

Simulations on Early Warning Malware Identification System Based on Peer to Peer Networks

BOUNTOLOS CHARALABOS

Master of Science



T.E.I. OF LARISSA



STAFFORDSHIRE UNIVERSITY

TECHNOLOGICAL EDUCATION INSTITUTE OF LARISSA

June 2014

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognize that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without proper acknowledgement.

TECHNOLOGICAL EDUCATION INSTITUTE OF LARISSA
THESIS DISSERTATION

Simulations on Early Warning Malware Identification System Based on Peer to Peer Networks

BOUNTOLOS CHARALABOS

Master of Science, 2014

Inquiry Commission:

1. Reviewer: Vasileios Vlachos
2. Reviewer: Alexandros Papanikolaou
3. Reviewer:

ABSTRACT

The outbreak of malicious activity over the last years due to the rapid spread of the Internet and the emergence of a new generation of rapidly spreading viral software have made insufficient the use of conventional protection programs. The PROMIS algorithm (PROactive Malware Identification System) that we propose intends to identify the malicious activity on the Internet, using an architecture based on peer-to-peer networks and downloading appropriate restrictive security measures to protect computer systems.

The PROMIS algorithm creates a peer to peer network in which its members exchange information on security incidents that they record locally, in order to extract the overall malicious activity of the Internet. Each node of the peer-to-peer network captures different security incidents and communicates with a finite number of other nodes.

The aim of the system is not to protect all the members from specific threats, but as introducing the concept of Computer Hygiene, the members of the peer-to-peer network automatically disable all the helpful but not the critical applications during a malware epidemic that is on the rise. Later, when the phenomenon presents recession, the specific services reactivated. Therefore, a further differentiation of the system is aimed at the end users as opposed to most other implementations, which are installed in a network infrastructure (e.g., routers). In addition, because the entire process is automated, it can be used by home users who do not have the necessary technical knowledge to protect themselves sufficiently.

In order to verify and evaluate the effectiveness of the algorithm, a large number of simulations are performed to obtain and analyze the necessary experimental data. For the creation of the appropriate graphs for being as realistic as possible to the existing network topologies, the NGCE tool (Network Graphs for Computer Epidemiologists) was used. The PROMISsim simulator was used to perform the simulations in which all the experimental measurements were made. The results support the usefulness of the algorithm.

The experimental results converge that the PROMIS algorithm is able to protect the systems from a sufficient number of malware outbreaks. The effectiveness of the algorithm depends on various inherent and environmental factors, which were

scrutinized, but particularly aggressive security policies can protect almost all the members of the peer network.

KEYWORDS: Peer to Peer Networks, Malware, Network Security, Computers
Worms, Computer Epidemiology, Simulations

*To my parents, my
grandmother and Danae.*

ACKNOWLEDGEMENTS

Reaching almost at the end of this journey, first and foremost, I would like to express my sincere gratitude to my advisor Dr. Vasileios Vlachos, who has been a tremendous mentor for me. I would like to thank him for his patience, motivation, enthusiasm, and immense knowledge. His advice on both research as well as on my career have been priceless.

Besides my advisor, I would like to thank the rest of my master's professors: Dr. Alexandros Papanikolaou, Dr. Costas Kokkinos and Dr. Costas Chaikalis, for their encouragement, insightful comments and excellent cooperation. My sincere thanks also goes to Dr. Nicholas S. Samaras, for his encouragement, continual help and advice.

I would like to express my gratitude to Kyprianos Vasilopoulos and Pantelis Reditis for their great inspiration, motivation and guidance. Also, many thanks to Fotis Liatsis for his great friendship, patience and support in my research.

A special thanks to my family. Words cannot express how grateful I am to my parents and my grandmother for all of the sacrifices that they have made on my behalf. The most special thanks goes to Danae for her endless love and support.

At the end, thanks to you, reader. If you are reading this line after the others, you at least read one page of my thesis. Thank You.

'The only secure computer is one that's unplugged, locked in a safe, and buried 20 feet under the ground in a secret location...and I'm not even too sure about that one.'

--Attributed Dennis Huges FBI

List of Symbols / Abbreviations

DDoS	Distributed Denial of Service (attack)
DoS	Denial of Service (attack)
P2P	Peer-to-Peer
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Table of Contents

List of Symbols / Abbreviations	8
CHAPTER 1	13
INTRODUCTION	13
1.1 Aim of Thesis	16
1.2 Novel Features of Thesis	16
1.3 Outline of Thesis.....	17
CHAPTER 2	18
PRELIMINARY CONCEPTS - BACKGROUND	18
2.1 Introduction.....	18
2.2 Basic types of malware	18
2.2.1 Virus.....	21
2.2.2 Worm.....	23
2.3 Characteristics and factors that propagate malware	25
2.3.1 Main gateway	25
2.3.2 Selection methods of target discovery.....	25
2.3.2.1 Random Scanning.....	26
2.3.2.2 Localized Scanning.....	26
2.3.2.3 Hit List Scanning	27
2.3.2.4 Topological scanning	27
2.3.2.5 Permutation scanning	28
2.3.2.6 Multi-vector worms.....	28
2.3.2.7 Theoretical worms.....	28
2.3.2.8 Tarpits and Honeypots	28
2.3.3 Infection Rate	29
2.4 Techniques for Malware Containment.....	30
2.4.1 Categories of containment for malware techniques.....	31
2.5 The Evolution of worms.....	32
CHAPTER 3	34
NETWORKS, EPIDEMIOLOGY AND EPIDEMIOLOGICAL MODELS.....	34
Introduction.....	34
3.1 Networks	35
3.1.1 Random Networks.....	35
3.1.2 Spatial Networks.....	36
3.1.3 Scale-Free Networks.....	37
3.1.4 Exponential random graph models	38

3.2 Epidemiology	39
Introduction in Epidimology - Biology and Computers	39
3.2.1 Epidemiology in Biology	39
3.2.2 Epidemiology and Computers	40
3.3 Epidemiological Models	40
3.3.1 Introduction in epidemiological models.....	40
3.3.2 Goals and limitations of epidemiological modeling	42
3.4 Basic Epidemiological Models	45
3.4.1 Susceptible - Infectious (SI) Model.....	45
3.4.2 Susceptible - Infected - Recovered (SIR) Model	47
3.4.3 Susceptible - Infected - Susceptible (SIS) Model	48
3.4.4 Susceptible - Infectious – Detected - Removed (SIDR) Model	50
3.4.5 Susceptible-Infected-Removed-Susceptible Model (SIRS)	51
CHAPTER 4.....	52
STATE OF THE ART	52
4.1 Introduction.....	52
4.2 Systems based on Peer-To-Peer (P2P) Networks.....	53
4.3 Centralized Systems with Distributed Sensor Networks	55
4.4 Alternative Implementations	57
CHAPTER 5.....	59
PROBLEM STATEMENT AND METHODOLOGY CHOICE	59
5.1 Introduction.....	59
5.2 Problem Statement	60
5.3 Methodology Choice	61
5.3.1 Choosing simulated over realistic experiments	61
5.3.2 Evaluation of simulations	63
5.4 PROMIS Algorithm.....	66
5.5 NGCE - Network Graphs for Computer Epidemiologists	68
5.6 PROMISsim	70
CHAPTER 6.....	72
SIMULATION RESULTS	72
Gephi	72
6.1 Network Topologies	73
6.1.1 Homogeneous graphs.....	73
6.1.2 Random Nodes	74
6.1.3 Full Scale Free Graphs	80

6.2 PROMISsim	85
6.3 Threshold of Security Levels.....	92
6.4 Maximum and minimum number of communicating nodes	97
CHAPTER 7	99
7.1 The aim of the Thesis.....	99
7.2 Recommendations for Future Research.....	99
7.3 Conclusions.....	99
REFERENCES	100
APPENDIX	108

CHAPTER 1

INTRODUCTION

The cost reduction of personal computers and the ease of use that have been resulted from the usage of programs which are friendly to the user, have caused the rapid spread and have made personal computers accessible to a wide number of people who do not have any special technological background. In addition, the rapid spread of the Internet and particularly the presence of broadband connections in an affordable cost, but also the rapid increase of wireless connections have enabled a large percentage of users to be connected. The benefits of these developments are evident.

The last 15 years, it became very clear that the Internet constitutes a major business and financial field that is daily traded on huge amount of information. The widespread use of distributed databases, distributed computing and applications of telecommunications finds direct implementation and it constitutes a fundamental element in the communications, defense, banks, stock exchanges, health, education and other important sectors.

However, the dependence of our society on such a degree of computing systems makes it vulnerable to threats from malicious software. But in order to protect something we must first need to understand and analyze what is threatened. It easily can be seen that the economic and political significance is huge and it is logical to have caused the interest of any kind of criminal activity [1, 2]. The incentives are obvious, but some examples of objectives are:

Identity theft: Infecting a personal computer with some type of malicious software that has the same functionality like Trojan horses, a criminal can collect all the personal details of the holder and use them to be benefited. Although this can be done with a message-mail, it has the ability to divert the user's navigation of legitimate sites that wants to visit, in identical false in order to intercept the data [3, 4].

Cyber Scams and Spam Email: If the more savvy users seem unthinkable to trust suspicious emails and transact with their senders, it has been accepted that even a small percentage is enough to yield significant benefits to senders.

Cyber Extortion: Particular preference is given to sites of betting companies before the start of a match, forcing their owners to pay without any thought in order not to be offline the days that are expected to do the maximum earning. To achieve this, various cyber criminals infect with malware a few thousand computers, creating networks of malicious agents (botnets) which perform Distributed Denial of Service Attacks (DDoS-Attacks) [5, 6].

Cyber-Terrorism and Cyber-War: It has been accepted that the Cyber-War and Cyber-Terrorism are important components of the modern world and can develop effective malware which is one of the most important and powerful cyber weapons [3, 7, 8]. It is also possible that this kind of attack can be used by several extremist groups in the future, as it doesn't require significant costs or large groups to perform the attack, but some talented developers who will develop the appropriate malicious software [9].

The last decade, for the first time the scientists were able to predict the techniques that should be followed in the development of malware [10, 11]. Researchers have demonstrated that it is feasible to construct malicious software that can infect rapidly all vulnerable targets in the Internet [12, 13, 14]. These forms of malware that are contained in the category of network worms, called Warhol worms, because it takes just 15 minutes to hit all the exposed targets, while even more dangerous are the Flash worms that can have equally devastating in just one minute [10, 11].

New forms of malware can be spread much faster than the extraction of the appropriate signatures and the updates of antivirus programs and other security applications, requiring dozens of hours to be completed [15, 11, 16, 17]. So, it is very likely that when the security applications will be updated, the systems which are protected, are already infected to the virus [18].

The ominous predictions of scientists began to be confirmed in 2001 and later, as there were many network worms with highly sophisticated features, causing rapidly epidemic infections. The most known cases that are referred to the epidemics were caused by the network worms Code Red, Code Red v2, Code Red 2 [19, 20, 21, 22] and Nimda [23, 24, 25]. The scale of damage has highlighted the lack of effective means of response.

Lack of appropriate mechanisms for dealing with rapidly spreading malware can be confirmed by the fact that 3-4 years after these events, new network worms such as Slammer [26] and Blaster [27], failed to cause more damage in less time, while recently network worms succeeded to spread in almost all the vulnerable population in minimum time like the Witty [28], proving that still now there is no complete way of reaction.

The major problem for neutralizing the threat from the latest generation of malware is the extreme pressing time for its interception [18]. The restriction is possible only in the first minutes of its action, and at this time, the existing security mechanisms cannot be operated effectively. Addressing this issue is directly related to the resources that are protected, but also with the way in which it pursues to protect.

Additionally, as the time needed for security applications and antivirus programs to create appropriate signatures and sufficiently protect the vulnerable systems, is greater than the time needed by modern forms of malicious software to harm all the exposed targets, so it is created a significant gap with very serious consequences that existing approaches cannot cover, that will be discussed in the next chapter.

```
msf > use exploit/multi/browser/java_jre17_exec
msf exploit(java_jre17_exec) > show targets

Exploit targets:

  Id  Name
  --  ---
  0   Generic (Java Payload)
  1   Windows Universal
  2   Linux x86

msf exploit(java_jre17_exec) > set target 0
target => 0
msf exploit(java_jre17_exec) > set payload java/meterpreter/reverse_tcp
payload => java/meterpreter/reverse_tcp
msf exploit(java_jre17_exec) > set lhost 192.168.2.7
lhost => 192.168.2.7
msf exploit(java_jre17_exec) > set srvhost 192.168.2.7
srvhost => 192.168.2.7
msf exploit(java_jre17_exec) > set uripath /
uripath => /
msf exploit(java_jre17_exec) > exploit
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.2.7:4444
[*] Using URL: http://192.168.2.7:8080/
[*] Server started.
msf exploit(java_jre17_exec) > |
```

Figure 1. 1 A typical exploit

1.1 Aim of Thesis

The aim of thesis is to update the assessment, which made in accordance with the developments in Information and Communication Technology, of new species of malicious software and test different networking topologies that exist today. The problem, in which the thesis is based, is to design and deploy a peer-to-peer network and with the use of epidemiological models to collect significant data results in order to optimize the performance of the PROMIS algorithm under a malware epidemic and in what degree it manages to protect its own members.

1.2 Novel Features of Thesis

The novel aspects of this thesis are:

An alternative way for dealing with malware. The Distributed Containment Algorithm PROMIS uses the peer-to-peer-networks in order to face malicious software. The aim of PROMIS is to highlight the positive contribution that can be made to increase the security of information systems, instead of the common belief which supports that they peer-to-peer networks degrade it. More specifically:

- In the detection of an outbreak of malicious software that is on the rise.
- In the warning of the other systems which belong to peer-to-peer network.
- In the automatic increase of the security level of the system in which performed during the peak of the epidemic and the restoration of the system in normal mode by reducing the level of security when this epidemic fallen into recession.

It is true that peer-to-peer networks is the key factor in spreading malware due to the fact that these ones is the largest contributor of network traffic on the Internet and are considered to be a huge security problem. Apart from their extensive use of the conduction of several illegal activities such as the file sharing that are protected by copyright legislation, they are also constitute the main gateway of malware in many cases. Therefore, peer-to-peer networks have been recorded as a threat against the security of information systems.

1.3 Outline of Thesis

In this section we give a brief insights on this research work by showing the organization of the other Chapters. In detail, we have: Chapter 2 show us what is malware, basic types of malware, the propagation techniques and the evolution of worms. In Chapter 3 we talk about networks, basic topology graphs, epidemiology and basic epidemiology models are described. In Chapter 4 there is the state of the art section while Chapter 5 problem statement and the research methodology takes place. In Chapter 6 we have the simulation data collected from our experiments and significant findings in order to optimize the operation of the algorithm. Chapter 7 contains the overall conclusion and possible future work in order the thesis be expanded.

CHAPTER 2

PRELIMINARY CONCEPTS - BACKGROUND

In this chapter we define what malware is and some types of malware, next we present the factors that propagate malware and techniques for malware containment.

2.1 Introduction

The term malware includes all the applications that have been designed, to deliberately incorporate undesired and non-obvious to the user functionality [29, 30, 31, 32]. In many books, there is a tendency the whole malware generally to be described by the term virus. In fact there are several distinct types of malicious software with different characteristics. Then we study and analyze the factors that accelerate or limit the spread of malicious software. The analysis of the determining factors for the development of some form of malicious code in an epidemic, performed with the use of basic epidemiological principles. Furthermore, various research efforts for modeling the propagation of malware will be examined. Finally are listed the major original systems and techniques for limiting the spread of malicious software.

2.2 Basic types of malware

Malware can be classified as a malicious code placed on a user's computer, thereby giving the attacker important privileges with respect to the control of the computer system. This system ceases to perform user commands and is obedient to the commands which accepts from the attacker. A definition for the malicious code is the following: *“a set of commands to run on a system or a computer and force it to perform what the attacker wants, which is none other than the one that created or simply uses the malicious code”*. Below is attempted to classify malicious software, quite brief and general, as most malicious software incorporate features and functions of many different categories. The basic malware types are showed on the table below:

Adware	Cyberweapon	Octopus	Rogue security software	Trapdoors
Backdoors	Dialer	Pharming	Rogueware	Trojan Horses
Blended threat	Flip button	Polymorphic code	Rootkit	Typhoid adware
Bots	Form grabbing	Polymorphic engine	Scareware	Virus
Chargeware	Hover ad	Quine	Spy-phishing	Watering Hole
Code injection	Logic bomb	Rabbit	Spyware	Worm
Computer worm	Malvertisement	Ransomware	Stealware	Zero-day virus
Crimeware	Malvertising	Riskware	Targeted threat	Zip bomb

Table 2. 1 Types of Malware

Quine

It is defined as a program which, when executed with or without input returns a code and if compiled (it may be a script as well), has resulted in a similar executable file. Usually, when referring to those, it is deemed always that it returns the source code. However, there are Quines which develop or return codes in a number of languages.

Virus

The Virus was identified with the self-replicating program. But today it has acquired a definition closer to the non-digital namesake, seeking production copies through files.

Rabbit

Rabbit defines two types of programs with that name. One definition describes a unique program that is transferred between network nodes without creating copies, so

we are not concerned. The other definition describes a program which is played in memory, buffering it.

Worm

The Worm is the network version of the virus. The aim is to be spread throughout the target population. A key feature is the self-propagation, namely that the spread does not require user intervention. Of course, there are worms which require partial user intervention (mass-mailers). A worm may have characteristics of virus, making the separation generally a difficult task.

Octopus

Octopus is referred to an evolution of the Worm. The program is dispersed in a series of nodes. So the parts of the code usually communicate with each other to implement a function. The goal remains the same as that of the Worm. Generally, it worth saying that the distinction is not clear and in some cases there is the challenge of authors in source codes categorized.

As it was mentioned, there is an abundance of open problems [76], of which we will focus on the propagation models of these issues. To achieve this objective satisfactorily, we should recognize the mechanisms involved and the environment in which we work. We will be referred to the structure and to the enforcement issues that exist for worms and virii.

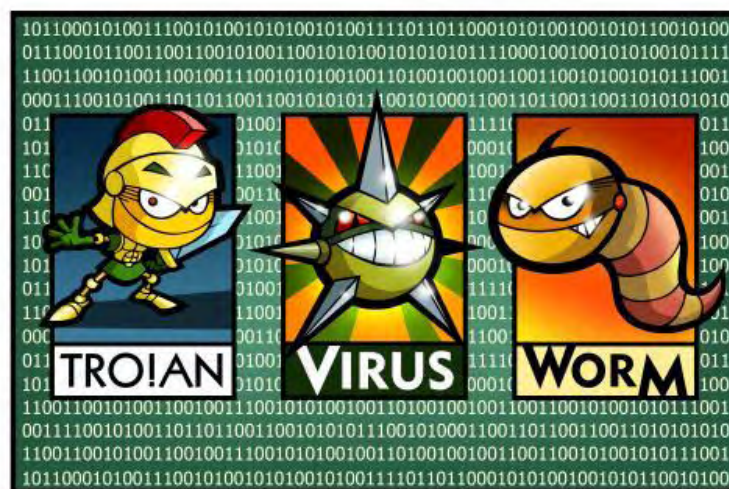


Figure 2. 1 Types of Malware

2.2.1 Virus

A virus when is executed, it tries to achieve the following [77, 78, 79]:

- To find a new compatible host
- To modify the appropriate host that when he is executed to perform another version of the virus.

Another common objective is not to be detected its presence on the system. Therefore, the general structure of a virus is as follows:

Search routine: Looking in the file system (real or virtual) for candidate files for infection.

Coping mechanism: An algorithm that essentially infects files (copies the virus code in the file for infection). Not only it decides where the entry point will be (point call of the virus code), but also the functions of the virus in the new program, modifies appropriately the headers etc.

Powered exploit: A virus usually needs to acquire rights, not only to list files, but also to be able to make the appropriate system calls.

