or UDP) packet. Figure 6 illustrates the IP header.

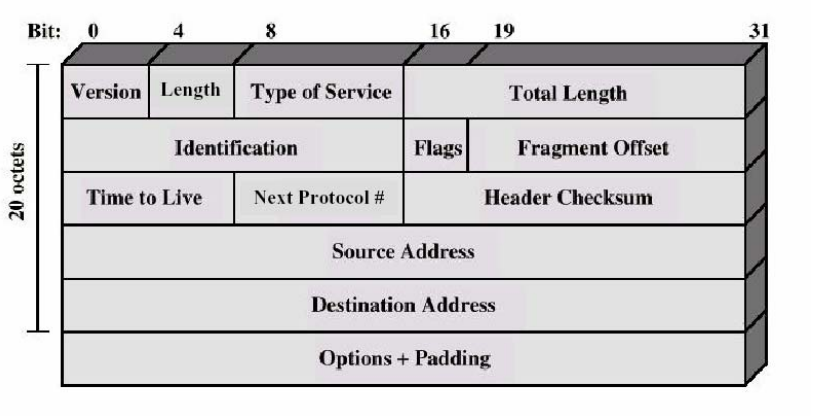

Figure 6 - IP Header Structure

Arguably, the most significant field in the IP header is the source and destination IP address. IP addresses, much like home addresses, identify a particular machine on a network. Every system directly connected to the Internet has a unique IP address that is 32 bits in length. Therefore, in order to attack a computer, the attacker must first determine the IP address of the target system. Further, many computer attacks utilize a technique known as "IP Address Spoofing." Although vital in the field of computer security, computer attacks and attack techniques at this level are not discussed in this thesis. Of note, however, is the Internet Control Message Protocol (ICMP.) This protocol also utilizes an IP header to direct a package from one system to another across a network. However, instead of appending a TCP or UDP packet to the IP header, an ICMP message is added. ICMP messages are generally used to transmit command and control information between systems to foster the transmission of data and report errors. There are many ICMP messages, some of which are listed in Table 7.

ICMP Message Type	Purpose of Message
Echo (Echo Reply)	Used to determine if a system is running and connected to the network. Also referred to as a ping message
Destination Unreachable	Indicates that an earlier IP message could not be delivered to the destination IP address
Timestamp (Timestamp Reply)	Used to determine the time of the destination machine
Information Request (Information Reply)	Used to determine the status of a network

Table 7 - ICMP Messages

3.5.3 Data Link and Physical Layer

Although layers two and one of the OSI Model are not be discussed in detail, a brief summary is included for the sake of completeness. The Data Link layer consists primarily of the software drivers used by the network interface cards (adaptors) and the physical layer consists of the network cards and the actual wires that make up the network. The most widely used technology at layers one and two is Ethernet. Ethernet utilizes a separate header that is appended to an IP packet and has its own *address* known as a Medium Access Control (MAC) address. Figure 7 illustrates the structure of an Ethernet header.

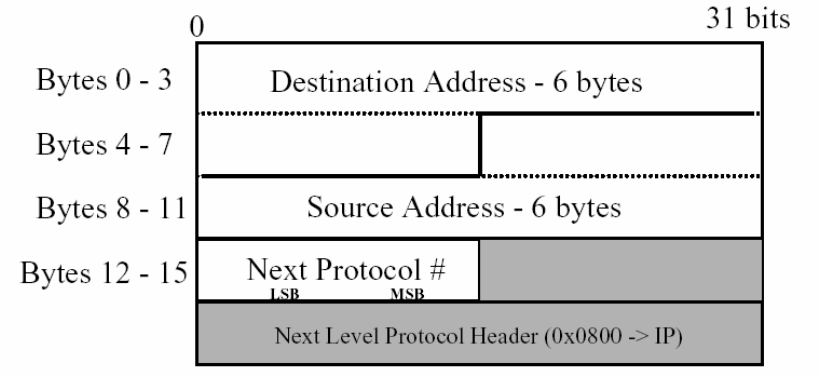

Figure 7 - Ethernet Header Structure

In summary, network communications occur by following the seven layer OSI model. Figure 8 illustrates a typical “packet” with the appropriate headers appended. Each layer in the OSI

Model is responsible for either appending or removing the appropriate information from each header, and in turn, reliable communications between systems over a network is established.