Anti-detection routines: routines that make the things difficult for the user or anti-virus programs (anti-virus technics and algorithms) to detect the presence of virus. This routine can usually be the most complex in the program and together with the exploit code that change substantially from virus to virus.

Return routines: if the virus does not destroy the original file - victim must restore the necessary registers (eax, ds, es, ss, esp), after having made the appropriate corrections / reloads in various structures - such as DTA (Disk Transfer Area). Essentially this is a task switching function between virus and host.

Categories, regardless of the architecture of the executable file, (com, exe, elf, pe32, pe32+, etc.) the virii are divided into different categories depending on the type of their infector. By environment (file):

- *Boot Sector Infectors:* they occupy the boot sector of the disk and performed with the system startup.
- *File Infectors*

- *Multi-partite*: they essentially compose a mix between categories e.g. they infect both files and the boot sector.

The separation can be done at the level of relationship with the memory:

- *Memory Resident*: they remain in the memory even after the execution of the infected file. They cannot attack just loaded into memory but hide and infect programs that the user may then open.
- *Non-Resident*: they stay loaded in memory and the file that contains them. Once that is finished the execution of the infected file and the operating system closes the file freeing the memory occupied, then automatically released and the virus from the memory.

Regarding the file infectors, they are further classified into:

Overwriting: they copy their code on the code of the victim-host completely destroying the original file. When the user is going to execute the host, the program will naturally have different behavior than that which was before the infection.

Companion: The name was given because of the logic of maintaining of two files of host and virus, companionship one another. That is the usual tactic, renaming and probably hiding of the original file-host, while the virus takes the role of this. So instead of carrying the host, we execute the virus, which returns control to the host. Certainly companion can be described and the case where part of the code of the host or virus (usually the virus) are kept in a separate file, so we have a mix Companion with Parasitic.

Parasitic: They are attached on the code of the victim, but to destroy it, taking care to maintain the behavior (computes the same function). The virus can be placed at the beginning, at the end (appending) or in the intermediate file and regardless of the position of attachment takes control at some point, perform the necessary functions for it and then passes it to the host program . For performance reasons the virus code usually is placed at the end of the file, so we do not have to rewrite the entire file back to disk, while for convenience localization in intermediate file. It is virtually a non-

trivial approach. Now depending on the type of file that attack (com, exe, elf, pe32, pe32+) are called respectively: EXE infectors, COM infectors etc.

2.2.2 Worm

The worm is more in the structure. It can be consisted of one or more files or in very rare cases nowadays can be divided between nodes-computers. A worm should support the following procedures in order to achieve the main feature, the self-propagation.

Therefore is consisted of the following functions:

- *Relay Function*

It undertakes to transfer the code and any other information in the given node and to perform / to witness the start-up of the offspring of the worm.

- *Scan Function*

It undertakes the detection of potential vulnerable nodes and the evaluation perhaps of these nodes.

- *Objective Function Selection*

It usually is a subpart of Function Scan or it is absent. Essentially it returns a subset of selected nodes to become normal scan or the decision is what politics scan would be followed.

- *Exploit List*

It is a set of functions that guarantee the rights of both receiving for transmitting the code, but to ensure the stay of the worm and the proper operation of this at a node.

- *Payload*

It reflects the injected code to achieve various objectives such damage, data mining, etc.

ASLR /dynamicbase Table			
Base	Name	DLLCharacteristics	Enabled?
75870000	imagehlp.dll	0x0140	ASLR Aware (/dynamicbase)
73a90000	wdmaud.drv	0x0140	ASLR Aware (/dynamicbase)
02f70000	BlazeDVDCtrl.dll	0x0000	
03850000	EqualizerProcess.dll	0x0001	
74f00000	dhcpcsvc6.DLL	0x0140	ASLR Aware (/dynamicbase)
70ae0000	oledlg.dll	0x0140	ASLR Aware (/dynamicbase)
02f10000	RecorderCtrl.dll	0x0000	
70730000	msg711.acm	0x0140	ASLR Aware (/dynamicbase)
76150000	iertutil.dll	0x0140	ASLR Aware (/dynamicbase)
70ad0000	inaadp32.acm	0x0140	ASLR Aware (/dynamicbase)
02a80000	PowerManagementCtrl.dll	0x0000	
74f30000	WINNSI.DLL	0x0140	ASLR Aware (/dynamicbase)
75720000	ole32.dll	0x0140	ASLR Aware (/dynamicbase)
77060000	USER32.dll	0x0140	ASLR Aware (/dynamicbase)
00400000	BlazeDVD.exe	0x0000	
73a70000	midimap.dll	0x0140	ASLR Aware (/dynamicbase)
740f0000	OLEACC.dll	0x0140	ASLR Aware (/dynamicbase)
76390000	SHELL32.dll	0x0140	ASLR Aware (/dynamicbase)
751b0000	MSASN1.dll	0x0140	ASLR Aware (/dynamicbase)
76280000	CLBCatQ.DLL	0x0140	ASLR Aware (/dynamicbase)
70720000	msgsm32.acm	0x0140	ASLR Aware (/dynamicbase)
75130000	iphlpapi.dll	0x0140	ASLR Aware (/dynamicbase)
74770000	UxTheme.dll	0x0140	ASLR Aware (/dynamicbase)
74710000	MMDevAPI.DLL	0x0140	ASLR Aware (/dynamicbase)
6c460000	MMVCore.DLL	0x0140	ASLR Aware (/dynamicbase)
74130000	winmm.dll	0x0140	ASLR Aware (/dynamicbase)
761a0000	kernel32.dll	0x0140	ASLR Aware (/dynamicbase)
706e0000	asycfilt.dll	0x0140	ASLR Aware (/dynamicbase)
76ea0000	ntdll.dll	0x0140	ASLR Aware (/dynamicbase)
02c90000	ProfileStore.DLL	0x0000	
758a0000	WININET.dll	0x0140	ASLR Aware (/dynamicbase)
6c300000	WMSPDMOE.DLL	0x0140	ASLR Aware (/dynamicbase)
70650000	msadp32.acm	0x0140	ASLR Aware (/dynamicbase)
74a10000	version.dll	0x0140	ASLR Aware (/dynamicbase)
751d0000	DNSAPI.dll	0x0140	ASLR Aware (/dynamicbase)
75680000	PSAPI.DLL	0x0140	ASLR Aware (/dynamicbase)
6c890000	WMADMOE.DLL	0x0140	ASLR Aware (/dynamicbase)
02c40000	AudioProcess.dll	0x0000	
76000000	comdlg32.dll	0x0140	ASLR Aware (/dynamicbase)
71920000	winsock.drv	0x0140	ASLR Aware (/dynamicbase)
74f40000	dhcpcsvc.DLL	0x0140	ASLR Aware (/dynamicbase)
004a7000	USERENV.dll	0x0140	ASLR Aware (/dynamicbase)
004a7000	Diablo.dll	0x0000	
004a7000	VideoWindow.dll	0x0000	
004a7000	MSCTF.dll	0x0140	ASLR Aware (/dynamicbase)
004a7000	skinscrollbar.dll	0x0000	
004a7000	GDI32.dll	0x0140	ASLR Aware (/dynamicbase)
004a7000	msdmo.dll	0x0140	ASLR Aware (/dynamicbase)
004a7000	Configuration.dll	0x0000	
75a40000	OLEAUT32.dll	0x0140	ASLR Aware (/dynamicbase)
74510000	RVRT.dll	0x0140	ASLR Aware (/dynamicbase)
03850000	RHACtrl.dll	0x0000	
61600000	EPG.dll	0x0000	
73650000	audioeng.dll	0x0140	ASLR Aware (/dynamicbase)
75010000	CRYPT32.dll	0x0140	ASLR Aware (/dynamicbase)

Figure 2. 2 A typical exploit list

The categorization of a worm is usually based on two factors:

- *The policy scanning which is followed.*

- Random / Uniform Scanning Worm

- Hit-List Worm

- ...

- *The average spread that it uses:*

- Mail Worm

- P2P Worm

- ...

The classification of viruses and worms is always included the operating system for which it is made. Here are names of various worms / virii (as we notice) that include the categories in which each one belongs. About naming, we refer to [77] and the relevant articles of each relevant company.

1. Win32.worm.mytoB.C
2. W32.Stuxnet
3. W32/Eliza.virus

2.3 Characteristics and factors that propagate malware

The construction and the development of effective malware is not obviously an easy and trivial process. The effectiveness of the infection of a malware depends on many factors which determine and the degree of its spread. The malware factors are summarized in the following three factors:

- The main gateway
- The selection of new targets
- The infection rate

2.3.1 Main gateway

The worms in order to be able to spread from one system to another, seek to exploit a security hole which usually due to a programming error. The choice of the appropriate security gap is crucial for the further propagation of a worm. The factors which affect the efficiency of the security gap is how widespread is or otherwise:

- Demography of this security gap
- The time which is known this vulnerability
- How easily exploitable is this security gap by malicious entities.

2.3.2 Selection methods of target discovery

A worm, in order to propagate effectively, need to have an efficient algorithm for the selection method of new targets. Researchers have shown the importance of the

propagation strategy for each worm, as different strategies lead to large differences in the final number of infected systems and required time to achieve it.

As we begin to understand one of the most important parts of a worm is the process of finding the next target. To do this, the worm must search the sub-agents to generate a list of candidates and likely to prioritize them, evaluate and ultimately select a subset in each round.

The discovery is implemented by scanning e.g. only ping through specific doors and maybe send some packages to verify if it is vulnerable to used exploits. For certain types of scanning is necessary and determining if the target is already infected (e.g. permutation scan). The most likely propagation strategies that have been described are the following.

2.3.2.1 Random Scanning

Random Scanning is the simplest method for launching attacks [10, 11]. The worm simply selects a random IP address and if there is a device which owns the specific address, then it attacks to the device. This can be implemented very easily by reducing the overall malicious code complexity. Main drawback of random scanning is the lack of truly random numbers generators by most modern personal computers, although some hardware manufacturers recently experienced some related products. In addition, another problem of random scanning is that it produces and imports on the Internet huge amounts of data. A positive point for researchers who study the spread of malware is that the random detection can easily be described using known epidemiological models

2.3.2.2 Localized Scanning

In the local scanning [11], instead of randomly selected IP addresses, these are selected in such a way as to be on the same subnet as that the computer from where the attack is attempted [10]. Usually computers which are on the same subnets are belonging to a large organization or business and showing homogeneity both in hardware and in software that they use.

2.3.2.3 Hit List Scanning

In this case, before the attack is preceded with the creation of a list where future targets are indicated according to several factors that have been compiled by various sources, such as meta-servers, public surveys, stealth scans or distributed scans.

The creation of this list aims to the selection of appropriate goals so as the infection propagates in the early stages of its spread with the most rapid rate so as the restriction to become difficult.

Pre Generated Target List (Hit List)

The aim is to speed up the propagation through a pre generated target list of addresses.

Complete Hit List

We guarantee the hitting probability unit. The worms are actually acquiring characterization as flash worms, because of their propagation speed which their only drawback is the limited bandwidth.

Incomplete Hit List

We try to speed up the propagation to guarantee an initial infected population, without paying the full cost of hosting the entire list targets. We rely on the observation that the first stage of the epidemic is more important than the propagation speed.

Generated Target List

However we have the ability to produce all of its targets, by data mining supplied by the nodes and the environment which infects. For example, the p2p worm can be easily added in the list which discovers with peers and to propagate.

2.3.2.4 Topological scanning

In the topological scanning of new targets, where the malware infects a system, then searches for user elements such as e-mails or entries in files, which can reveal the existence of other possible targets.

2.3.2.5 Permutation scanning

The permutation scanning is an effective technique for the further propagation of a worm. It is based on the partition of all the IP addresses into smaller sections. Whenever a worm manage to infect a new target, automatically assigns to the new worm that was created in the infected computer, to scan the half of IP addresses of those which originally took. In this way both more targets managed to hit by the malware more its propagation is accelerated after distributed in a larger scale, the process of finding new targets.

2.3.2.6 Multi-vector worms

Multi-vector worms use existing communication standards instead of creating new ones to locate and infect their victims, such as by active probing, bulk e-mailing itself as an attachment, copying itself across open network shares, by adding exploit code to Web pages on compromised servers and by scanning for backdoors left by Code-Red II.

2.3.2.7 Theoretical worms

The combination of the hit list scanning and the permutation scanning leads to a new type of a worm, capable of infecting all susceptible servers in a few minutes. Staniford *et al.* [10] named the worm with these properties Warhol.

2.3.2.8 Tarpits and Honeypots

It is not quite as harmless as scanning, nor as a defenseless network. A Honeypot is like saying the word a node showed up as a potential target (perhaps with increased profit), but basically a trap which detects malware that spread in the network. On the other hand, the Tarpit has additionally goal the delay of propagation. A common way (LaBrea) is reset the TCP window at startup of data sending and rejection of requests for close connections. A workaround is using custom implementation or configuration of the window or the use of UDP protocol.

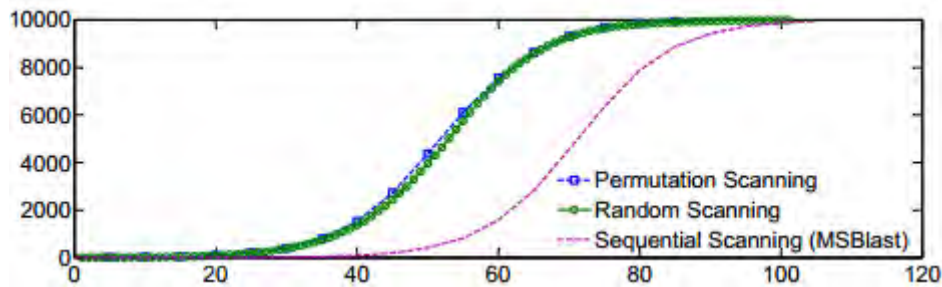


Figure 2. 3 Comparison between Use of Permutation, Random and Sequential Scanning

2.3.3 Infection Rate

The concept of the infection rate of a malware is associated with its destructiveness and its hazard. It is noted at this point that most of the worms that failed to be developed into epidemics, they didn't bring any devastating charge. The most devastating is a malware, the most directly can be understood as the symptoms which cause are quickly perceived. Therefore, a worm that has been constructed aiming for example in monitoring the victims and causing no damage, hardly come to perception of the user. The way of discovery new targets determines the mode of transmission of malware to have greater infectivity. Thus, a worm that uses a strategy of propagating the hit list scanning or the permutation scanning, the best option is to start destroying their hosts when they have exhausted all available targets.

A worm that uses localized scanning should first attempt to infect all neighboring systems before it starts destroying its host. Similarly, in the topological scanning after exhausting all the information that can be gathered from logs and other data that are stored on the computer that controlled by the worm and make similar attempts to attack these targets, will be the malware safely to destroy its host. Finally, for random scanning could be chosen the solution of the destruction of the host just when the worm repeats a predetermined number of attacks without being able to detect a susceptible uninfected target.

2.4 Techniques for Malware Containment

The types and the techniques for malicious software containment are designed to protect information systems and networks from malware and generally malicious actions. So the main concern is the security of systems and networks. A typical and indicative definition of the term “*Information Systems and Networks Security*” is the prevention of the attacks that aimed at unauthorized exploitation of computers and networks and their resources. These basic principles that must be met before the development of such software are shown below:

Confidentiality

The principle of confidentiality protects sensitive information from non- unauthorized access or its interception.

This information should be apparently only among the legitimate users of a communication and not in those who possibly eavesdrop communication channel.

Integrity

The principle of integrity ensures that the information or software is complete, correct and authentic, in other words that there hasn't been a change in an unauthorized manner. We want to ensure that there are appropriate mechanisms in the right places, which protect us from accidental or malicious modification of the original information.

Availability

The principle of availability ensures that the information or services are operational and accessible when required by someone who is authorized to access them. This principle relates the concept of trust. This concept means how a user can trust a computer system and is appeased that the system does what it claims and not some other undesirable action.

2.4.1 Categories of containment for malware techniques

- IDS - Intrusion Detection Systems
- IPS - Intrusion Prevention Systems
- Firewall
- Protection programs against malicious software – Antivirus
- Antispyware and Anti-adware Programs

In the following section we present some types of these containment systems.

- *IDS - Intrusion Detection Systems*

Intrusion detection systems are designed to identify any attack and their presence to the user. The most intrusion detection systems operate by using *Signatures*, i.e. detect illegal sequences of actions, as do the most antivirus programs. Popular IDS are: ossec, suricata, snort [85, 86, 87].

- *IPS - Intrusion Prevention Systems*

The intrusion prevention systems are essentially the next stage of intrusion detection systems and often these two systems coexist. The IDS is a system that will identify the attack and the IPS will usually deal with it automatically. Popular IPS are: McAfee Network Security Platform, Sourcefire IPS [88, 89].

- *Firewall*

It is a mechanism that controls the access to and from the network. It acts as an intermediate key element of which passes all the network traffic from and to the external network.

- *Antivirus*

They are the most popular way of dealing with malicious code. The detection of malware is achieved by the *signatures*.

- *Antispyware and Anti-adware Programs*

They are programs that detect spyware or adware malicious applications. It consists of a search engine, which is based on *signatures* through which can recognize if installed any adware or spyware and remove it.

2.5 The Evolution of worms

- *Polymorphism*

Polymorphism is referred to that process whereby the malicious code change his appearance in order to avoid identification without changing the function for which it was made. The term polymorphism demonstrates that the code can take several forms, but all have the same function.

- *Metamorfism*

Metamorfism leads the process of change of the virus or the worm one step further as we also change the operation of the code as it propagates. This is usually done in an intelligent way so as to ensure that the worm or the virus will avoid the detection without losing at the same time its ability and activity.

- *Worms of multiple operating systems*

Today, most worms attack in one operating system, which requires that the administrators of these systems should install the necessary applications to create a pretty good line defense line. In the near future, what we called superworms will operate more than one type of operating system, including Windows, Linux, Solaris, BSD and more, all within to a head. In May 2001, the Sadmind / IIS worm made its appearance, infecting the operating SUN Solaris and Microsoft Windows.

- *Worms exploiting several vulnerabilities*

New worms operate more than one vulnerabilities in a computing system, which then use them for their own propagation. With more vulnerabilities to exploit, these worms will be propagated more effectively and faster. So far the representative of this category is the worm Nimda which could be propagated in a system with many ways.

- *Zero-Day worms*

Worms that exploit vulnerabilities which have just been released and have become known.

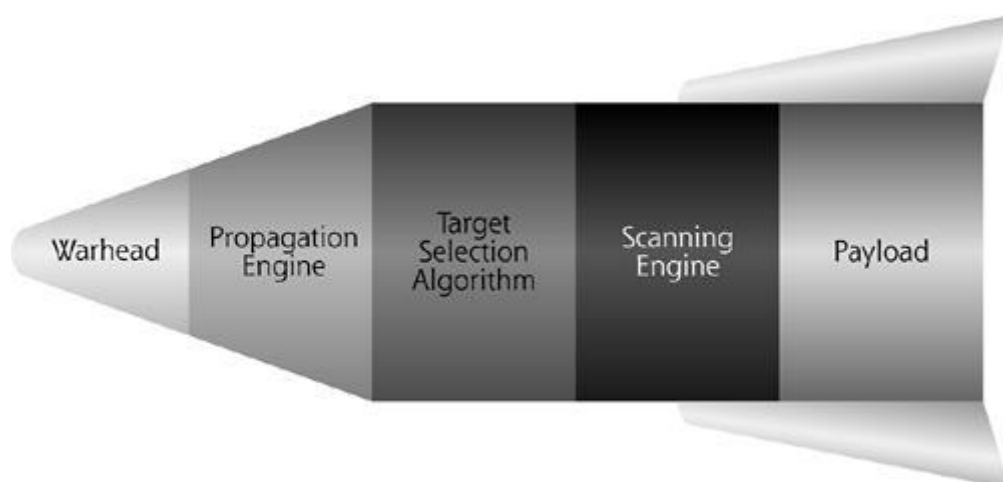


Figure 2. 4 The skeleton of a worm

CHAPTER 3

NETWORKS, EPIDEMIOLOGY AND EPIDEMIOLOGICAL MODELS

In this chapter we mention briefly some principal fundamentals of networks, describe the epidemiology and the epidemiological models and how these are connected together.