Ethernet Header	IP Header	TCP Header	Data Packet i.e. e-mail	Trailer

Figure 8 - Typical Packet Used for Network Communications

4. Computer Network Traffic DNA Sequences

As mentioned in Section 2, genes specify human characteristics. Figure 9 is a detailed illustration of a typical gene. Genes are composed of sequences of base pairs composed of four known bases: Adenine, Guanine, Cytosine, and Thymine. By examining a human's genetic composition, it is possible to predict certain human characteristics such as gender, height and hair colour.

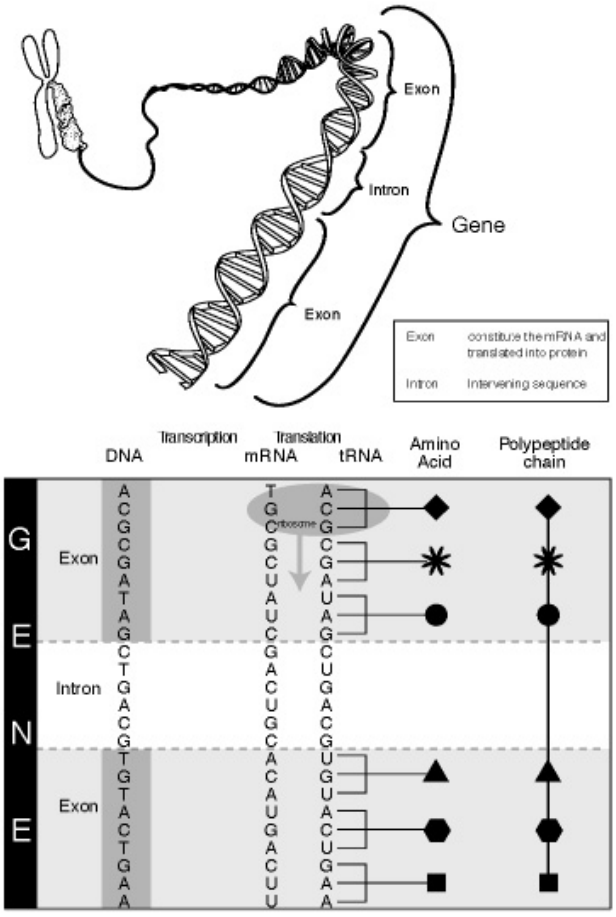

Figure 9 - Gene Structure[5]

Analogously, it has been proposed that there are DNA sequences (genes) that determine a computer system's characteristics. For the purposes of this thesis, the computer characteristic examined is computer network traffic.

[5] http://whyfiles.org/126dna_forensic/ 3.html

4.1 Base Structure Definition

Before a computer DNA sequence can be generated, the structure of the *base pairs* must first be determined. For this thesis, two *base* structures composed of different amounts of information are examined.

The first base structure contains only three pieces of information: amount of TCP, UDP, and ICMP traffic over a certain time period. Sequences generated from this base only present the most basic information about a computer system's network activity. A sequence of this type is illustrated in Figure 10.

Number of TCP packets	Number of UDP packets	Number of ICMP packets

Figure 10 - Fundamental Base Structure

To retrieve information pertaining to packet type, the PROTOCOL field in the IP header must be examined. Although various protocols exist, the three protocols of interest to this thesis are denoted by the protocol numbers outlined in Table 8.

Protocol Number	Packet Type
1	ICMP
6	TCP
17	UDP

Table 8 - Protocol Number Assignments

Although very basic in nature, sequences based on this fundamental base structure can provide important information about computer network usage. For example, a sequence may be generated for a home computer that is commonly used to retrieve information from the Internet but is not used to download streaming audio or video files. Therefore, this sequence would predict average TCP network traffic but minimal UDP network traffic. However, if a computer attacker attempts to access this particular computer via a UDP flood or a similar attack, the increase in UDP network traffic would be detected and the user can be alerted to a possible intrusion.