Introduction

Networks and epidemiology of directly transmissible infected diseases are basically connected. The foundations of epidemiology and early epidemiological models were based on widely random mixed populations but in practice each individual has a certain number of contacts in which he may pass the infection. The sum of all these contacts creates a "Mixed network". The knowledge of the network structure allows the models to calculate the dynamics of the epidemic in proportion to the population of the level of individual behavior infections. For this reason, characteristics of mixed networks, and how they deviate from the typical randomly mixed, have led to significant concerns that may increase the understanding and the prediction of epidemiological patterns and proxy measures. Various forms of computer networks created on a computer have been studied in context of disease transmission. Each of these networks can be conveniently determined by how individuals are distributed in space (geographically and socially) and how the contacts are made, thus simplifying and making clear the various complex processes which are involved in the creation of a network in real populations . We see here a set of the most known types of networks and their impact on the spread of the epidemic.

3.1 Networks

3.1.1 Random Networks

In random networks, the position of individuals in space is irrelevant and the contacts which are randomly generated [33].

In the most detailed convenient version of a random network, each person has a fixed number of contacts through which the infection can be spread. The random network therefore is characterized by a lack of teamwork and homogeneity individual-level network properties.

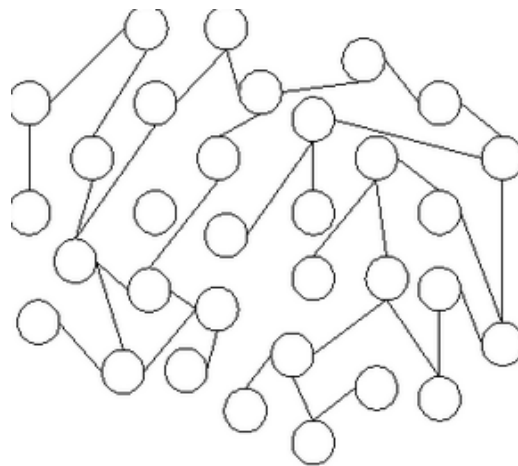


Figure 3. 1 Presentation of a Random Network

Law *et al.* [71] showed that the dynamics of diseases in random networks can be studied as a single branching process, from which we conclude that the early growth of the disease and the final epidemic size are reduced when they would be compared with the randomly-mixed model.

Growth rate in random network = $t(n-2) - g$

Growth rate in random development = $b - g = t^n - g$

Where t is the rate of diffusion through a contact, n is the number of contacts within the network, and t^n is the effective number of contacts per unit time in a random mixed model. The reduction in growth rate is created for two reasons: first, each infected person is infected by one of his contacts, reducing the number of susceptible to $n-1$. Secondly, as a transfected person begins to infect the vulnerable contacts,

reduces the local environment, even if the preponderance of the population is low, and therefore limits the rate of spread of the disease.

These two processes are common in all epidemics of networks (although the intensity of the impacts may vary). Detailed results derived from the study of simple networks allow us to develop an intuitive understanding of the effects of more complex social networks in spreading of a disease.

An alternative construction of a random network is the interconnection of two nodes with probability p . This leads us to a network with an approximately Poisson degree Distribution and an average number of contacts per *node* $\bar{n} = p(N-1)$, where N is the total number of nodes. In such a network, the growth rate is further reduced.

Growth rate in a random Poisson network $T = [(\bar{n} - 1) / (\bar{n} - 1)] - g$.

Barbour *et al.* [34] however, as that active conversion on a smaller scale is given, the epidemic proportions in this network are similar to a SIR epidemic to a random mixed population

3.1.2 Spatial Networks

Spatial networks are the most flexible networks. The nodes (entities) are placed in a given area and two nodes are associated with a probability that depends on their separation and is defined of a core connection. Changing the distribution of entities or the core is possible to create a wide variety of networks, such as strong aggregate grids, universally connected random networks, etc. [36, 37]. These networks exhibit a high degree of heterogeneity logic, wherein degree distribution is usually Poisson. Additionally, when local connections are preferred, then a wave spread of infection is observed and it features mesh models.

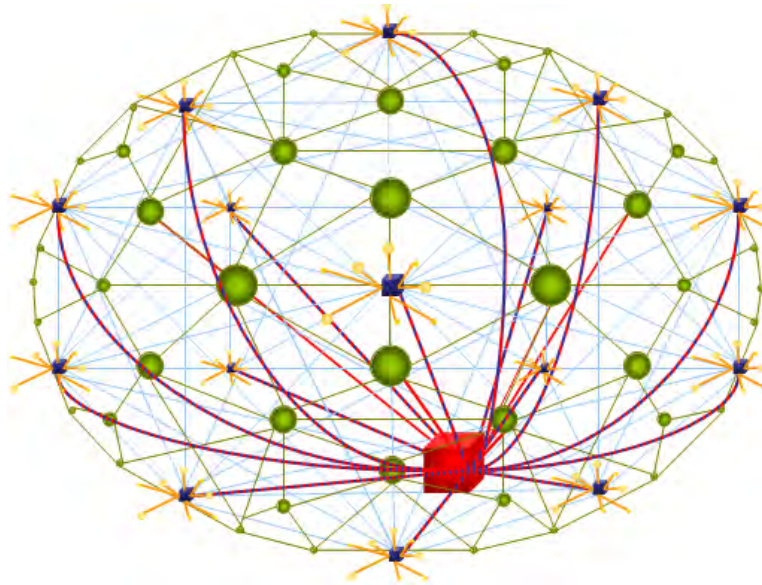


Figure 3. 2 Representation of a Spatial Network

3.1.3 Scale-Free Networks

One of the most important criteria of a network is the distribution degree of nodes. In many observed networks this is far from homogeneity. It is usually the case where many entities have a small number of neighbors while few have significantly more connections [38, 39]. Networks of small world, random networks and mesh models show small variations in the sizes of their neighbors, while spatial networks have usual distribution degree that follows approximately the Poisson distribution. However, as the entities with many connections (super-spreaders) are likely to be disproportionately important to the spread of an infection, being incorporate such entities in the networks is necessary if we want to grasp the complexities of spreading of an infection. The scale-free networks provide the means so as to achieve such extreme levels of heterogeneity.

The scale-Free Networks can be constructed dynamically with the addition of new entities in a network one at a time with a mechanism connection that simulates biological mechanisms. Each new node (entity) which is added to the population, prefers to be associated with nodes that already have a large number of connections, which represent the people who want to become friends with those who are most popular. This leads to the conclusion that the number of the contacts per entity follows a power law distribution.

This property was originally observed for the connections of World Wide Web [39] but has also been reported in networks of sexual contacts of people [40] and also at the graph resulting from the collaboration of actors [38].

The extreme heterogeneity of the numbers of contacts which is appeared in a Scale-Free network is a characteristic of populations that raises the interest of epidemiologists since long time. The nodes which act as super-spreaders and the core groups, play a fundamental role in propagation and maintenance of an infection. It is important to understand that when an entity has many contacts, this has two important effects: this entity is at a higher risk of being infected and when it is just infected, then it can spread the infection to many others. If we are now talk about groups of people, then the target vaccination in these networks is quite effective. Because of the dominant role of super-spreaders and with the vaccination of some of them we can manage to prevent an epidemic.

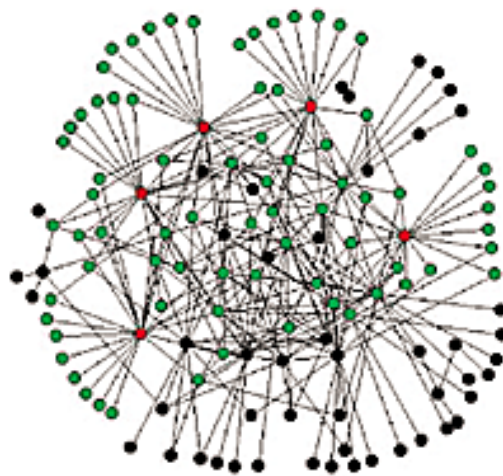


Figure 3. 3 Representation of a Scale-Free Network

3.1.4 Exponential random graph models

Frank et al., [41] described these models (also known as ' p * models') as to provide a method to create networks with a given set of properties. If we are concerned only with if the average value is correct, then we can either to add a fixed number of edges in a set of nodes or to add an edge between two nodes with a constant probability, p ,

independent of all other edges. For example, if the average degree of a population N is \bar{n} , then with probability $p = \bar{n} / (N-1)$ is produced such a network with the desired number of connections. However, these networks always show little clustering (as affiliated entities are not likely to be shared with a neighbor), short paths and binomial degree. These networks have a simple property that the probability of connection between two nodes is independent of the connection between any two other nodes. This allows the probability of any nodes to be connected and it can be estimated dependency graph which has certain properties.

3.2 Epidemiology

Introduction in Epidimology - Biology and Computers

The term “epidemic” has been defined as an outbreak of a contagious disease which is propagated rapidly and over a wide range with respect to the area of contamination. In a similar way, the epidemiology in computer science can be defined as a computer virus or a worm that is propagated rapidly and widely infecting computer systems in a region or a population simultaneously.

The epidemiology in computer science, was originally studied by Kephart *et al.* [69] and who described the manner with which computer worms / viruses are propagated. It was found that there are several analogies in the way of an epidemic spread if someone studies from the perspective of biology or the perspective of computer science. Before we refer the basic epidemiological models, we will talk about epidemiology, how it started and which are the main sources of which influenced what we study and the use of epidemiological models in computer science. A key factor as we will see below is biology and models that have been developed for this field.

3.2.1 Epidemiology in Biology

The epidemiology in general tries to explain what happens in humankind and especially in emerging diseases. Generally we can say that describes a scientific methodology in biology so as to study the nature, the prevalence and the causes that

cause a disease. This science provides a method for the understanding but also the response to a disease as it spreads in a population. The epidemiology uses mathematical models to quantify, to characterize and to predict the propagation and the impact of a disease. Demographic analysis that is usually applied, it is used to determine the relationship between disease and population. The role of those who involved with the epidemiology is to destroy or damage this relationship so as to prevent the contamination of the population. The main goal of epidemiology is to prevent the spread of the disease and to prevent possible future recurrence.

3.2.2 Epidemiology and Computers

The digital epidemiology applies the biological epidemiology seen above in cyberspace, and more generally of what we call today computer science. Network and system administrators and the researchers realized that the security of a system depends on the safety of the whole population which may include the subnet, the university or the company network, or even the entire Internet.

Techniques of Organic epidemiology offer methods to understand and deal with the security issues that threaten the health of this population.

3.3 Epidemiological Models

3.3.1 Introduction in epidemiological models

An epidemiological model is a pretty good tool to understand the spread of an infection by relating the propagation process with the properties which a host can have. However, epidemiological models are not so easy to implement and to be sure about the results because:

- Results depend on certain claims which are rarely accurate.
- They depend on the values of some parameters such as, for example, the number of the population and the contacts between them, which they are just allegations.

- The epidemic levels, which arise, are strong enough and usually easily observed because of the values that we give in several parameters.

An epidemic algorithm is dealing with a population that can be represented by a set of individual entities which interact with certain rules. Those rules have a significant role in the dissemination of information. Those entities (hosts) should have every time one of the following three states:

1. Susceptible:

The user has no idea about the specific information (virus, worm, malware), but is able to accept, in other words to be infected.

2. Infective:

The user knows now about this information that is infected, and is able to infect others by spreading this information, into other susceptible hosts and who have not yet become victims.

3. Recovered:

The user is aware of this information, and thus may not infect others or be re-infected later.

In general, the epidemiologic models can be split into two major categories: stochastic and deterministic.

The stochastic models are usually used for a small or isolated population because they focus their attention on each user. These models require a lot of work to produce a result which could confirm the predictions that had been made. Also these models are difficult to understand and they have complex mathematics.

On the other hand, the deterministic models are mainly used in large populations, and try to inform us about the average for this population, based on some initial conditions and situations. These models place the users into subclasses or better in situations.

For instance, SEIR model includes the following statements about the users: Susceptible, Exposed, Infected, Recovered. The deterministic models are widely used because they do not need huge amounts of data and they are not very complex.

The transition from one state to another, occurs with a rate, for example, the rate of the infection is a well-known factor which pushes users who are in a susceptible state to go over to an infected state. When an epidemic breaks out, just because users interact with each other, this causes changes in the situations they are, when time passes. This transition becomes with a rhythm.

In the beginning, each user and each entity can be considered to be in situation Susceptible (S), with the passage of time the number will be reduced and the number of other (Infected (I), Exposed (E), Recovered (R)) will be increased with some selected pace.

The epidemiologic models for networks that exist today are becoming more popular as the propagation of viruses / worms. A network can be represented by a graph, where each graph has some vertices which represent these entities that mentioned (hosts) and some edges which represent the connections and the interaction they have between them. Each node in the graph is in one of the following situations: Susceptible, Infected, Exposed, Recovered, Removed.

Each infected node can carry the infection to any neighboring node which is in a susceptible state.

There are several reasons which affect the spread of an infection:

- The number of infected nodes at the time.
- The infection rate.
- The number of vulnerable nodes.
- Whether the population has some vulnerabilities.
- Levels of immunity.
- Time where an infected node remains infected.
- The degree of connectivity and interaction between nodes.

3.3.2 Goals and limitations of epidemiological modeling

Below we will present and refer the purposes for which the use of epidemiological models, as well as the limitations are set. Main reasons for epidemiological modeling:

The developing model of the process we're going to describe, simplifies and explains the various assumptions, variables and parameters that we set each time. The behavior

of the accuracy of mathematical models that we use can be analyzed using mathematical methods and computer simulations. The modeling allows exploring the impact for various claims and formalities and provides some basic concepts such as limits, breeding numbers, etc. Furthermore, modeling is an experimental tool in order to check out some theories and evaluate quantitative speculations. Models with appropriate complexity may be constructed in order to answer some specific questions.

The modeling can be used to assess some basic parameters and the models provide structures for organizing, union and intersection of different pieces of information. Also, the models can be used to make comparisons of certain epidemics deferments' types in different timeslots and in different populations as well as to make a theoretical evaluation, comparison or improvement of various programs related to the discovery, prevention, treatment and control. The models can be used to evaluate the sensitivity of the results which obtained from changes in values of some parameters.

In addition, the modeling may propose the collection of some significant data, which in other cases they could be ignored and can contribute to the design and analysis of some research on the epidemiology (during the design stage, the modeling can help identify of some significant issues and questions that should be answered in order to have sufficient information for successful results). Moreover, the models can be used to identify new directions in this field, to make general forecasts and evaluations of the uncertainty of some predictions (ability to make predictions about the future impact of an epidemic). The reliability and robustness of the results of modeling could be determined using a range of values for the parameters in many different models.

Main restrictions of epidemiological models

An epidemiological model is not something real. It is one oversimplification of reality. The deterministic model does not reflect the role of luck and the probability of an epidemic spreading. The stochastic model introduces the concept of luck, but is usually difficult to analyze with respect to the deterministic model, because of complex mathematical relationships.

Regarding to the limitations encountered in terms of epidemiological modeling, we can mention the following:

As it was previously mentioned, the first and most important limitation is that all epidemiological models are simplifications of reality. For example, we often assume that a population is unchanged and homogeneous. This is a simplification for the model, but the deviation of the reality of this simplification differs depending on the infection and conditions which are prevailing. This deviation is rarely measured or could be controlled. All the models have as main characteristic of man, user, and the interactions he has.

People do not behave with a predetermined way and this affect the models that have developed. Because the propagation models are simplifications with usually unknown relationships with infections and epidemics, no one can ever be sure on the results, projections, comparisons, etc. Even when models become more complex in order to approach even more of an epidemic is still an abstract concept. He who developed a model should be pursued his judgment to be able to decide which factors are relevant and which are not when analyzing an infection or answering some basic questions.

The deterministic models are those that use differential equations (integral or functional) to describe the changes in relation to the time of the sizes of some populations. Having some initial conditions for a good deterministic model, the solutions as a function of time are unique. In stochastic models, there are chances in each time slot to go from one state to another.

The simulation of these models is done by calculating probabilities using random number generator and the results in each passage are different, so this method is called Monte Carlo simulation. The findings-final results are obtained from the average results from many simulations on computers. The simple deterministic models for epidemiologists have an exact threshold that determines whether an epidemic will occur or not. On the other hand, the stochastic models for epidemics, introduce some quantities such as probability of happening one epidemic or the time it will be eliminated. So the process, basic concepts, approaches but also the appropriate questions as to obtain the answers are relatively different for stochastic models.

Both deterministic and stochastic models have other limitations beyond the fact that they are simplifications of reality. The deterministic models do not receive the factor

of luck in spreading an infection. Sometimes the values of some parameters in deterministic models are set to be equal to the average of the observed values and some other information is ignored. A set of initial conditions results in a single solution in a deterministic model. So, there is not available information on the reliability of the results.

The stochastic models incorporate the concept of luck, but are usually very difficult to obtain analytical results for these models. Also the computational results are difficult since the Monte Carlo simulations require several passes on computers to detect patterns and to produce quantitative results.

3.4 Basic Epidemiological Models

3.4.1 Susceptible - Infectious (SI) Model

In the SI model, which is described as a classic simple epidemiological model, each host is either susceptible (vulnerable) to an infection, or has already been infected (infected). The only acceptable transition between states in this model is from vulnerable to infected as it is shown in the following figure, and particularly in Figure (a):

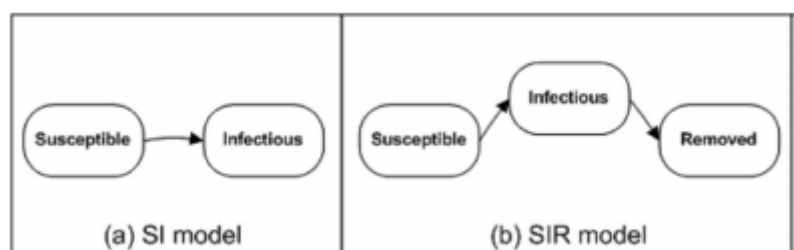


Figure 3. 4 Representation of the states of SI and SIR models

This means that an infected node is considered to remain infected forever. This model defines a set of parameters as are shown below:

$S(t)$: the number of vulnerable hosts at time t .

$I(t)$: the number of infected hosts at time t .

N : the size of population that is vulnerable.

β : the average infection rate (can be expressed as a function of the average rate exploration of a worm, r : a scanning worm which explores the entire field of IPv4 addresses randomly, leads to $\beta=rN/232$)

This model can be described by a set of two differential equations, as are shown below:

$$\begin{cases} \frac{dI(t)}{dt} = \beta I(t)S(t) \\ \frac{dS(t)}{dt} = -\beta I(t)S(t) \end{cases}$$

Because all the hosts in this model, as we mentioned are either susceptible or infected, it is easy to see that the growth of susceptible hosts are inversely proportional increase of infection, as it is shown in the following graph:

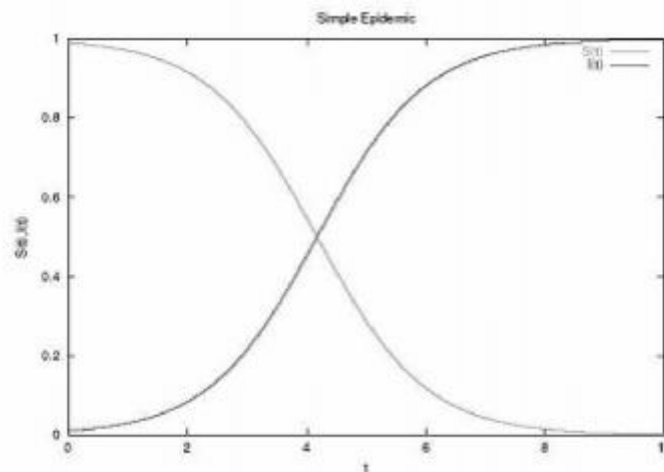


Figure 3. 5 Simple Epidemiological SI Model

The model assumes that the initial number of hosts that are susceptible, is considerably higher than the number of those who are infected. As a result, the initial infection rate is exponential. As the number of susceptible and infected hosts comes to an equilibrium, the infection rate begins to be decreased, but it does not stop until all the nodes that are vulnerable to become infected. This can be concluded from the

unreal claim that the only situations transition that we have is from susceptible to infected. Next, we present a more realistic model which introduces the fact that an infected host may be recovered or die.

3.4.2 Susceptible - Infected - Recovered (SIR) Model

The SIR model, also known as classic or general epidemiological model or Kermack McKendrick model [42] of its inventors, adds an additional condition called removed in simple epidemiological model. The condition removed, represents the hosts who have recovered from the infection and no longer can be re-infected, those who have entered quarantine and have been withdrawn from circulation, and those who have died of the infection. So there are two acceptable transitions in this model: the transition from susceptible to infected, proportionally with the SI model, and the transition from infected to removed condition as it is shown in the figure (b) above. Additionally to the parameters which we introduced in the previous model, here we have another two new parameters:

$R(t)$: is the number of removed hosts at time t .

c : average removal rate.

This model describes the propagation of a worm with a set of differential equations, which are the following:

$$\begin{cases} \frac{dI(t)}{dt} = \beta I(t)S(t) - \gamma I(t) \\ \frac{dS(t)}{dt} = -\beta I(t)S(t) \\ \frac{dR(t)}{dt} = \gamma I(t) \end{cases}$$

By inserting the relative removal rate, $p = \gamma / \beta$, the first equation in above system can be rewritten as: $dI(t) / dt = \beta [S(t) - p] I(t)$.

Because we believe that the population is finite and each host can once be infected, the epidemic will eventually stop and will 'die'. When this would happen, then all hosts in the population will be either vulnerable to infection or would have been removed.

Looking at the last equation, someone can notice an interesting property of the SIR model. Obviously $I(t) > 0$ and β is greater than or equal to zero. As a result, we have that: $dI(t) / dt > 0$ if and only if $S(t) > \rho$

Because the quantity $S(t)$ is a monotonically decreasing function (not added new susceptible hosts in the existing population at any time), if $S(0) \leq \rho$ then $S(t) \leq \rho$ for all $t > 0$ so that $dI(t) / dt \leq 0$ for all positive values of t . This means that there will be no epidemic unless the initial number of susceptible is greater than a critical value ρ .

3.4.3 Susceptible - Infected - Susceptible (SIS) Model

The key property of this model is that it can stop the spread before all hosts become infected. In this model, nodes that have been removed, they can be infected again.

This model is one of the simplest epidemiological models. It is consisted of two situations, the Susceptible (S) and the Infected (I). A node that is susceptible, may be contaminated by a neighboring node at a time step and then proceeds to state infected. At the same time of this step, the nodes that are infected, they will receive a treatment with a probability and they will become susceptible again. And because of that the nodes will move from one state to another and back continuously. This model does not take into account the properties of removal (death, protection or immunization). In the SIS model there are infection rates and recovery from which however the hosts become susceptible again.

This model is primarily used to study propagation of those worms where some nodes are 'off' for some time but they have not been cured of the infection, for example when an infected machine is closed for some time.

The SIS model can be described by the following differential equation:

$$\frac{d_i(t)}{dt} = \beta \bar{d}(1 - i(t))i(t) - \gamma i(t)$$

where:

β : is the infection rate

d : is the average degree of an infected node

γ : is the rate of recovery.

The recovery is proportional to the number of infected nodes and rate recovery.

The solution of the above equation is:

$$i(t) = \frac{(1 - \delta)i(0)}{i(0) + (1 - \delta - i(0))e^{-(\beta' - \gamma)t}}$$

This is describing the rate of infection.

Where is:

$$\beta' = \beta \bar{d}$$

And d is the cure rate.

Similar to the SI model, in case of a complete graph with n vertices, then are:

$$\bar{d} = (n - 1)$$

And the fraction of infected will have the following solution:

$$i(t) = \frac{(1 - \delta)i(0)}{i(0) + (1 - \delta - i(0))e^{-(\beta(n-1) - \gamma)t}}$$

(This is describing the rate of infection and recovery for a complete graph).

3.4.4 Susceptible - Infectious – Detected - Removed (SIDR) Model

In this model, we now have four states: susceptible, infected, detected (at this stage the worm has been detected but it is not active to infect), and removed.

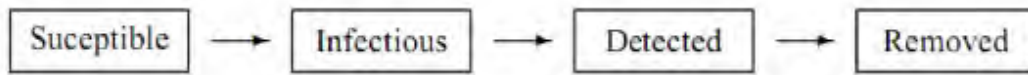


Figure 3. 6 Representation of states of the SIDR model

This model is used to study the “throttling” of a worm which an automatic mechanism is aiming to contain or reduce the spread of a worm and the information that it carries. The evolution of this model is consisted of two main phases: the first is in which the signature of a virus is appeared which leads a node to change the status of susceptible to infected with a rhythm. The second phase is the discovery (detection) of the virus. The nodes will be divided into two groups which are called “throttled” and “un-throttled”. If a node belongs to the category throttled and become infected, the infection will not go to other nodes and in a moment will change state and will become from infected, detected.

3.4.5 Susceptible-Infected-Removed-Susceptible Model (SIRS)

This model consists of three states: susceptible, infected, removed.

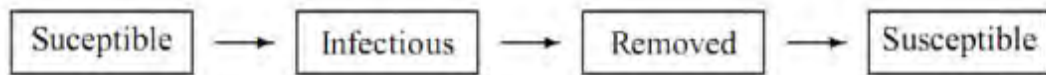


Figure 3. 7 The possible state transitions in SIRS model

Instantly after a node has been removed, it will remain in this situation for a while, and this period is called standby state. Immediately after this state, it will go to susceptible state.

CHAPTER 4

STATE OF THE ART

In this chapter we present the approaches and solutions with different concepts and architectures proposed by others researchers to front the spread of malicious software.

4.1 Introduction

The rapidly spreading malware is a distributed threat since the starting points of an attack are many and scattered. Therefore, the nature of this threat requires a more coordinated response to the problem, compared with the existing centralized and partially decentralized security architecture. These considerations led to the development of a new generation of systems, designed almost exclusively to the problem of the spread of malware and operate distributed, decentralized and in cooperative basis. These systems will be classified into three different categories to make it easier to understand the potential of their operation and the results they can achieve in limiting the spread of malicious software. The first category concerns collaborative containment systems based on peer network architectures. The second category includes centralized intrusion detection systems, but which have a wide network of sensors. Essentially this is an extension of traditional intrusion detection systems, which are used extensively in the past decades. The intensity of the problem of malware and especially the time reduction that required to infect all vulnerable targets, led to the modification, so they can address more effectively such threats. The way in which they expanded the action area in malicious software, includes the installation of hundreds of sensors that record the evolving malicious activity and inform a central server. The last category relates to various research efforts, which follow completely distinct and separate approaches, making their grouping in some general category unfeasible.

4.2 Systems based on Peer-To-Peer (P2P) Networks

Cai *et al.* [43] presented a Cooperative Containment System with dual targeting, since according to this, it can help both in limiting the spread of malicious software while dealing with Distributed Denial of Service Attacks (DDoS Attacks). The system is installed in edge networks and it is implemented in the network layer, and as sub infrastructure uses the existing Chord architecture.

The functions are not restricted to simple recording of the malicious activity, but it has also the ability to build automatically signatures for the worms which it finds. The communication between nodes is based on DHT (Distributed Hash Tables), which offer fast search. The detection of the new worms performed by the Rabin footprint algorithm which it captures the local and the global evidences about the observed malicious activity for all the members that participate in the system and then it creates new signatures. The experimental results that Cai *et al.* [44] shown, are very encouraging, while the approach contains many common elements with the algorithm that forms the main part of this thesis. However, the basic difference is the fact that this system cannot be implemented at the level of a personal computer, but only in terms of router.

Coull *et al.* [45] propose an algorithm which is based on a peer-to-peer network of individual BGP (Border Gateway Protocol) routers. The most significant contribution is that they apply trust management between nodes forming the peer-to-peer network. Therefore, the validity of the recommendations for malicious activity, initiated by each node is directly related to the degree of trust shown by other nodes to him. The approach based on trust management, which was followed by the authors, is widely accepted in other applications of peer-to-peer networks and naturally it makes sense to build in security applications. In this project they present an algorithm of trust, which seems to meet the requirements of their architecture. However, it is not absolutely clear the way in which routers recognize and record the attacks.

Arora [46] presents an algorithm for distributed monitoring of the spread of malware. The architecture of the system is based on two levels. At the lowest of them, each Internet Service Provider (ISP) monitors exclusively a range of IP addresses allocated to them, and then transmits this information to all the other cooperating Internet Service Providers. The author follows a purely mathematical approach of the issue

and it does not address all the technical issues, but focuses on how we can extract useful statistical data obtained from measurements of the system, which is still in theoretical level. The ultimate goal is to extract a non-biased estimate for the number of infected computers on the Internet and the rate of infections.

Locasto *et al.* [47], use the peer-to-peer network architecture in order to implement the distributed early warning system Worminator. The design of Worminator system and other functional elements, as the way to select nodes for the messaging are similar to the PROMIS algorithm. The basic difference between the two systems is mainly the overall approach of Worminator, as according to the authors, it aims to identify malicious exploratory activity (via distributed crawling, hidden scans etc.) and is proposed to install in large enterprise networks.

The Indra system [48] and the PROMIS algorithm share many common elements. First of all, both of them utilize the infrastructure of peer-to-peer networks in order to provide accurate information to their members. The basic difference is that PROMIS tries to protect its members, making as far as possible a realistic assessment of the overall malicious activity whereby each member of peer-to peer network may automatically select the optimal security policy. Furthermore, the Indra system uses its members to identify initially and then exclude the sources of malicious activity. Since the scope of Indra is much more ambitious, it is much more difficult to be achieved. This finding is supported by the event that there is no system that utilizes this algorithm and operates in real conditions in order to evaluate accordingly.

Keromytis *et al.* [49] have implemented the COVERAGE algorithm that has enough characteristics with the PROMIS algorithm. The COVERAGE algorithm is more complex than the PROMIS, because it is targeted to the recognition and categorization of new forms of malware. Moreover, an additional positive element of their work is the fact that they have experienced all the technical problems about the implementation. Also, they have an operating prototype in a very stable form.

4.3 Centralized Systems with Distributed Sensor Networks

Singh *et al.* [50] deepened especially in how the signatures could be extracted automatically for unknown worms. Using, also the Rabin footprint algorithm, they implemented the EarlyBird protection system, which is clearly more centralized than other approaches. It is obvious that Singh *et al.* [50] primarily gave special importance to the efficient extraction of the signatures and secondary in the architecture of their system. The performance results are satisfactory, more specific in optimizing the extraction of the signatures.

The PISA system [51] follows a similar approach in automatic extraction of the signatures. Analyzing the characteristics of different packages such as the source address, the message size, the protocol that is used and the destination port, and the percentage of available bandwidth that have similar packages, various signatures are exported. The generated signatures depending on their appearance in various web flows are characterized permanents or temporaries and are used to recognize the existence of malware or a Distributed Denial of Service Attack (DDoS Attack) that is ongoing. The first results of the survey are encouraging, but we should know the exact frame time required for the extraction of signatures.

Toth *et al.* [52] showed some interesting approaches to deal with malware. Specifically mentioned the similarity of connection patterns which used by both Cai and the Singh for their systems. In addition, Toth *et al.* [52] cite concepts the causality of connection patterns, i.e. the time sequence in which they occur some suspicious events and unsuccessful connections that attempts to make each infected system. The prototype which they built is in very early stages, and the three criteria on which based are suitable for the detection of rapidly spreading software, particularly if it can work distributed on a large scale.

Cheetancheri *et al.* [53] created a Collaborative Intrusion Detection System based on Sequential Hypothesis Test - SHT. Their algorithm chooses random cooperatives node and forwards the message that a worm is detected. If the node that receives the message has also detected the worm, it confirms and then it forwards to other random nodes. When it completes some predetermined number of forwards of the original message, depending on the percentage of nodes which are confirmed the initial observation, the algorithm decides on the existence (or non-existence) of the software.

This technique may contribute to the updating of some nodes on new forms of malware, but it is not clear if it simply aims to enhance the identification of unknown forms of malicious software, or the early warning of other nodes on upcoming malicious activity. In the second case it may not be very clear how could react the nodes that received the relevant information. Cheetancheri *et al.* [53] contribute to face the problem of false positives that have the security applications. Their work, in this way, it fixes to a certain degree this problem, but it may be recorded as a system that limits the rapidly spreading malware.

The DSHIELD system [54] is the most distributed and collaborative attempt to deal with any kind of digital threat. The DSHIELD is powered by data from more than 500,000 computers, which they are located scattered in 50 countries. Hence, it is a fully functional and stable system with many years operation. The central servers of DSHIELD collect these data, then analyze them and export some general trends about the risk assessments of the Internet. The big data available to DSHIELD allow it to proceed to a more detailed analysis, calculating trends for increase or decrease the assays in specific ports. This function helps to become evident the type of the security vulnerability that cybercriminals attempt to exploit. Also, this system provides geographic information about the country of origin of the various attacks, as well as historical information on the evolution of these phenomena in time. According to the author, the most important information is that the DSHIELD provides the overall risk assessment of the Internet, which is represented by the color coded Green, Yellow, Orange and Red. This code is also used in many other human activities, such as to describe the state of vigilance in the army or the risk of a potential terrorist hit. The model that is presented in this dissertation will be described as a decentralized and fully distributed version of DSHIELD. Therefore, these two systems share several common elements and follow a similar philosophy.

Similar architecture follows the DeepSight [55] from Symantec Company, which also is based on a distributed sensor network through which data are collected. The warning system DeepSight is a commercial product and the manufacturer, does not provide all the data required for the detailed study of their operations.

4.4 Alternative Implementations

Yang *et al.* [56] presented the CARDS System (Coordinated Attack Response & Detection System). The architecture is interesting because it represents the distributed attacks with some general patterns which modeled as graphs. Each node that participates in the CARDS records the security incidents.

Then the CARDS is composing information from various sources and procedures the appropriate graphs. The graphs can confirm or refute the evolution of distributed attacks, which otherwise would be extremely difficult to be detected. However, the CARDS is still in planning mode so it has not come into operational function in order to evaluate in action the real capabilities.

Michael *et al.* [57], following the military philosophy, recommended the implementation of software decoys, when it distinguishes suspicious sequences of actions, then it behaves in such a way to give the impression that the prospective attacker has achieved the infection. In fact, the software performs at a very slow rate, the orders that received, by delaying the attacker. The biggest difference from the known honeypots [58, 59] is that the deceptive software by Michael *et al.*, proposed to be installed in mainstream commercial systems and not in any special designed systems for this purpose, such as the honeypots. Furthermore, the examples that presented are quite complicated, and the fact that there is not provided an implementation in the proposed system, means that it would be required enough effort to include all the critical applications that are necessary for its effective operation.

Ye and Farley [60] are using signal processing techniques to detect and separate the malicious from the normal activity in a system. Specifically, looking at various filters to cut noise from the traffic signal, they propose an application that depends on the type of malicious software. The analysis of the existing technologies (pattern recognition and behavioral analysis) for the detection of malicious code is valid, but they do not mention a specific example of theoretical or practical way, to determine if their method can be applied and if the results that it has, it indeed better than that offered by the existing approaches.

Webster and Malcom [61] use algebraic transformations to a subset of the IA32 instruction set, to detect polymorphic and metamorphic viruses. Yoo and Ultes-Nitsche [62] present a technique based on Self-Organizing Maps to identify unknown viruses. While experiencing significant successes, by identifying 84% of unidentified

forms of malware have been tested in their system, they also exhibit a fairly high rate of false positives. This percentage is around 30%, which is prohibitive for the adoption of this system in a wide area from normal users.

CHAPTER 5

PROBLEM STATEMENT AND METHODOLOGY CHOICE

5.1 Introduction

It is true that peer-to-peer networks is the key factor in spreading malware due to the fact that these ones is the largest contributor of network traffic on the Internet and are considered to be a huge security problem. Apart from their extensive use of the conduction of several illegal activities such as the file sharing that are protected by copyright legislation, they are also constitute the main gateway of malware in many cases. [63, 64]. Therefore, peer-to-peer networks have been recorded as a threat against the security of information systems [65]. The Distributed Containment Algorithm PROMIS uses the peer to peer networks in order to deal with malware, aiming to highlight the positive contribution that can be made to increase the security of information systems, instead of the common belief which supports that they degrade it.

The treatment and the containment of the spreading malware as seen from the relevant section of the literature review, constitutes a distinct research area with specific conferences taking place on this very important issue and scientists dealing with this subject. At the same time, the importance of addressing reflected by the fact that most technologically developed countries have special groups and vectors for the shield of their critical infrastructure. In addition, the problem of uncontrolled spread of malicious software is also evident in less special users, forcing the computer and software industries to pay considerable resources dealing with the immediate impacts.

The proposed PROMIS algorithm uses techniques, tools and the method of the following areas:

- Securing Information Systems. (ACM: K.6.5, D.4.6, K.4.2)
- Theory and Modeling Network. (ACM: E.1)
- Simulation. (ACM: B.2.2, I.6.1)
- Epidemiology. (ACM: -)

The combination of different scientific fields, led to the creation of specific research subareas, as Kepahrt [66, 67, 68, 69] with the Computer Epidemiology and recently with the Computer Hygiene [70].

5.2 Problem Statement

The problem, in which the thesis is based, is to design and deploy a peer-to-peer network and with the use of epidemiological models to collect significant data results in order to optimize the performance of the algorithm under a malware epidemic and in what degree it manages to protect its own members. Our goal is to achieve significant data results from the simulations, to assess the performance of our algorithm in order to be reflected the degree to which it can reduce the spread of malware by investigating the parameters which affect the behavior of the algorithm.

PROMIS follows a distributed, complex architecture and therefore affected by various parameters. The simulations allow the recording and the evaluation of the impact of each of these parameters on the overall performance of PROMIS. The algorithm is affected by both internal and external factors. The external factors, relating to the environment and the assumptions in which simulations are carried out, affect each cooperative containment system of malware.

Environmental factors with particular importance are:

- The network topology in which simulated the performance of the algorithm.
- The infection rate β (pairwise rate of infection) malware.
- The number of initially infected nodes.

Instead, the following parameters are related exclusively to the algorithm:

- The threshold limits of security levels
- The number of security levels.
- The total number of members of PROMISGROUP.
- The number of the past measurements used by each node to extract the local network malicious activity.

The simulations focused on the investigation of two fundamental questions concerning the usefulness of the algorithm:

- Whether a system based on the algorithm is able to limit the spread of malicious software
- What configuration in the parameters are necessary to maximize its effectiveness.

5.3 Methodology Choice

In order to answer the above questions we follow the widely accepted methodology “*Simulation Modeling and Analysis*” [71], the conducting of experiments by varying only one parameter and keeping all other constants.

Since the topology of the graph that uses the simulator is particularly important and affects greatly the results, all the experiments were performed in full scale free graphs in order to have a more realistic approach in to the problem. The remaining parameters during the conduct of each type cycle of experiments were held constant, but is maintained the same values to the parameters for all cycles of the experiments.

5.3.1 Choosing simulated over realistic experiments

The simulation is based on the numerical computation of the value of the subsystems of a model in order to assess the overall system behavior. It is widely used by almost all scientific disciplines while it is the most widely used method for risk assessment of various biological viruses in epidemiological research. It is also the basic technique for testing new algorithms, models and architectures in computer science and the theory of networks. The simulation is the basic tool for supporting the efficiency of PROMIS algorithm for technical and ethical reasons. The PROMIS algorithm must be carried out at the same time by a large amount of peer to peer networks in order to have significant results. The measurement of its efficiency, via the implementation and functionality of this system, in real time remains a very challenging task due to the below factors:

Risk: The creation of a virus or a worm in the laboratory, in order to measure the effectiveness of this algorithm in a local network would pose important risks, since many viruses and worms have managed to escape from the laboratories that were created and caused serious damage. Even more dangerous it would be the fact if we are going to use real malware, since in many cases is not fully known the whole functionality and therefore it could involve destructive properties. Therefore, in the case of an accident, maybe an involuntary escape, it would cause serious and perhaps irreversible disruption in the systems which would infect. So, it is considered necessary the use of the simulator.