A second, more detailed base structure includes more information that can be collected by examining the TCP header. As with the fundamental base structure, this sequence will include information about the number of UDP packets and ICMP packets over a given time period. However, TCP packets are further separated into three segments: TCP packets with SYN flag set, TCP packets with ACK flag step, and TCP packets with both SYN and ACK flags set. Figure 11 illustrates this more detailed base structure.

Number of TCP packets	SYN packets	ACK packets	SYN+ACK packets	Number of UDP packets	Number of ICMP packets

Figure 11 - More Detailed Base Structure

The premise behind using a structure of this form is to detect a common computer attack known as a SYN flood. As described in the Basic Networking section, communications between two computers is initiated by a three-way handshake. When the destination host receives a TCP packet with the SYN flag set, the destination host replies with a SYN ACK packet and then waits for an ACK packet from a source host. While waiting for the ACK packet, a connection queue of finite size on the destination host keeps track of connections waiting to be completed. This queue typically empties quickly since the ACK is expected to arrive a few milliseconds after the SYN ACK.

A computer attacker exploits this design by generating numerous SYN packets with random source IP addresses. These packets are then directed towards a victim and the victim replies with a SYN ACK packet back to the random source IP addresses. Further, an entry is added to the victim's connection queue. Because the SYN ACK packet is send to random IP addresses, the victim does not receive the final ACK packet and the last part of the three-way handshake is never completed. However, the entry remains in the connection queue until a timer expires, typically for about one minute. When the connection queue is full, legitimate users will not be able to access TCP services such as e-mail and web browsing.

4.2 Computer DNA Sequence Generation

Once the base structures have been defined, sequences of these bases must be generated to form Computer DNA. How this thesis generates a computer DNA sequence is illustrated by example.

1. The first step in generating computer DNA involves the collection of the raw data pertaining to network activity. The method by which this is accomplished is discussed in Section 5.
2. After its collection, the raw data must be processed into the two base structures described above. Each base structure provides information pertaining to network activity for a predetermined time period. Figure 12 illustrates a few sample base structures. For example the base structure [10, 3, 1] would indicate that for a pre-determined time period, the computer being monitored received ten TCP packets, three UDP packets, and one ICMP packet.

Figure 12 - Sample Base Structures

3. After all base structures have been generated for a given time interval, repeated structures are determined via Teiresias. A sample output from Teiresias is illustrated in Figure 13. In this example, the base structure [10, 3, 1] occurs seventeen times in the given time period.

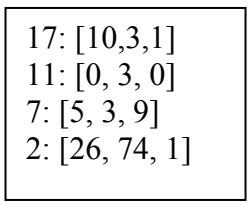

Figure 13 - Sample Teiresias Output

A collection of the commonly repeated base structures then forms the computer DNA sequence to predict computer network activity.

Once the computer DNA sequence is created, new base structures generated by real time network activity can be compared to structures contained within the DNA sequence. For, example, if a new base structure of the form [1000,500,40] is generated for the predetermined time period, but a base structure of this form is not found in the DNA sequence of the computer in question, a flag would be raised indicating a computer attack may be occurring.

4.3 Parameters in Computer DNA Sequence Generation

In generating computer DNA, several parameters must be set. The first parameter that must be determined is the time period allocated to a particular instance of a base structure. For example, each line of the sample base structures illustrated in Figure 12 indicates computer network activity for a time period of five seconds. Generally, if a longer time period is used to generate base structures, the number of packets that a computer system must process increases.

The second parameter that must be determined is the time period allocated to the collection of base structures used to create the computer system's DNA. As described in Section 4.2, base structures indicating computer network activity will be collected for a certain time interval (i.e., parameter in question). This collection of base structures is then processed by Teiresias to determine commonly repeated structures. Generally, if a longer time interval is allocated for base structure collection, the number of repeated base structures will increase, thereby generating a larger DNA sequence.

The final parameter that must be determined is referred to as the *Tolerance*. By strictly counting the number of packets received and sent during a certain time period, an exact number is generated. However, when attempting to find instances of base structures that repeat themselves, it is more desirable that similar sequences be treated as equal. Therefore, a tolerance factor is included to allow for similar numbers to be treated as equal. Essentially,

the tolerance factor is an integer used as a parameter of a function to round a number to the closest multiple of the tolerance factor. For example, one base structure may have the value [0, 38, 2] and another base structure may have the value [1, 39, 8]. Clearly, these two base structures indicate a similar amount of network activity. However, Teiresias does not detect similar patterns, and therefore, these particular base structures would be treated as distinct and not recognized as a repeated pattern. Using a tolerance factor of five, the values of these particular base structures would be [0, 40, 0] and [0, 40, 10], respectively. Although now the base structures are more similar, Teiresias would still not recognize the two as a repeated structure. However, with a tolerance factor of ten, the respective values of the two base structures would be [0, 40, 0] and [0, 40, 0], and the two structures would be recognized as a repeated structure.

Several factors must be examined when choosing the value of each of the parameters. The implications of varying the parameter values is discussed in depth in Sections 6 and 7.

5. Network Monitoring

5.1 Network Monitoring Tools

Because monitoring network traffic is crucial in many intrusion detection and prevention schemes, various companies and research institutions have developed tools to track a computer's network activity. Several such tools are TDIMON developed by Sysinternals, IPMON, and SNIFFIT. For this thesis, a utility named WINDUMP[6] developed by the University of California's Lawrence Berkeley Laboratory was utilized. WINDUMP is a utility ported for use on machines running Microsoft's various Windows operating systems and is modeled after the popular TCPDUMP utility that is commonly used on UNIX systems. Among its many features, WINDUMP has the ability to capture entire packets that are sent to and received by a particular network adapter. By analyzing the raw packets captured by a particular computer system, information required to generate the base structures, and in turn, the system's DNA sequence can be gathered. A sample output generated by WINDUMP is shown in Figure 14.

```
21:37:30.130157 oemcomputer.bc.hsia.telus.net.1052 >
helium.bc.tac.net.53: 557+ (43)
                        4500 0047 9ba9 0000 8011 bf2b 40b4 c965
                        d135 0482 041c 0035 0033 c54d 022d 0100
                        0001 0000 0000 0000 0236 3102 3638
21:37:30.789342 helium.bc.tac.net.53 >
oemcomputer.bc.hsia.telus.net.1052: 557* 1/2/2 (150) (DF) [tos 0x28]
                        4528 00b2 2a6c 4000 fc11 73d5 d135 0482
                        40b4 c965 0035 041c 009e 6377 022d 8580
                        0001 0001 0002 0002 0236 3102 3638
```

Figure 14 - Sample WINDUMP Output

The output shown in Figure 14 can be generated by using the command line input:

.\windump –x –s 60 > data.txt

[6] http://windump.polito.it/install/default.htm

This command prints up to sixty bytes of raw packet information to a file named “data.txt.” The pertinent information can be extracted from the data file using utilities written for this thesis.

5.2 Network Monitoring Location

In order to determine the accuracy of the computer DNA sequence generated by procedures outlined in this thesis, network traffic from various sources should be collected and analyzed. However, for the purposes of this thesis, a single source of data is analyzed. This source is a network switch used to connect two home computers to the Internet via an ADSL modem. The five inhabitants of this particular residence each use the computers to access online information to varying degrees. Common uses of the computers include checking electronic mail, web-surfing, and file share via peer-to-peer networks. Unfortunately, by only analyzing one source of network traffic, the results generated by this thesis are limited. Further discussion of this topic is conducted in Section 8.

6. Data Collected

One of the key parameters in determining a computer's DNA sequence is the time allocated to collecting base structures eventually leading to the DNA sequence. For this thesis, it is initially proposed that a time of four (4) hours will be sufficient to collect enough pertinent network traffic data. Theoretically, if a longer time period is allocated for the collection of initial base structures, more repeated structures would be found, thereby creating a more detailed DNA sequence. However, it is also vital to quickly generate a sequence and then utilize it to detect possible computer attacks.