Distributed Nature of the Algorithm: The Cooperative Containment Algorithm PROMIS performs better when there is a large number of collaborating entities. It was estimated that it would be extremely difficult to gather hundreds of volunteers who would be able to obtain sufficient amount of information in order to perform a satisfactory evaluation of the risk and taking the appropriate countermeasures. Additionally, it is reasonable even though there were so many volunteers available, they had significant concerns to install a system that is in progress and hence to be in an unstable situation, and much more to allow it to configure automatically the security policy of their computers. Moreover, it would be really difficult to achieve in a system like this software maintenance and technical support to users for a long time, because it needs time to make valuable points as it should be executed in many different hardware and software systems.

Difficulty in capturing the empirical data: The Cooperative Containment Algorithm is designed exclusively to help dealing with malware. Therefore, in cases where the spread of a virus or a worm is limited, the PROMIS algorithm is not the most appropriate form of protection. In contrast, the effectiveness of this algorithm can be measured only in large-scale malware attacks. These attacks do not happen very often, neither regularly, while the cost and malfunctions that caused are disproportionate of the frequency in which appear. Therefore, for practical reasons, it would not be possible to wait for the epidemics in order to examine the performance

of the algorithm, especially if taken into account that the algorithm until it reaches into a final stage needs multiple corrections, modifications and optimizations, which in order to achieve this, are required repeated tests.

The spread of malware has significant similarities and is subject to the same restrictions as the spread of biological viruses. For this reason, we would use methods and assumptions which are widely accepted in epidemiological research. Specifically, the chosen study of the spread of malware through of the simulations is a key research tool for epidemiologists [72].

5.3.2 Evaluation of simulations

According to the theory [71] should check first the evaluation of the simulator in simpler models and make sure it works properly before simulated large and complex systems. Alternatively, it can be verified that the simulator behaves properly in the simulation of a system for which there is detailed solution. The PROMISsim simulator checked in this way. Specifically, was modeled the spread of malware without the implementation of any restrictive measure in homogeneous graphs and compared the results of the simulator with that resulting from the solutions of differential equations that describe similar effect [42, 73] The results, as shown in the next figure confirm the correct implementation of the simulator.

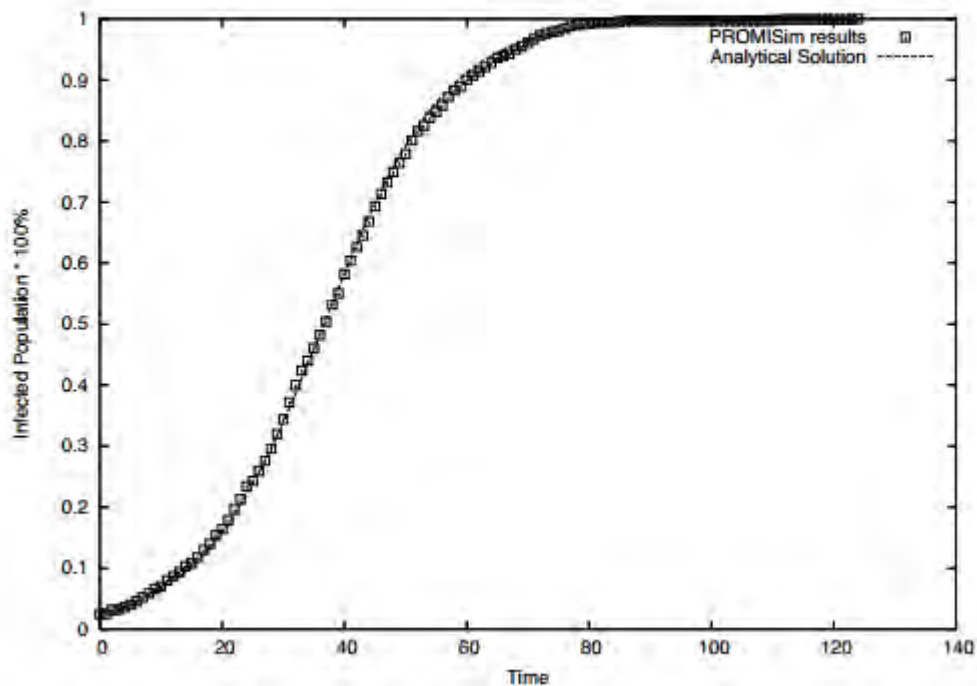


Figure 5. 1 Comparison of the theoretically expected results with the experimental data obtained by the simulator

Another parameter that affects the reliability of the results is random or stochastic phenomena which can occur during the experiments leading to completely different results from expected. This problem is treated relatively easy with the conduction of multiple simulations and the calculation of the average results. In our experiments which performed, the above method was followed, the multiple confirmation of measurements via the sequential conduction of various experiments and the calculation of the average of the results. (Figure 5.1 and Figure 5.2)

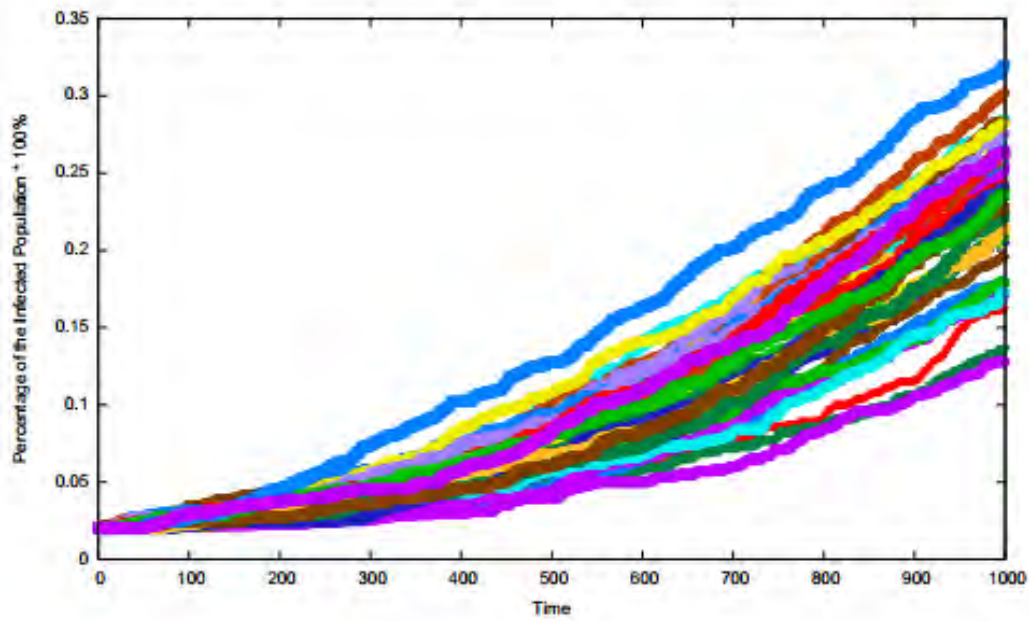


Figure 5. 2 Comparison of the theoretically expected results with the experimental data obtained by the simulator

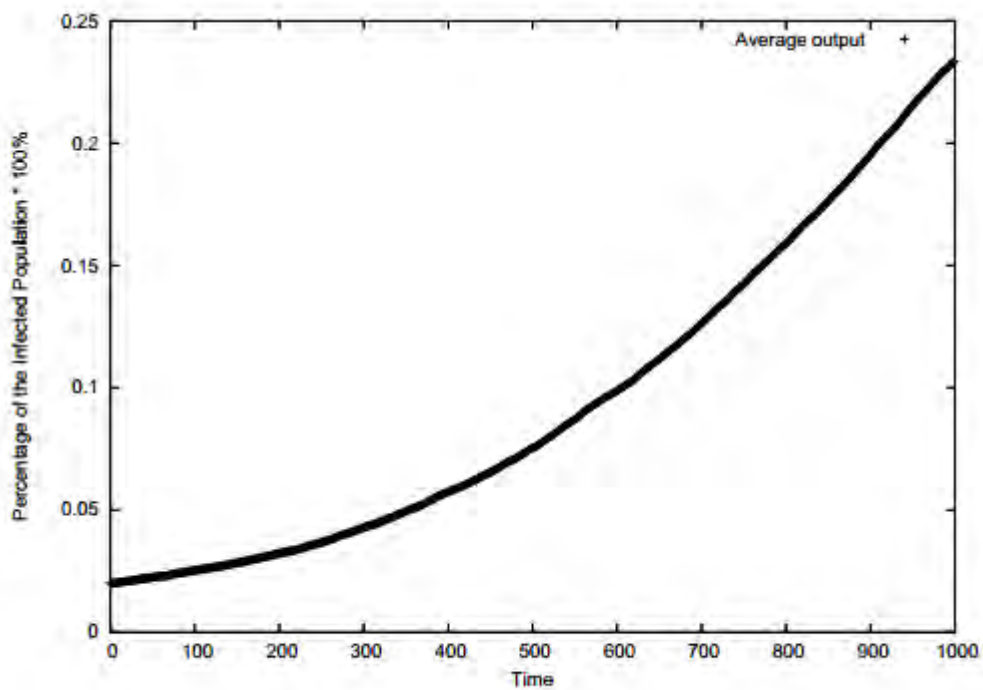


Figure 5. 3 The average of multiple executions of the experiment

5.4 PROMIS Algorithm

The initial basic idea behind the design of PROMIS algorithm is that the information which may has a user, by monitoring the local system, is insufficient to assume, even if he is highly experienced in security, if some malware epidemic takes place. It is clearly easier and possible to verify the presence of a malware epidemic, comparing the log files of the security incidents of a large number of users.

However, it is extremely difficult for users of a peer to peer network, specially designed for this purpose, to constantly exchange log files of security incidents, because of the load that would introduce in the Internet as and it would require significant processing power of each node to analyze them. In addition, these files contain enough data which are considered confidential. In our thesis, only the change rate of the events recorded locally on each system is transmitted to all the members of the peer to peer network at regular short intervals, i.e. the percentage of increase or decrease of security incidents of each participating node.

As the percentage is monitored at regular short intervals (timeslots), it can be extracted the change rate of the security incidents, for a time frame of several slots. Furthermore, each node of the application gathers all the change rates of the other nodes in the peer-to-peer network and calculates the average change rate of the peer to peer network [70, 82].

PROMIS constantly perform two operations. A daemon called *Handler* collects the messages which other nodes transmitted. These messages describe the malicious activity that have been recorded locally. After this, the overall malicious activity is exported. Having set the limits to increase and decrease the security level of the system, the Handler compares the overall malicious activity and either increases the security level of the protected system or decreases either do not make any change in the security policy using the following equation:

$$p_{avg} = \frac{(\sum_{i=1}^n P_t^i)}{n}$$

Figure 5. 4 Total Malicious Activity of P2P Network

A daemon call *Notifier* monitors the log file on the local file system at regular time intervals and export the rate of the intercepted malicious activity against this host using this equation:

$$p_t^n = \frac{h_t^n - \frac{(\sum_{i=t-k}^k h_i^n)}{k}}{\frac{(\sum_{i=t-k}^k h_i^n)}{k}}$$

Figure 5. 5 Local Malicious Activity

t	Time Space
h	Number of Attacks
p	Total Attack Rate in a set of time intervals
n	Node ID
k	Number of past periods
λ High	Default Upper Security Limit
λ Low	Lower Security Limit

Table 5. 1 Semantics of Variable Equations 1 and 2

The *Handler's* main responsibility is to automatically adjust the security level of the local system based on the subsequent directives

- If $p_{avg} > r_{high}$, then increase the security policy by disabling non-essential services as for example HTML preview in mail clients or by increasing the security settings of the installed web browser, where r_{high} is the predefined threshold to increase the security settings of the PROMIS system.

- If $p_{avg} < r_{low}$, then decrease the security policy by reactivating the above-mentioned services, where r_{low} is the predefined threshold to decrease the security settings of the PROMIS system.

- If $r_{low} \leq p_{avg} \leq r_{high}$ do nothing.

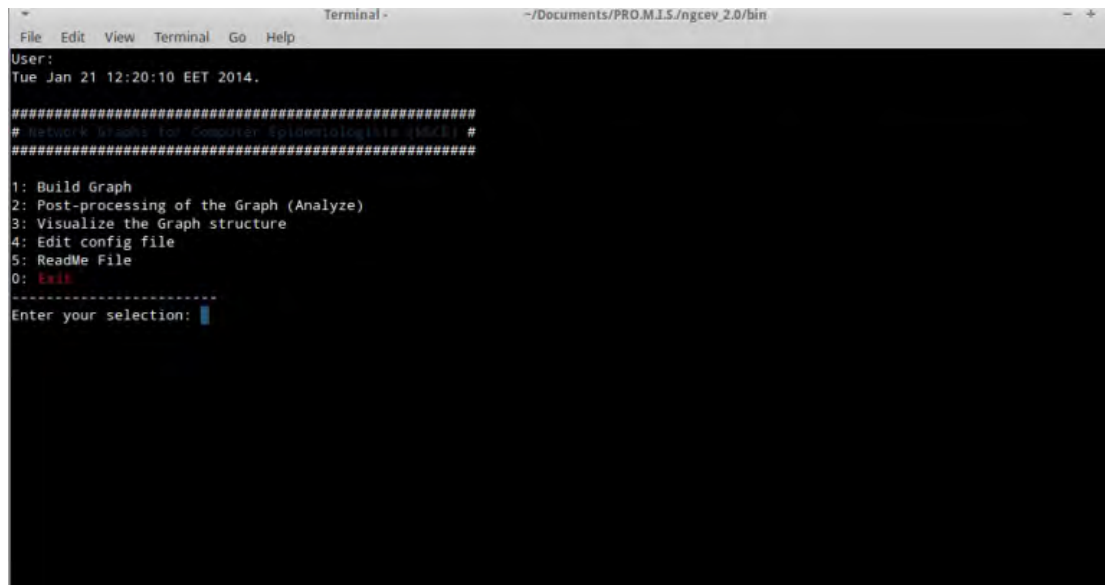
5.5 NGCE - Network Graphs for Computer Epidemiologists

The simulation of complex phenomena such as the spread of malicious software is a very complicated process. The quality of the results depends mainly on two basic parameters:

- Whether the simulation model realistically represents the test scheme.
- If the implementation of the simulation model contains programming or other errors.

The construction of a very large number of graphs with ad-hoc modes would be difficult and time consuming. By using the NGCE tool [80], it made possible the automation and customization of this work. Additionally, this application was designed to be a useful tool for a wider community of scientists engaged in similar research, technological or social networks, which require the use of graphs. For this reason was decided the interface of the tool to include both GUI and command line interface. NGCE has the ability to generate reproducible graphs and is implemented entirely in the Java programming language and consists of 10 classes with more than 4000 lines of code and is available via an open-source license at <http://ngce.sourceforge.net>. The structure is modular and is based on a class with

wider functionality, the Graph class. All the topologies made by NGCE tool are developed as separate modules, so if this required it is perfectly possible to add new topologies. The choice of the appropriate data structures is based on [81], while the initial implementation of the tool used some of these by permission of the author.



```
Terminal - ~/Documents/PRO.M.I.S./ngcev 2.0/bin
File Edit View Terminal Go Help
User:
Tue Jan 21 12:20:10 EET 2014.
#####
# Network Graphs for Computer Epidemiologists (NGCE) #
#####
1: Build Graph
2: Post-processing of the Graph (Analyze)
3: Visualize the Graph structure
4: Edit config file
5: ReadMe File
0: Exit
-----
Enter your selection: █
```

Figure 5. 6 The command line of NGCE Tool

The selection of appropriate parameters for the construction of a graph with the desired properties is in many cases a quite complicated process, which requires a certain level of knowledge. The creation of the specific application in Bash code (UNIX Shell) makes the process extremely friendly and simpler to the end user, since it directs him to use the selected commands to build any desired topology. This environment gives the user the ability to configure and choose, based on his own needs, the desired topology that wants to build by making the use of the option "**Edit Config file**". Along with the option "**ReadMe File**" which contains detailed information on how to create each topology, the user can select and adjust the config file properly. In the final stage, using the option "**Build Graph**", the user is able to implement the chosen topology and then with the commands "**Post-processing of the Graph**" and "**Visualize the Graph structure**" to construct the respective graphs using appropriate tools such as Gephi or Pajek. The application contains options 2

and 3 independently, giving the user the ability to construct and implement in a future stage according to the topology that has chosen the available graphs.

5.6 PROMISsim

In order to use the PROMIS algorithm, we run the PROMISsim simulator which developed entirely in Java programming language [82] and operates on graphs that have been generated with the NGCE tool, which so far covers homogeneous graphs, scale-free graphs, random graphs, lattices and custom graphs with specific properties that allows the generation of various graph models that are widely used in the computer viruses and worm propagation studies [80]. The complexity of PROMIS algorithm determined the size of the graphs which we used. We performed a number of simulations to eliminate stochastic phenomena in order to validate the correctness of PROMIS algorithm. To check the validity of our results we modeled the uncontrolled propagation of various worms in different full scale-free graph environments and compared the results of our simulator with the expected analytical solution of the *General Epidemic Model* [26]. PROMIS simulator is so far capable to model the spread of a worm or a virus using the well know S-I-R (Susceptible-Infected-Recovered) model.

The following figure gives an overview about the Script Code of PROMISsim

```
User:
Sat Jan 25 10:03:45 EET 2014.

+-----+
| PROM.I.S. - Proactive Worm Identification System |
+-----+

1: Run PROM.I.S. Simulation
2: Export PROM.I.S. Results
3: Show GraphTopology Information
4: Show Simulation Results
5: Remove Simulation Results
6: Create Required Folders
7: Edit config file
8: Create Network Graphs (NGCE Tool)
9: ReadMe File
0: Exit
-----
Enter your selection: █
```

Figure 5. 7 The graphic user interface of PROMISsim command line

This environment gives the user the ability to be transferred at an early stage of the NGCE tool using the option "*Create Network Graphs (NGCE Tool)*". Also, there are the options "*Create Required Folders*" for the creation of appropriate files for the optimal operation of the implementation as well as the option "*Remove Simulation Results*" in which the user can delete previous measurements from the database. In this Script Code, we have added the option "*Edit Config File*" in which we can configure the variables (infection rate, etc.). An important role plays the options "*Show Simulation Results*" and "*Show Graph Topology Information*" by which the user can see the simulation data which is recorded and can derive important data from each topology which carried out the experiments.

After using the first option "*Run PRO.MIS Simulation*" is now able to start the simulations by selecting the repetition rate and the folder name that results will be saved. At this point it is worth mentioning that the present Script Code has been created in such a way as to avoid the input of error parameter in one of the available options, as well as repetitive messages in case of error (e.g. not acceptable price to case number entry against alphanumeric.) or if deletion of past measurements.

CHAPTER 6

SIMULATION RESULTS

Gephi

We use the Gephi tool in order to design and carry out meaningful results as the average value of path of nodes, the total number of short paths of each topology as well as the average distance of a selected node from its neighboring.

For the design and the representation of each topology, the option "**OpenOrd**" will be used as graph layout, which allows us to extract important information and offer us the ability to parameterize by graphical interface each given topology.

In this chapter we present and analyze the following extraction methods of results:

1. Betweenness Centrality Distribution

This simulation records the frequency which determines how often a node appears on the nearby paths between nodes as follows:

For a graph $G: = (V, E)$ with V vertices calculated as follows:

- a. For each pair of vertices (s, t) , calculate the shortest paths between them.
- b. For each pair of vertices (s, t) , calculate the fraction of the shortest paths that pass through the vertex
- c. Calculate the sum of the fractions of all the pairs of vertices.