The tolerance parameter is also crucial in defining a computer's DNA sequence. For the initial DNA sequence generated for the home network switch, a tolerance factor of 10 was used. Finally, the time difference used for the generation of a specific instance of a base structure needs to be set. Again, for the generation of the initial DNA sequence, a time difference of 10 seconds was used.

Based on the parameter settings as outlined above, data was collected and formatted for input to the Teiresias Algorithm. As mentioned in Section 3.4, Teiresias possesses several options and parameters that need to be set. Because this thesis is only concerned with exact pattern recognition, only the 'K' parameter must be set. For the initial DNA sequence, a K value of 2 was used signifying that a pattern must repeat itself twice in order to be recognized. Figure 15 illustrates the initial three-element base structure DNA sequence generated for the home network switch.

```
0 0 0
10 0 0
30 0 0
40 0 0
80 0 0
20 0 0
50 0 0
130 0 0
110 0 0
360 0 0
370 0 0
20 10 0
```

Figure 15 - Initial DNA Sequence for Home Computer Network Switch

The initial six element base structure DNA sequence generated for the home network switch is shown in Figure 16.

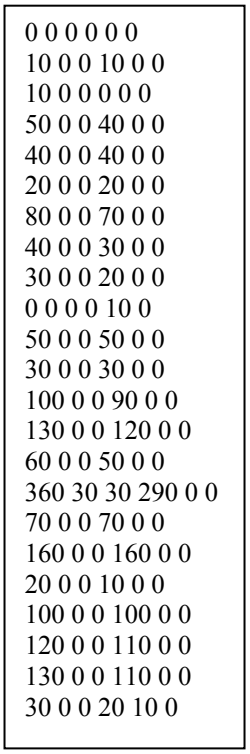

Figure 16 - DNA Sequence for Home Network Switch

Section 7 outlines several results of tests conducted on the initial DNA sequence and also includes several proposed modifications to the initial DNA sequence in terms of parameter selection.

7. Results and Analysis

7.1 Testing Procedure

After the initial DNA sequences were generated for the home network switch, the next step was to determine the usefulness of the generated sequences in predicting future levels of computer network activity. To accomplish this task, continued network monitoring of the home network switch was required. Subsequent base structures generated by the continued network monitoring were compared against the initial generated DNA sequence to determine if the generated DNA sequences are able to predict future levels of network activity.

In order to quantify the effectiveness of a computer intrusion detection method, two main criteria are examined. False positives occur when a non-existent intrusion is detected. This condition generally occurs when the threshold for an intrusion is set too low. False positives are dangerous in that they raise unnecessary worry and concern for the users of the systems. Another criteria are false negatives that occur when an intrusion is not detected by the intrusion detection scheme. Needless to say, false negatives are dangerous because potentially devastating intrusions are not detected. Therefore, the common goal in designing an intrusion detection system is to limit the number of both false positives and false negatives.

7.2 Results for Home Network Switch

In generating the DNA sequence for the home network location, a very low tolerance factor was used. Subsequently, many base structures occurred during normal computer usage not predicted by the generated DNA sequence. Some examples of the errors that were detected are shown in Figure 17 and Figure 18.

found error in base structure:
60 0 0 60 0 0
found error in base structure:
590 0 0 580 0 0
found error in base structure:
1610 0 0 1610 0 0
found error in base structure:
1550 0 0 1550 0 0
found error in base structure:
1570 0 0 1570 0 0
found error in base structure:
1600 0 0 1600 0 0
found error in base structure:
1580 0 0 1580 0 0
found error in base structure:
1590 0 0 1590 0 0
found error in base structure:
1590 0 0 1590 0 0

Figure 17 - Error Six Element Base Structures

found error in base structure:
0 10 0
found error in base structure:
100 0 0
found error in base structure:
70 0 0
found error in base structure:
120 0 0
found error in base structure:
600 40 0
found error in base structure:
320 0 0
found error in base structure:
240 20 0
found error in base structure:
40 10 0
found error in base structure:
80 10 0
found error in base structure:
90 0 0

Figure 18 - Error Three Element Base Structures

Based on the results and errors detected, several key conclusions were derived. First, several of the detected errors closely resemble base structures found in the DNA sequences. For example, the base structure [70, 0, 0] generated an error although the base structure [60, 0, 0] was present in the DNA sequence. Clearly, both base structures represent a similar amount of

network activity, and the error generated by the base structure [70, 0, 0] was obviously a false positive. From this fact, a conclusion was derived specifying that the tolerance factor was far too strict. However, some of the errors such as [1600, 0, 0, 1600, 0, 0] indicated an abnormally high amount of network traffic. At first, structures of this volume were treated as suspicious. But after some investigation into computer usage the day the data was collected, it was determined that file sharing over a peer-to-peer network had occurred during the time period in question, thereby drastically increasing the volume of network traffic. However, during the four-hour time period where the initial sequence was generated, no file sharing was conducted, and thus, the high level of network activity was not recorded. Therefore, it was concluded that the four hours of initial data collection was insufficient to adequately capture all aspects of network traffic activity.

With these new initial conclusions in mind, a new DNA sequence was generated using a tolerance factor of one hundred (100) and approximately eight hours of initial data. The second sets of generated sequences are shown in Figure 19.

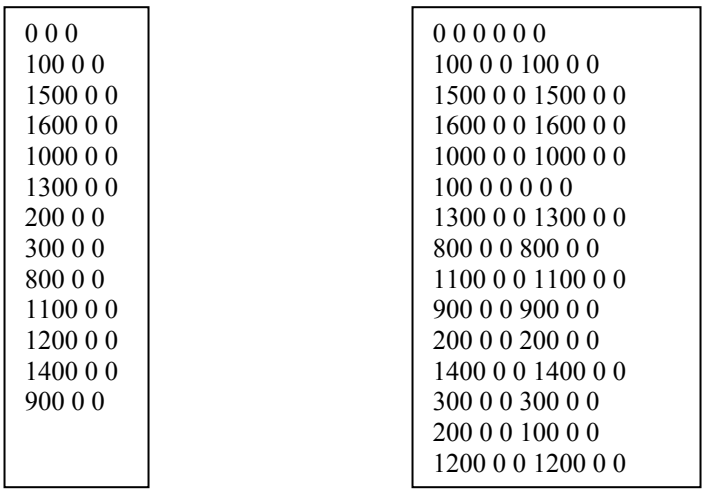

Figure 19 - Second Generated DNA sequences

Again subsequent network monitoring was required to determine the accuracy of the second set of generated DNA sequences. The results of the second round of testing demonstrated a drastic improvement mainly in the number of false positives that were detected. Figure 20 illustrates the errors that were detected. Unfortunately, the existence of false positives is still a concern, and a discussion of possible solutions is conducted in Section 8.

found error in base structure:
600 0 0

found error in base structure:
600 0 0 500 0 0
found error in base structure:
300 0 0 200 0 0

Figure 20 - Errors Detected in Second Round

A final conclusion based on the results illustrated above, as well as on other tests carried out, is that the sequences containing three element base structures generally produce fewer false positives than the sequences containing six element base structures. Further, the sequence files containing six element base structures were much larger and required more processing power to generate and analyze. However, the benefits of the larger sequence were virtually non-existent. Further discussion of various base structures is conducted in Section 8.

7.3 Detection of Computer Attacks

As well as being able to predict future levels of computer network activity, a computer DNA sequence must be able to detect potential computer attacks. Unfortunately, conducting an actual computer attack is rather difficult to regulate and may result in damage to the victim computer. However, under monitored conditions and directed supervision, a UDP flood attack was carried out on an undisclosed server computer. The resulting computer network activity is shown in Figure 21.

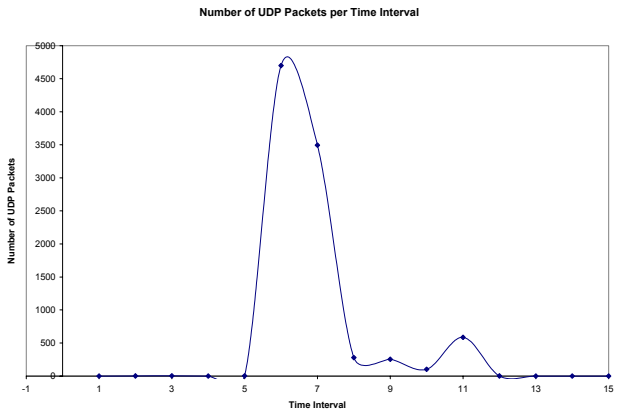

Figure 21 - Packet Count during UDP Flood

Clearly, the data used illustrate a computer attack in Figure 21 would not be predicted by the DNA sequence generated for the home network switch, and therefore, would be flagged as abnormal behavior.