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

2. Closeness Centrality Distribution

This action calculates the average distance from a given node to everyone else.

$$C_C(v) = \sum_{t \in V \setminus v} 2^{-d_G(v,t)}$$

6.1 Network Topologies

6.1.1 Homogeneous graphs

1. Construct a graph with N nodes and no edges.
2. Join each node to all other nodes.

A homogeneous graph with N nodes is always $(2 - N) / 2$ edges

NGCE system parameters to generate the Homogeneous Graph topology:

Model type	Homogeneous Graph
Number of Nodes	2000

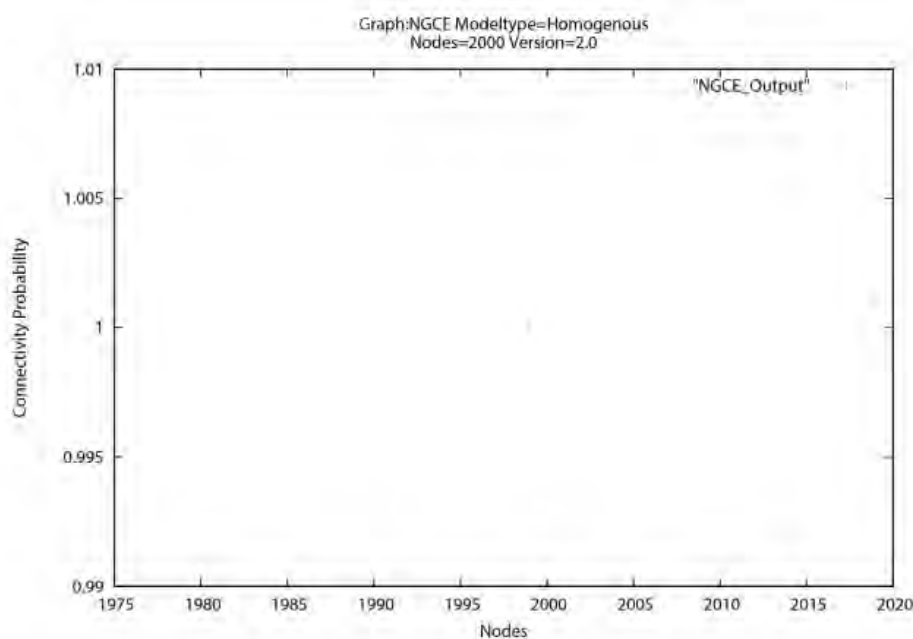


Figure 6. 1 Distribution Probability of Homogeneous Graph - 2000 Nodes.

NGCE system parameters to generate the Homogeneous Graph topology:

Model type	Homogeneous Graph
Number of Nodes	5000

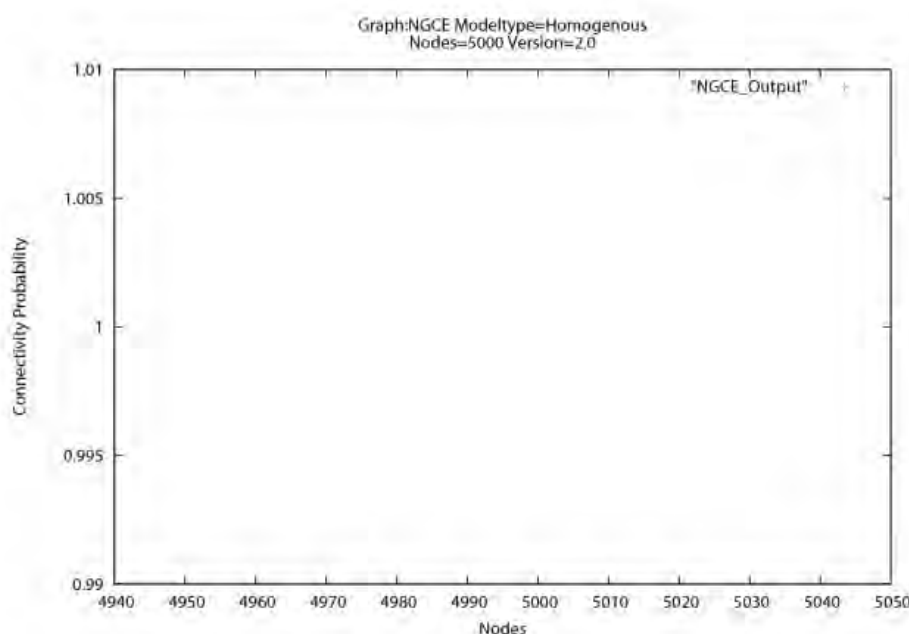


Figure 6. 2 Distribution Probability of Homogeneous Graph - 5000 Nodes.

6.1.2 Random Nodes

In random graphs, the position of persons in the space and the creation of contacts are random. More specifically each person has a fixed number of contacts through which it can spread the infection. The dynamics of diseases in random networks can be studied as a single branched process, from which we conclude that the early growth of the disease and the final epidemic size are reduced when compared with random mixed model.

The reduction in growth rate occurs for two reasons. Firstly, each infected person is infected by one of the contacts, reducing the number of susceptible in $n-1$. Secondly, each infected individual begins to infect vulnerable contacts, reduces the local environment even if the prevalence of the population is low, and therefore limits the rate of spread of disease.

The construction algorithm of random graphs according to Erdos - Reny [83] or Gilbert [84] by others is the following:

1. Construct a graph with N nodes and no edges.
2. Connect each pair consisting of nodes of the graph with probability P_{er} .

The curve of edges distribution that having each node, depends directly on the value of probability P_{er} .

Model type	ERGraph
Number of Nodes	2000
Number of Edges	29933
Propability	0.0075
Average Path length:	2.6174587293646825

Table 6. 1 NGCE system parameters to generate the ERGraph topology

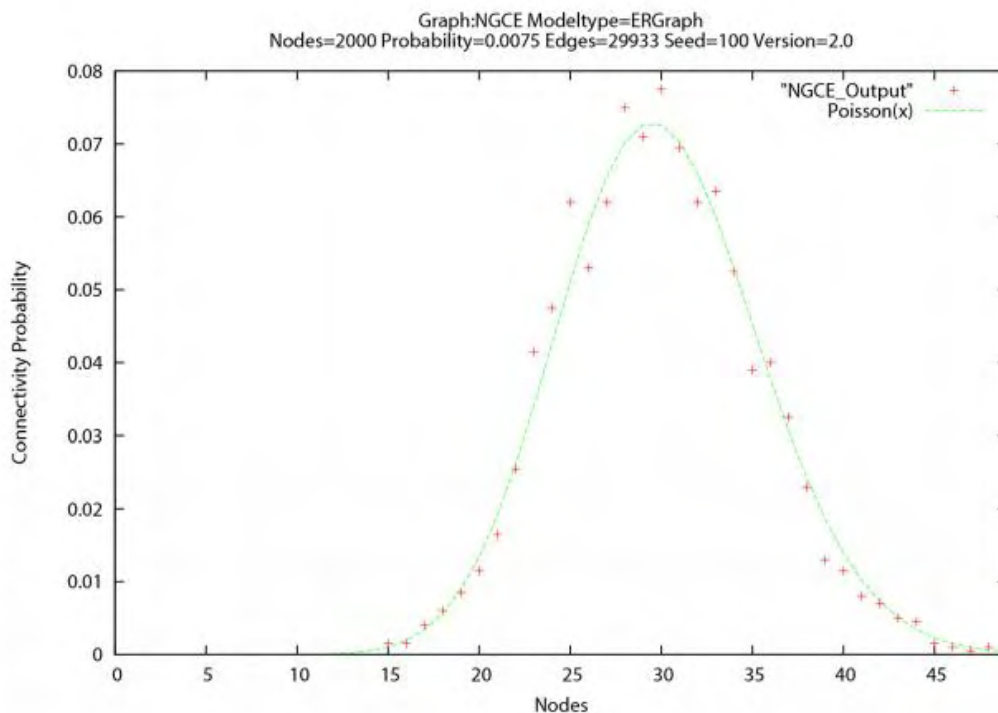


Figure 6. 3 Distribution Probability of ERGraph – 2000 Nodes.

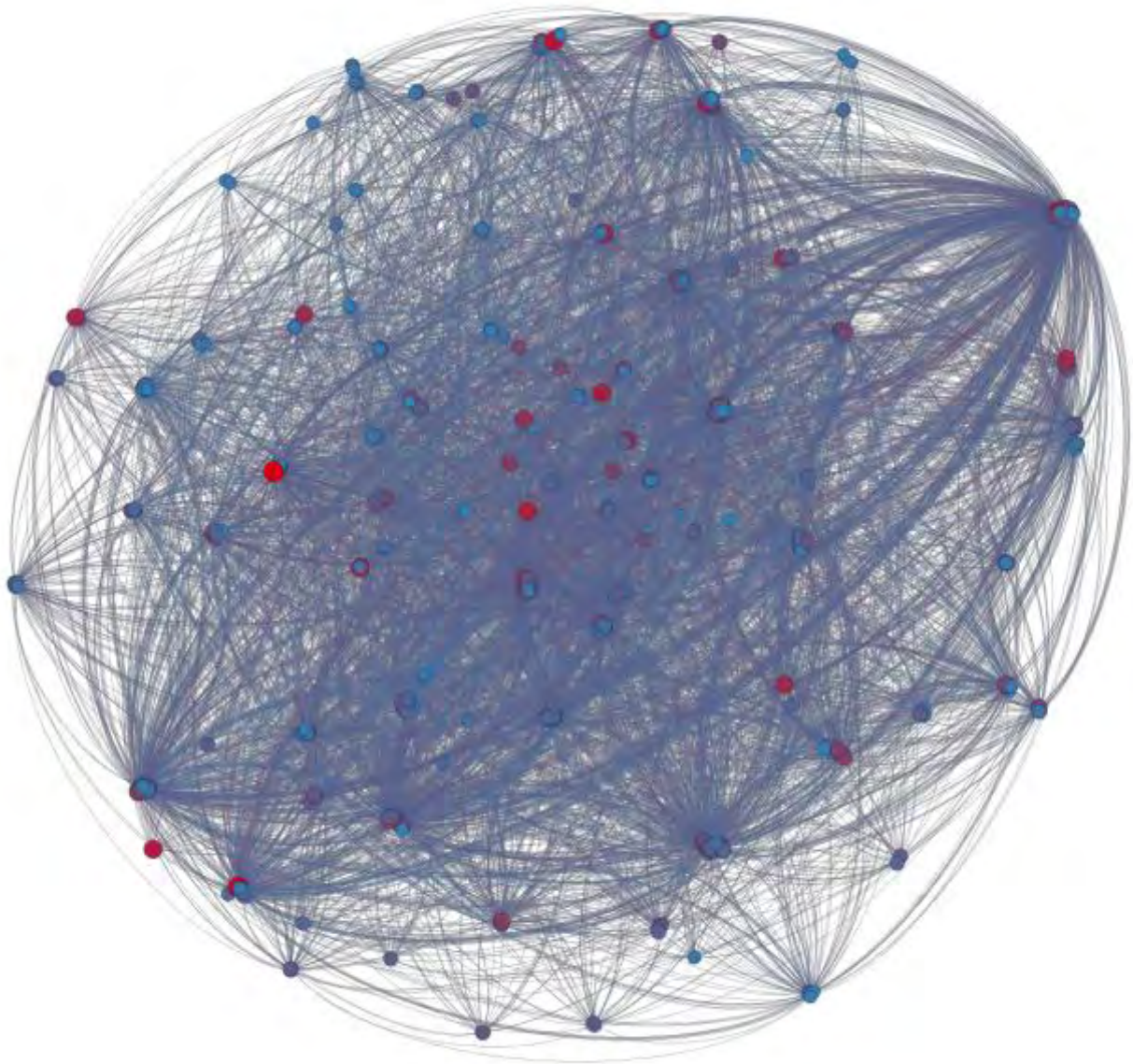


Figure 6. 4 Representation of ERGraph – 2000 Nodes

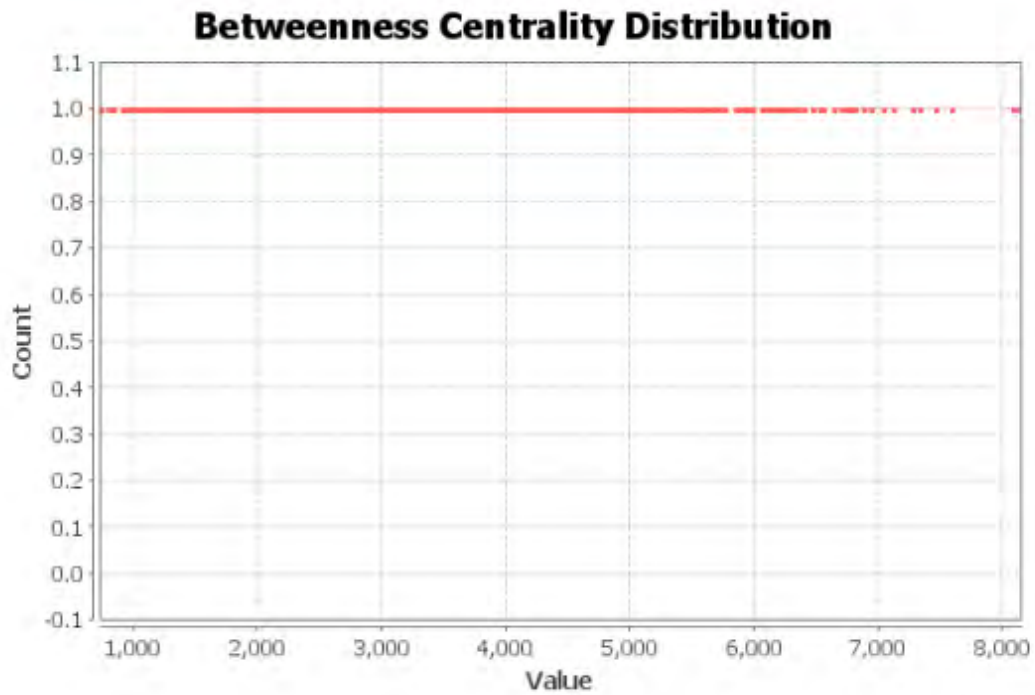


Figure 6. 5 Betweenness Centrality Distribution of ERGraph - 2000 Nodes

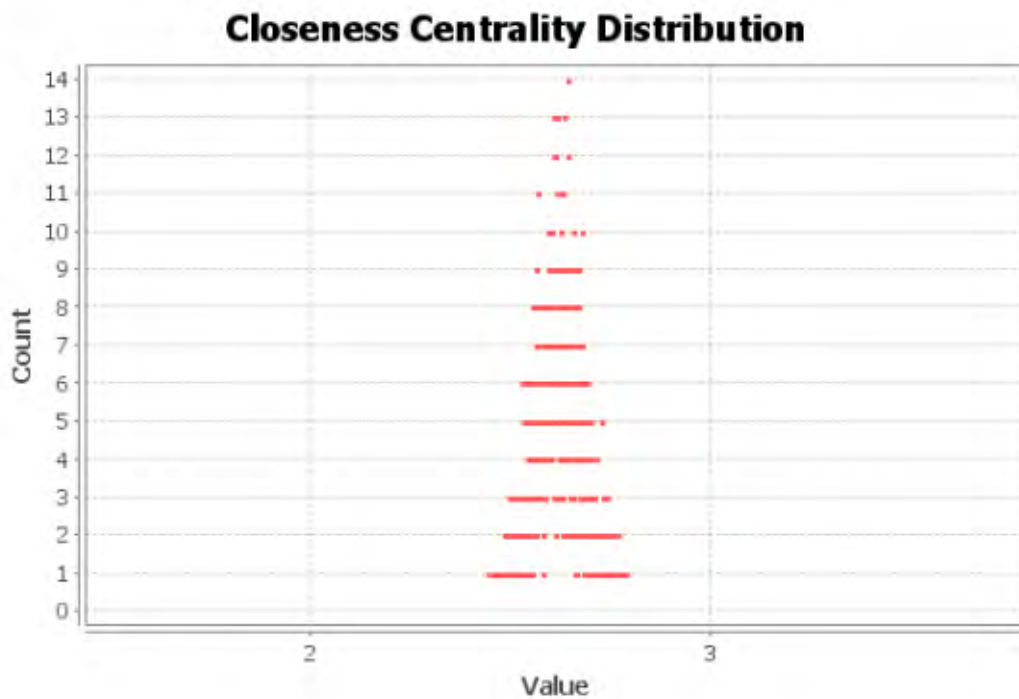


Figure 6. 6 Closeness Centrality Distribution of ERGraph - 2000 Nodes

NGCE system parameters to generate the ERGraph topology:

Model type	ERGraph
Number of Nodes	5000
Number of Edges	187529
Propability	0.0075
Average Path length:	2.3073226645329066

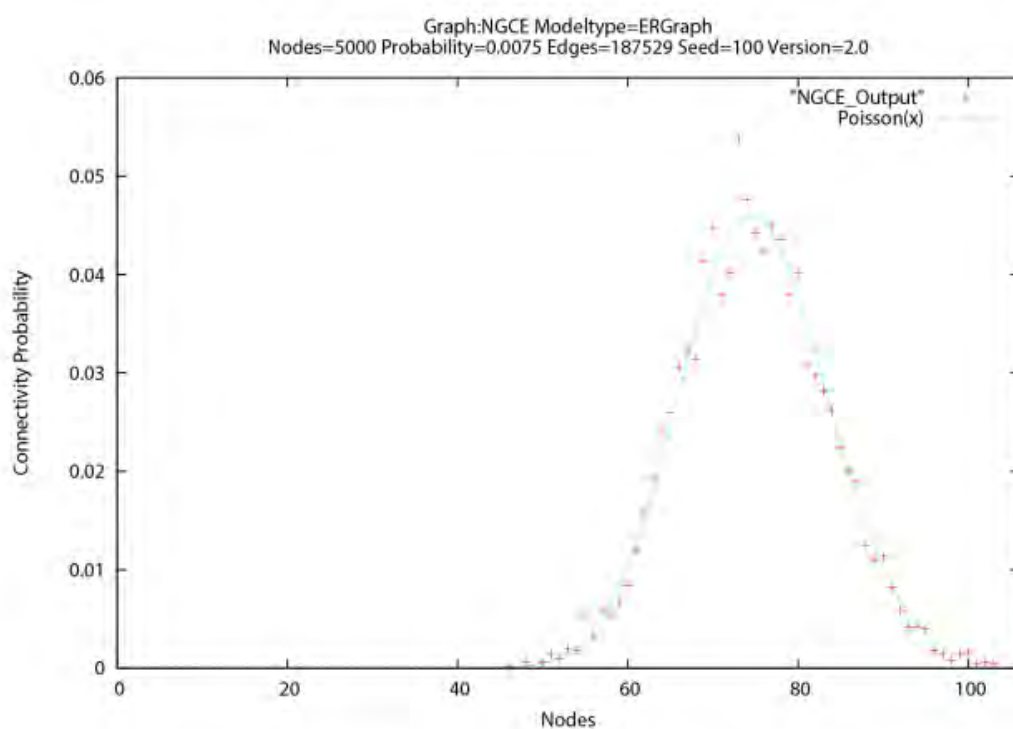


Figure 6. 7 Distribution Probability of ERGraph – 5000 Nodes.