Another clearly illustrated example of a computer attack is the Denial of Service attack carried out against the Gibson Research Corporation (GRC) in 2001[7]. GRC is connected to the Internet via a pair of T1 trunks that provide 3.08 megabits per second of bandwidth for network communications, which according to the GRC website is "ample for [their] needs." Therefore a DNA sequence for the GRC server would predict levels of network activity of well below two thousand packets for a ten second time period However, during the peaks of the attacks, roughly one hundred and ninety thousand packets were being directed to the GRC server every ten seconds. Again, the DNA sequence for the GRC server would not predict such high levels of network activity, and a flag indicating an attack would be raised.

Finally, Internet worms are another form of common computer attacks that would produce abnormal activity levels that would be detected via a computer DNA sequence. Attacks of this nature generally try to propagate by sending malicious code to all the e-mail addresses present in the victim's address book. Normal computer users rarely (if ever) try to send e-mail to everyone they know simultaneously. Therefore, the action in question would not be predicted by a DNA sequence, and a flag would be raised if the computer in question were infected with such a virus.

[7] http://grc.com/dos/grcdos.htm

8. Areas of Further Research

The initial results of an attempt to generate a computer DNA sequence to predict computer network activity proved to be successful. However, several areas of research still need to be examined in order to draw reliable conclusions about the accuracy and usefulness of computer DNA characterization.

8.1 Various Monitoring Locations

The positive results demonstrated in Section 7 outlined the usefulness of a computer DNA sequence for a switch connecting a home network to the Internet. However, a vital requirement to determine the overall effectiveness of computer DNA characterization is the ability to generate accurate computer DNA sequences for various types of computers. In particular, network monitoring of computers used by various types of professionals as well as server computers is desirable. However, because network monitoring for the purposes of this thesis requires collecting packet information that may be of a sensitive nature, it was difficult to obtain permission to monitor various networks.

8.2 Collection of Data Required to Generate DNA Sequence

As described in Section 7, the initial proposition that four hours of data collection would be sufficient to generate a complete DNA sequence was shown to be inaccurate. Further, the second generated sequence using a longer collection time period was shown to be more accurate in predicting future levels of network activity. However, more research is required to conclude the ideal time required to collect sufficient data to generate an accurate DNA sequence. Two other factors must be examined when collecting data to generate a computer DNA sequence. First, great care must be applied in order to prevent an attack during the initial collection of data. If an attack is underway during the initial collection of data, the resulting network activity levels will appear in the generated DNA sequence. Therefore, future attacks will appear normal resulting in dangerous false negatives. Second, another method of initial data collection could be conceived in which the computer in question is

directed to undertake all normal network activities, thereby generating a more accurate DNA sequence. However, this method may be more time consuming and require more user input and participation.

8.3 Parameter Selection

Again, as illustrated in Section 7, one of the key parameters that was varied in order to generate a more accurate DNA sequence was the tolerance factor. Specifically, a higher tolerance factor generated a computer DNA sequence that more accurately predicted future network activity with far fewer false positives. However, the tolerance factor parameter is highly specific and depends on factors such as network activity levels and user's security requirements. Specifically, if security requirements are higher, a lower tolerance factor should be used to generate a more specific DNA sequence capable of detecting smaller variations in network activity. Also, if network activity volume is high, a large tolerance factor should be utilized to limit the number of false positives.

Another parameter that was not examined in depth is the time allocated per base structure. Although a time period of ten seconds was used effectively through out research conducted for this thesis, future research may generate other results indicating a more effective value for this parameter. For instance, if the computer in question is subject to continuously high levels of network activity, a shorter time period may be more effective.

8.4 Other Base Structures

Section 7 outlines the results of generating DNA sequences using base structures of the form illustrated in Figure 10 and Figure 11. Further, it has been shown that the more simple base structure consisting of three elements of information is as effective as the slightly more detailed base structure described by Figure 11. However, it may be possible for base structures of a completely different nature to generate more accurate and useful DNA sequences. For example, a base structure containing information about the computer ports being used for communications may be beneficial. As mentioned in Section 3.5, network communications

occur via commonly used ports such as port 23, 80 and 25. Therefore, a base structure could enumerate the number of packets processed by commonly used ports as well as enumerate packets received and sent by other ports. An example of such a base structure is shown in Figure 22.

# of port 80 packets (HTTP)	# of port 25 packets (SMTP)	# of port 21 packets (FTP)	# of port 23 packets (TELNET)	# of other packets

Figure 22 - Possible Base Structure

DNA sequences based on the base structure shown in Figure 22 may be more beneficial in detecting common network attacks. For example, the attack carried out against GRC (Section 7.3) aimed malicious packets at port 666 – an uncommonly used port. Further, the majority of Internet worms re-transmit themselves using common e-mail clients such as Microsoft Outlook that communicates via port 25 using the Simple Mail Transport Protocol.

8.5 Evolution of a Computer DNA sequence

In terms of future research, the area that will be most beneficial is the topic of computer DNA evolution. Specifically, once the initial DNA sequence is generated, a mechanism needs to be implemented that adds to the sequence commonly repeated base structures that are not currently present. Further, base structures that are initially included in the DNA sequence, but are not commonly observed by future network monitoring, should be eliminated to generate a more accurate DNA sequence. By continuously evolving, the DNA sequence for a computer will more accurately predict the levels of computer network activity currently being experienced by adapting to changes in users and user activities.

8.6 Ability to Detect Attacks

Finally, in order to confidently conclude that computer DNA characterization is an effective means of network intrusion detection, a system must be given a DNA sequence and then exposed to various network attacks. Such testing would require strict supervision as well as the required resources to conduct such attacks.

9. Conclusion

In summary, the research that was conducted during the completion of this thesis centred primarily on the area of computer network and security. Basic networking concepts were demonstrated as well as background information pertaining to computer attacks and current intrusion detection schemes. Further, it was proposed that a computer DNA sequence can be generated for various computer characteristics, and a method for generating a DNA sequence using the *Teiresias* algorithm was outlined. The computer characteristic of particular interest to this thesis was computer network activity. Through the research conducted and by several examples, it was shown that generating a DNA sequence which accurately predicts future levels of computer network activity is possible. Furthermore, it has been shown by example that a computer DNA sequence is able to detect common computer attacks. Finally, several areas of future research have been highlighted.

During the past ten months, I have undertaken the various tasks and conducted detailed research to produce the results illustrated in this thesis. However, the research conducted for this thesis was primarily exploratory in nature and much work is still required to verify the usefulness of computer DNA characterization. Nevertheless, I am confident that advances in computer security will be achieved using some of the results I have produced.

10. References

Copeland, J.A. 2001. *Ethernet/IP/TCP Packet Headers.* Georgia Institute of Technology, Atlanta, Georgia.

Denning, D. February 1987. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering.* Vol. SE-13, NO. 2, 222-232.

Forrest, S., S.A. Hofmeyr, A. Somayaji, T.A. Longstaff. 1996. A Sense of Self for UNIX Processes. *Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy*.

Rigoutsos, I., A. Floratos. 1998. Combinatorial Pattern Discovery In Biological Sequences: The TEIRESIAS Algorithm. *Bioinformatics*, Vol.14, num. 1.

Skoudis, E. 2002. *Counter Hack*. New Jersey: Prentice-Hall, Inc.

Wepsi, A., M. Dacier, H. Debar. 1999. An Intrusion-Detection System Based on the Teiresias Pattern-Discovery Algorithm. *EICAR Proceedings 1999.*

Yu, B., E. Byres, C. Howey. 2001. *Monitoring Controller's "DNA Sequence" for System Security.* BCIT, Burnaby, BC.