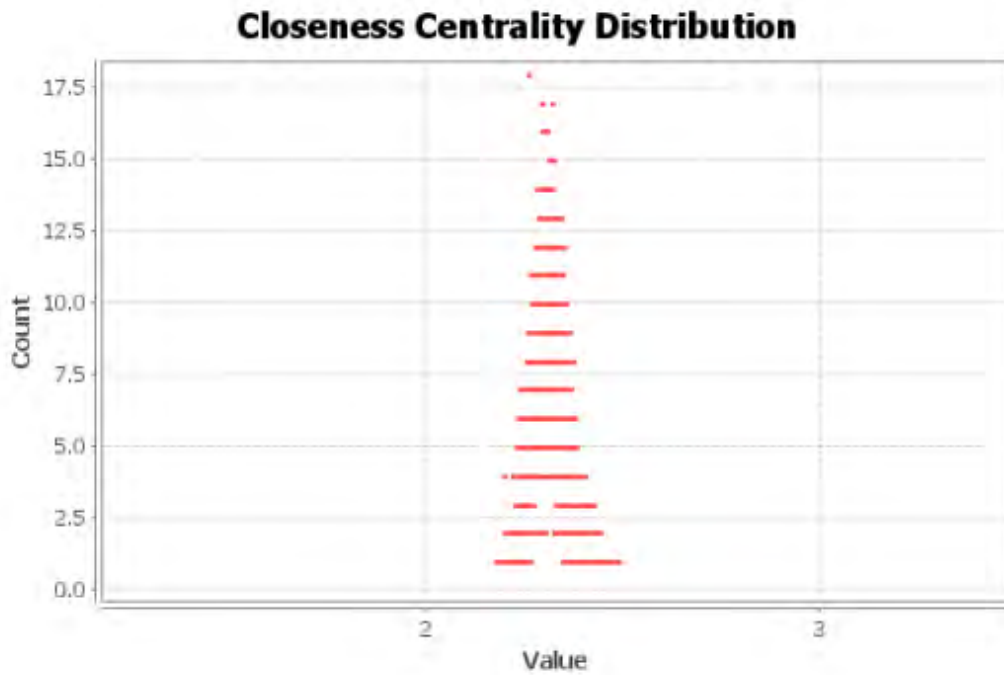


Figure 6. 8 Distribution Probability of ERGraph – 5000 Nodes.

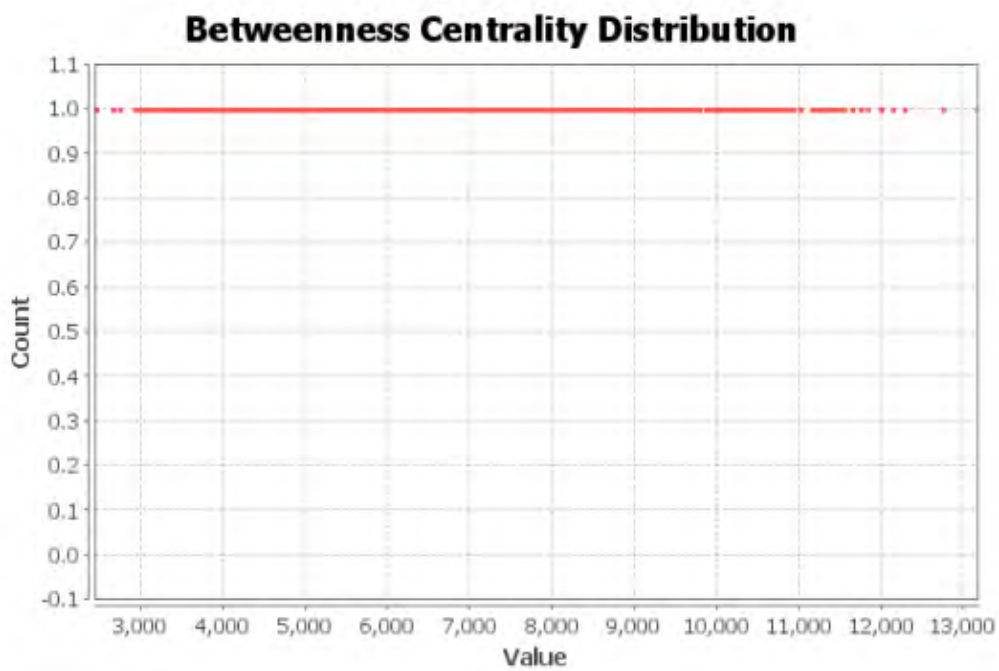


Figure 6. 9 Closeness Centrality Distribution of ERGraph - 5000 Nodes

6.1.3 Full Scale Free Graphs

Under certain circumstances it is possible for a graph to display temporarily features of a free scale graph. The removal of any of the above properties leads to graphs with only temporary features of a free scale graph.

The algorithm that generates graphs according to the methodology of Barabasi - Albert is the following:

1. If the number of nodes is less than the number of edges, randomly select a node and connect it with probability at node i .

$$P(k_i) = \frac{k_i}{\sum_j k_j}$$

2. Repeat step 1.

If the number of edges becomes about equal to the number of nodes, the manufactured graph shows features of free scale graph. The inclusion of new nodes in the graph leads to a progressive damage of the structure of the free scale that initially displayed.

- Build a tank K and add in this mo initial nodes.
- Build a tank L with all nodes.
- Remove a random node i from tank L and connect it with a random selected node from the tank K
- Add node i to tank K
- If the tank L is not empty, repeat 3.

Model type	Full Scale Free Graph
Number of Nodes	2000
Initial Connections	10
Initial Pre-existing Nodes	200
Average Path length:	1.9183491745872936

Table 6. 2 NGCE system parameters to generate the Full Scale Free Graph topology

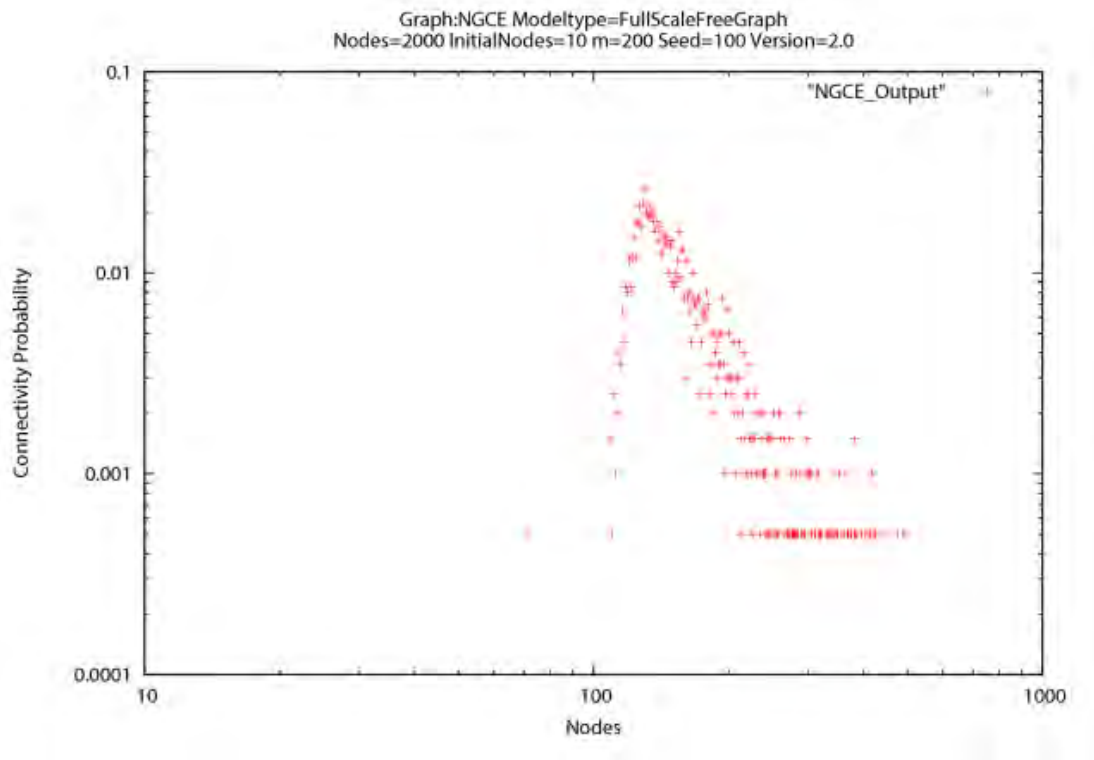


Figure 6. 10 Distribution Probability of Full Scale Free Graph – 2000 Nodes

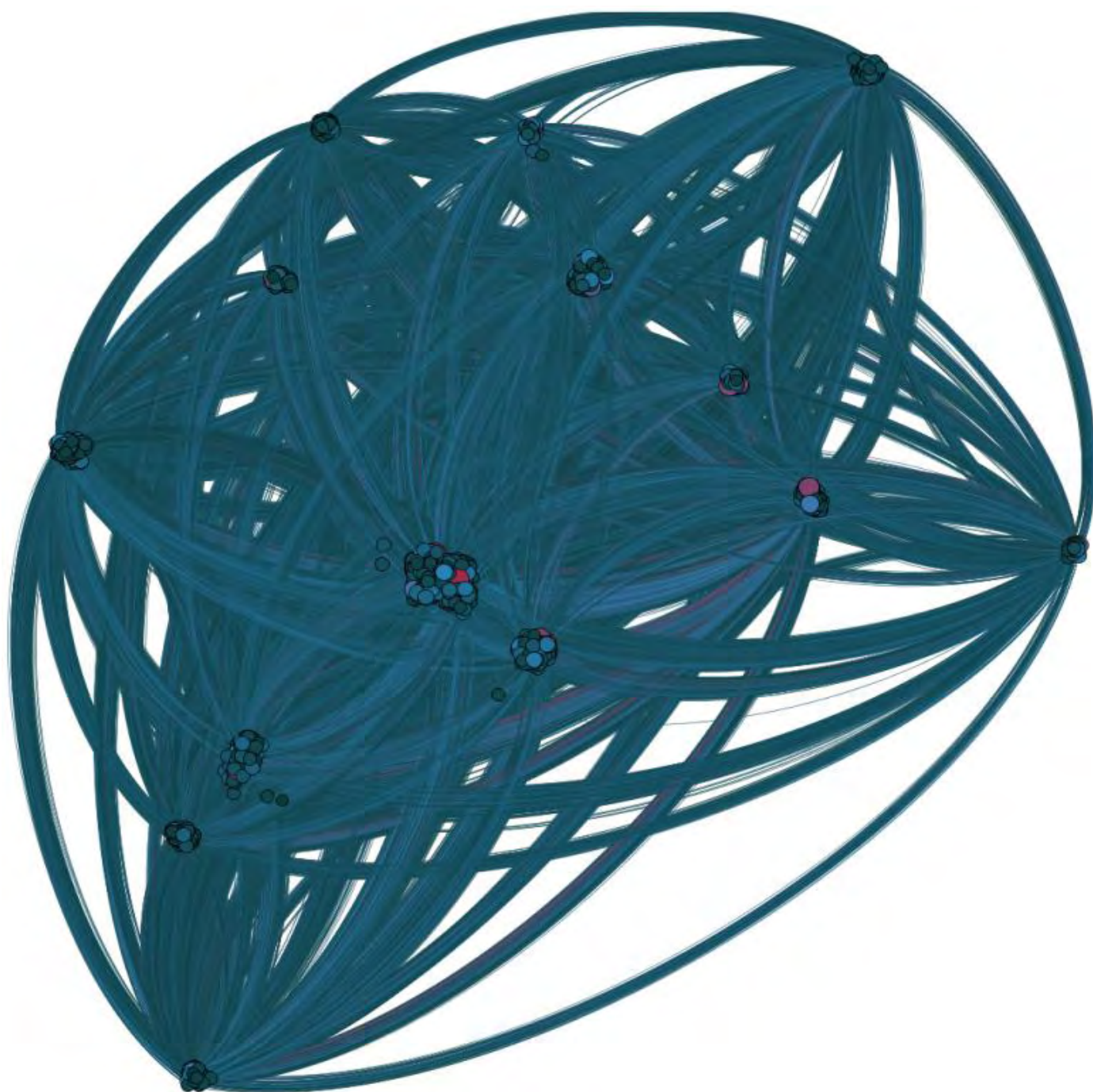


Figure 6. 11 Representation of Full Scale Free Graph - 2000 Nodes

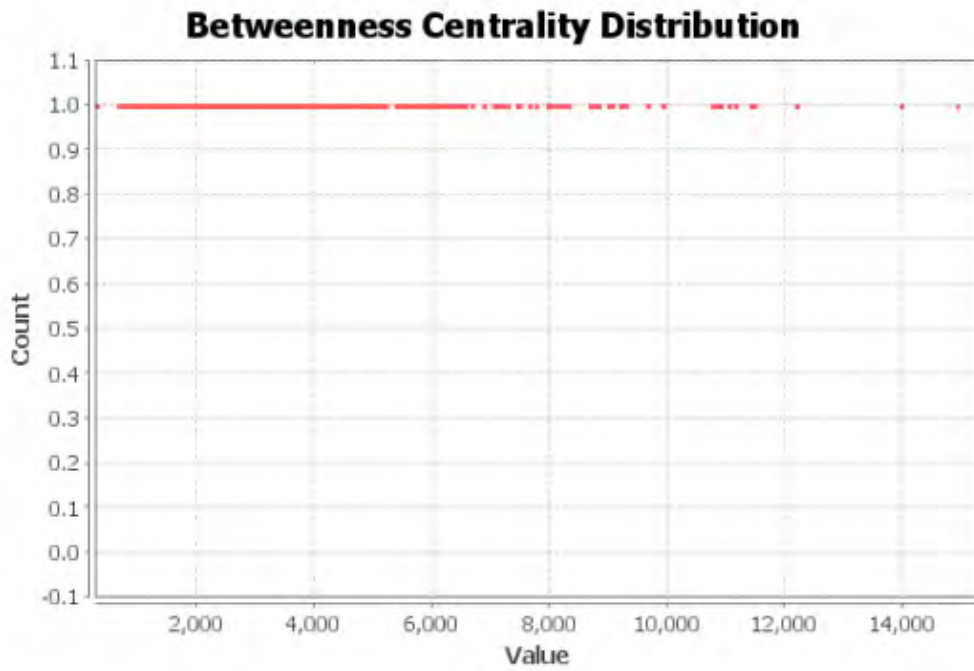


Figure 6. 12 Betweenness Centrality Distribution of Full Scale Free Graph with 2000 Nodes

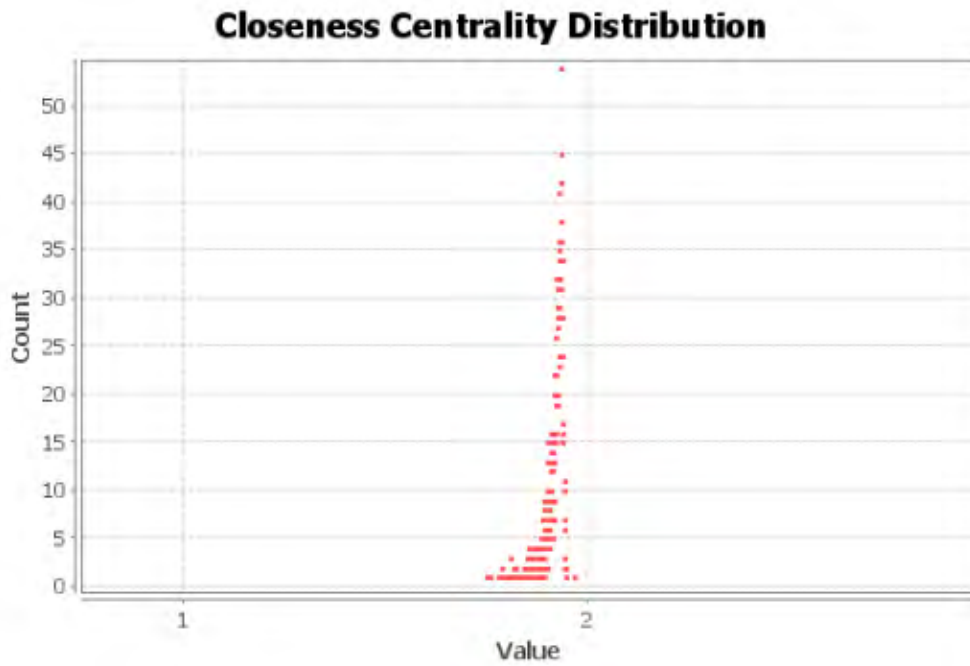


Figure 6. 13 Closeness Centrality Distribution of Full Scale Free Graph with 2000 Nodes

Model type	Full Scale Free Graph
Number of Nodes	5000
Initial Connections	10
Initial Pre-existing Nodes	100
Average Path length:	1.9183491745872936

Table 6. 3 NGCE system parameters to generate the Full Scale Free Graph topology

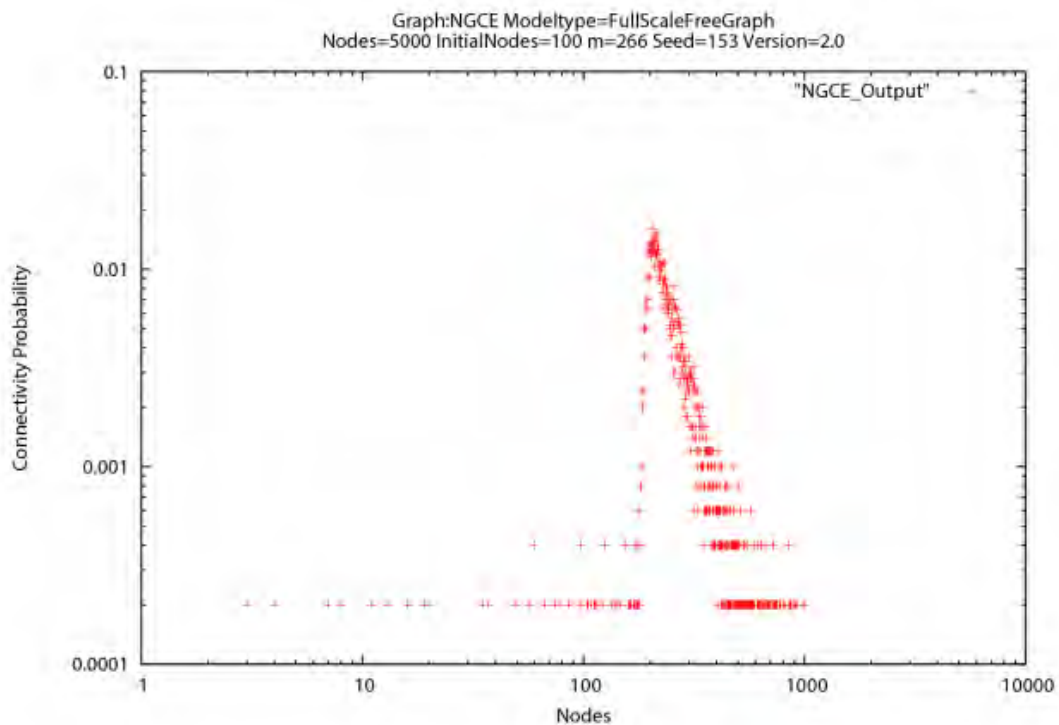


Figure 6. 14 Distribution Probability of Full Scale Free Graph – 5000 Nodes

In this point it is important to refer that our hardware infrastructure was not sufficient in order to complete the proper graphical display for Full Scale Free Graph in 5000 Nodes.

6.2 PROMISsim

We performed 62 simulations of PROMIS algorithm and obtained the computational results on a personal computer with Intel(R) Core(TM) i7-3537U CPU @ 2.00GHz and 8 GB RAM powered by Ubuntu Linux. First, we built 2 full scale free graph under the NGCE tool, with 2000 nodes and 5000 nodes. To finish a simulation with 2000 nodes, the time required is approximately 2 hours and for a simulation with 5000 nodes the time required is from 18 to 24 hours. The total time required to complete our experiments are about 350 hours.

Full Scale Free Graph	2000 Nodes	5000 Nodes
Rate of infection (β)	0.00141, 0.00241	0.00141, 0.00241
Number of P2P Nodes	400, 600	400, 1750
Incoming Size	50, 100, 200	200
Threshold Security Level (Lmax)	0.1, 0.4, 0.8, 1.5	0.1, 0.4, 0.8, 1.5
Number of Pre-infected Nodes	10	10
Maximum Iteration Number (t)	170	170
Time Required	2 hours	18-24 hours

Table 6. 4 Parameters for our simulations

The next diagrams are some typical simulation data results from the simulator's output and some of them are analyzed. In the end of the thesis, there is an Appendix with the total of our simulations.

Model type	<i>Full Scale Free Graph</i>
Environmental Parameters	
Birth Rate (β)	0.00141
Total Nodes	2000
Number of Pre-infected Nodes	10
System Parameters PROMIS	
Number of P2P Nodes	400
Maximum Iteration Number (t)	170
Threshold Security Level (Lmax)	0.80
Max number of communicating nodes (Cmax)	320
Minium number of communicating nodes (Cmin)	11%

Table 6. 5 Parameters of PROMISsim which simulations were performed

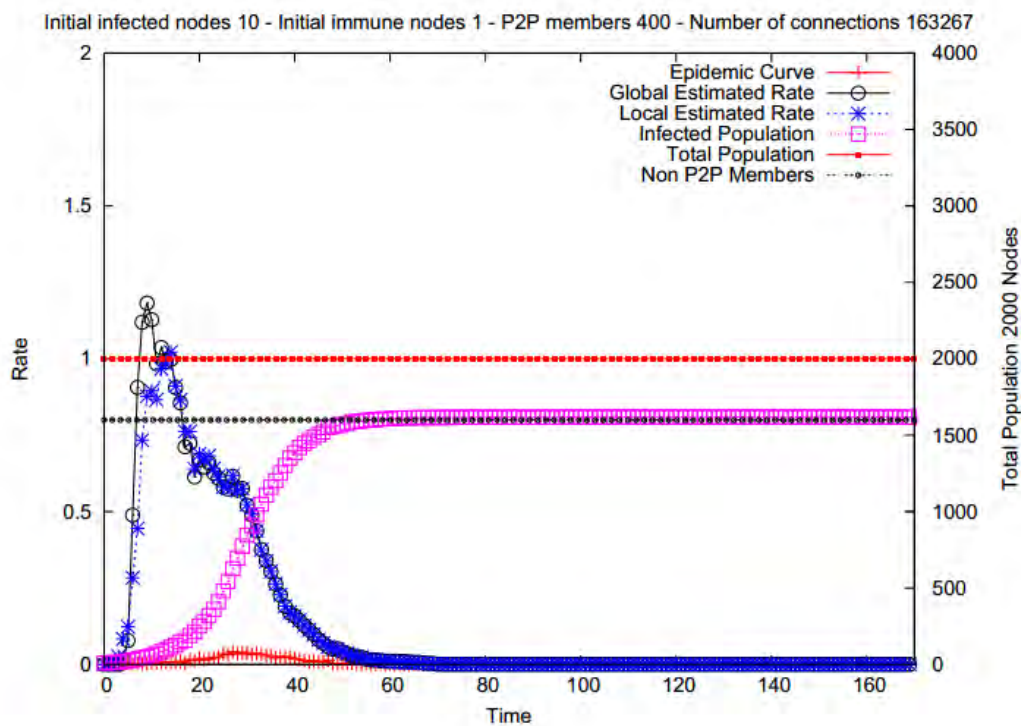


Figure 6. 15 Simulator's Output – Epidemiological Modeling of Malware Epidemic

Examining the diagram, it appears that each member of promisgroup understands significant outbreak of malicious activity in the early stages of an epidemic. Each node of the P2P network calculates, through PROMIS, the overall malicious activity. In this diagram, on a full scale free graph, each node realizes different the locally observed activity and the attendant overall malicious activity. The curves in the figure represent the average local malicious activity and overall malicious activity, as observed and calculated by all the members of the P2P network. The two horizontal lines distinguish the members of P2P network from other members of the experiment that did not participate in promisgroup. As is apparent from the graph, an important part of members of the P2P network promisgroup manages to avoid infection.

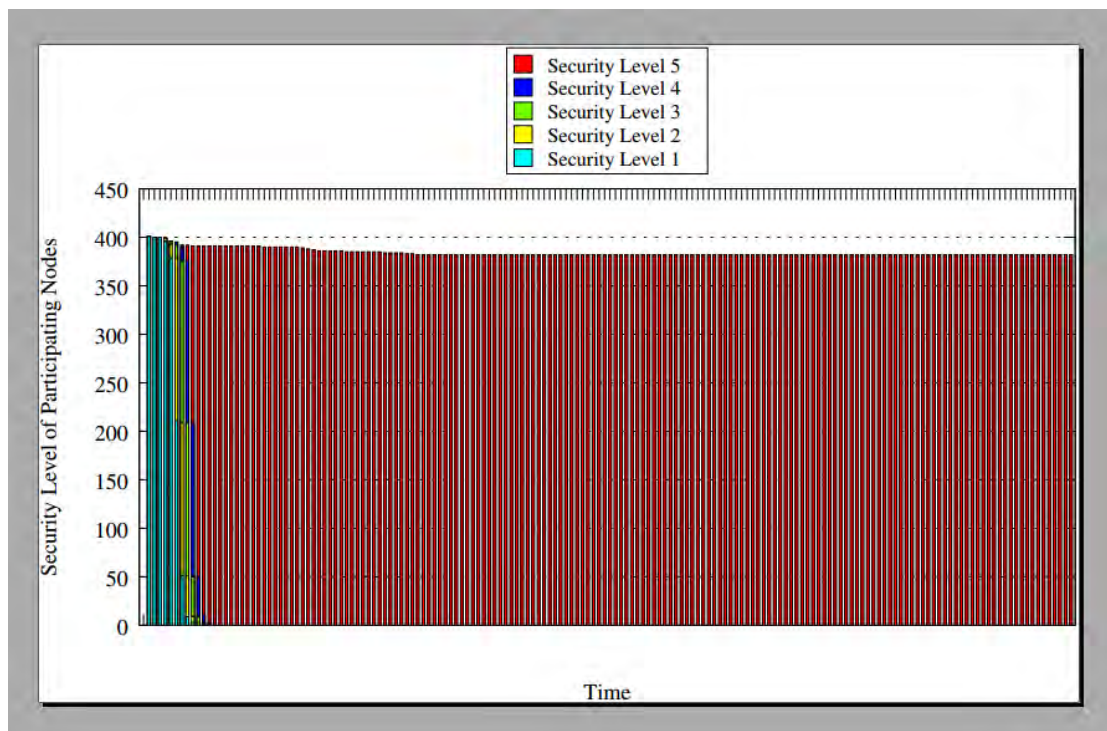


Figure 6. 16 Changes in security levels of nodes during the malware epidemic

At the end of the thesis, there is an appendix that have all the simulations and their data results gathered, for the best view it is recommended to view from the cd.

Model type	<i>Full Scale Free Graph</i>
Environmental Parameters	
Birth Rate (β)	0.00241
Total Nodes	2000
Number of Pre-infected Nodes	10
System Parameters PROMIS	
Number of P2P Nodes	600
Maximum Iteration Number (t)	170
Threshold Security Level (Lmax)	0.8
Max number of communicating nodes (Cmax)	320
Minium number of communicating nodes (Cmin)	11%

Table 6. 6 Parameters of PROMISsim which simulations were performed

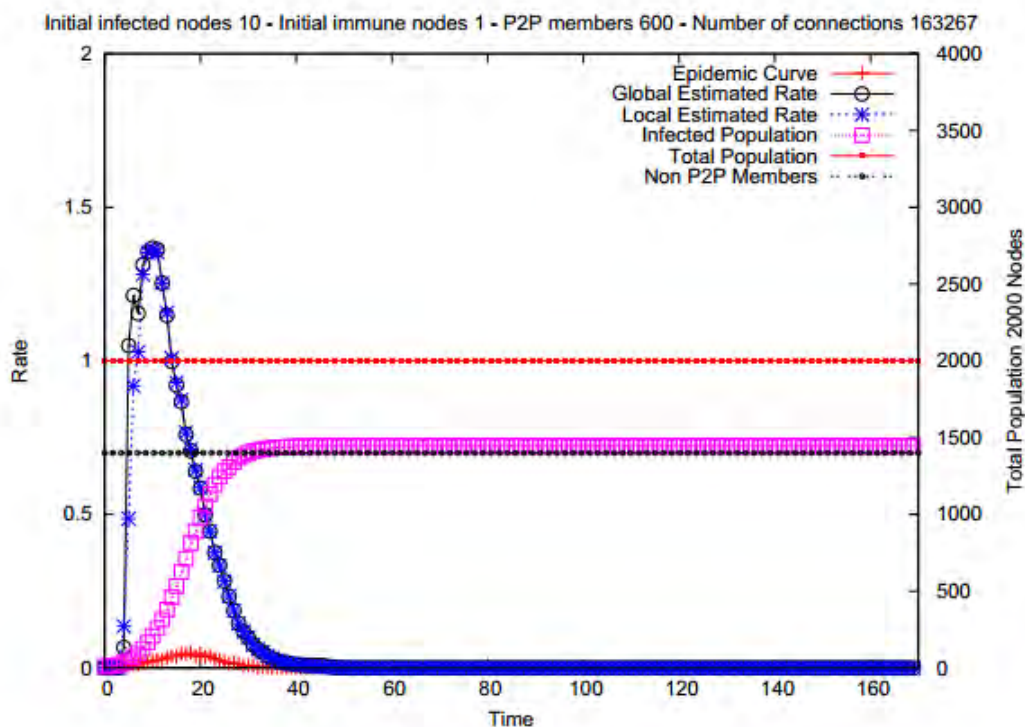


Figure 6. 17 Epidemiological modeling of 2000 Full Scale Free Graph with 0.00241 infection rate

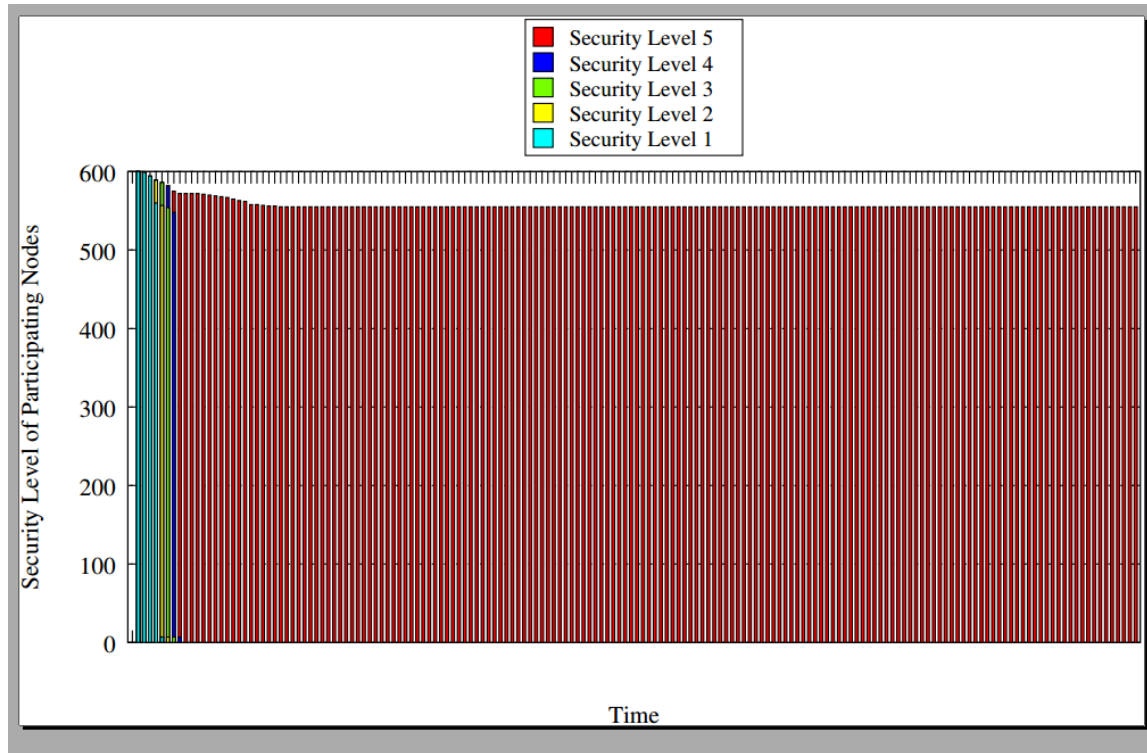


Figure 6. 18 Changes in security levels of nodes in Full Scale Free Graph with 2000 Nodes

As we see from the figures 6.17 and figure 6.18 PROMIS protects the users with a very sufficient way from the start of the outbreak. This configuration of simulation has more promisgroup users (600 nodes) and offers a very flexible security policy ($L_{max}=0.80$). In addition, the infection rate is 0.00241 which means it is a large load and PROMIS manages to operate in an optimal way to protect its users.

Model type	<i>Full Scale Free Graph</i>
Environmental Parameters	
Birth Rate (β)	0.00241
Total Nodes	5000
Number of Pre-infected Nodes	10
System Parameters PROMIS	
Number of P2P Nodes	1750
Maximum Iteration Number (t)	170
Threshold Security Level (Lmax)	0.10
Max number of communicating nodes (Cmax)	320
Minium number of communicating nodes (Cmin)	11%

Table 6. 7 Parameters of PROMISsim which simulations were performed

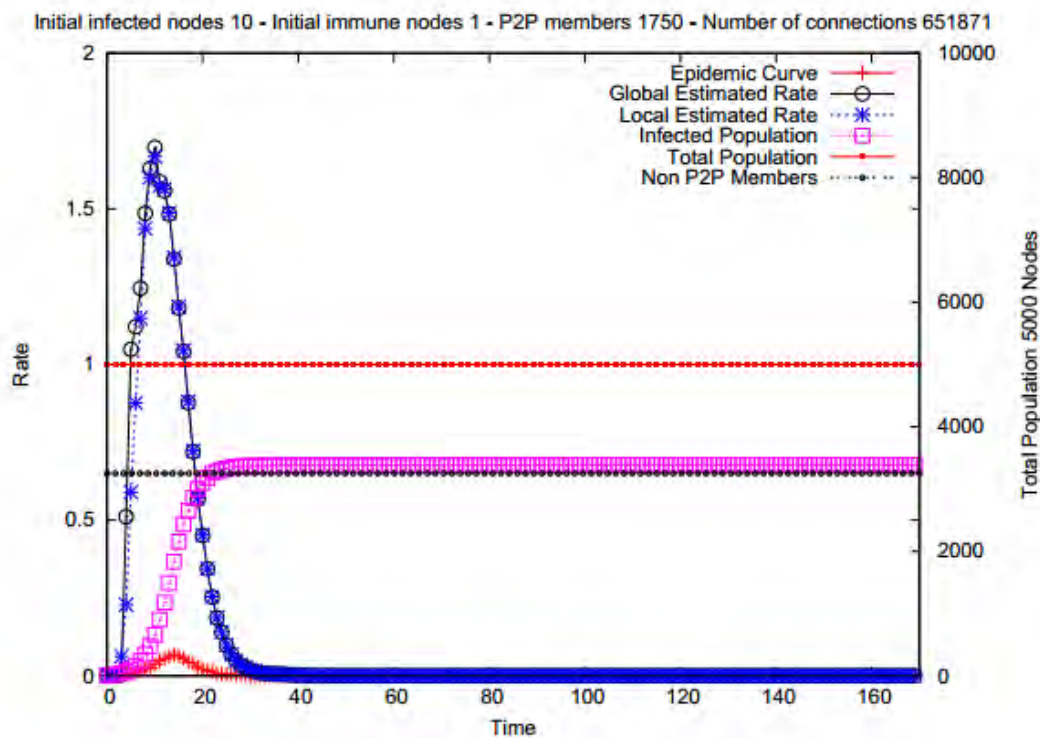


Figure 6. 19 Epidemiological modeling of 5000 Full Scale Free Graph with a very strict security policy (Lmax 0.10)

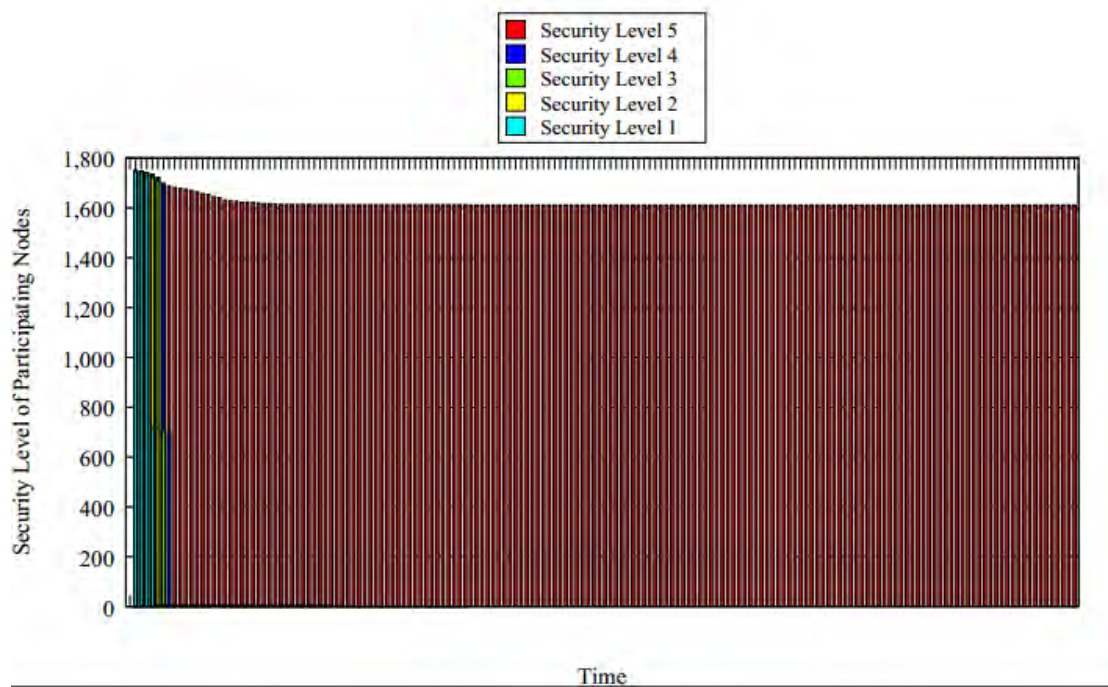


Figure 6. 20 Changes in security levels of nodes in Full Scale Free Graph with 5000 Nodes

In the figures 6.19 and 6.20 we can notice the changes in the security levels of the 1750 nodes of promisgroup users. In this simulation we have a bigger P2P network with 5000 users and the incoming size is 200 and that means that we have a lot of traffic. The security policy is very strict ($L_{max}=0.10$) and this offers maximum protection to the users of the promisgroup.

6.3 Threshold of Security Levels

The thresholds to increase or decrease the level of security are key factors that affect the performance of the PROMIS algorithm. As expected if the increase threshold of security level is very low, then the system can be hyper sensible even in cases where there is no reason at all. This phenomenon is called false positive recognition of threat and any security system attempts to minimize this. The false positive detection of an epidemic software will results in disrupting the non-critical system applications, in order to offer better protection to its users. The repeated appearance of false positive diagnoses with respect to alleged epidemics of malicious software is certain to cause significant discomfort to the user repeatedly interrupting him from his work, without any significant reason. Such a system is highly dysfunctional and maybe the users deactivate it. On the other hand, if the thresholds of security levels are too low, it is expected the existence of false negative diagnoses. The incorrect negative recognition existence of malicious epidemic software, will likely leave the protected system exposed the outbreak of the malicious software. The thresholds of security levels are therefore crucial for the proper successful operation of the PROMIS system. The experimental results that listed show that depending on the thresholds may be infected or survive almost the entire population of the P2P network.

Model type	<i>Full Scale Free Graph</i>
Environmental Parameters	
Birth Rate (β)	0.00141
Total Nodes	2000
Number of Pre-infected Nodes	10
System Parameters PROMIS	
Number of P2P Nodes	400
Maximum Iteration Number (t)	170
Threshold Security Level (Lmax)	-
Max number of communicating nodes (Cmax)	320
Minium number of communicating nodes (Cmin)	11%

Table 6. 8 Parameters of PROMISsim which simulations were performed

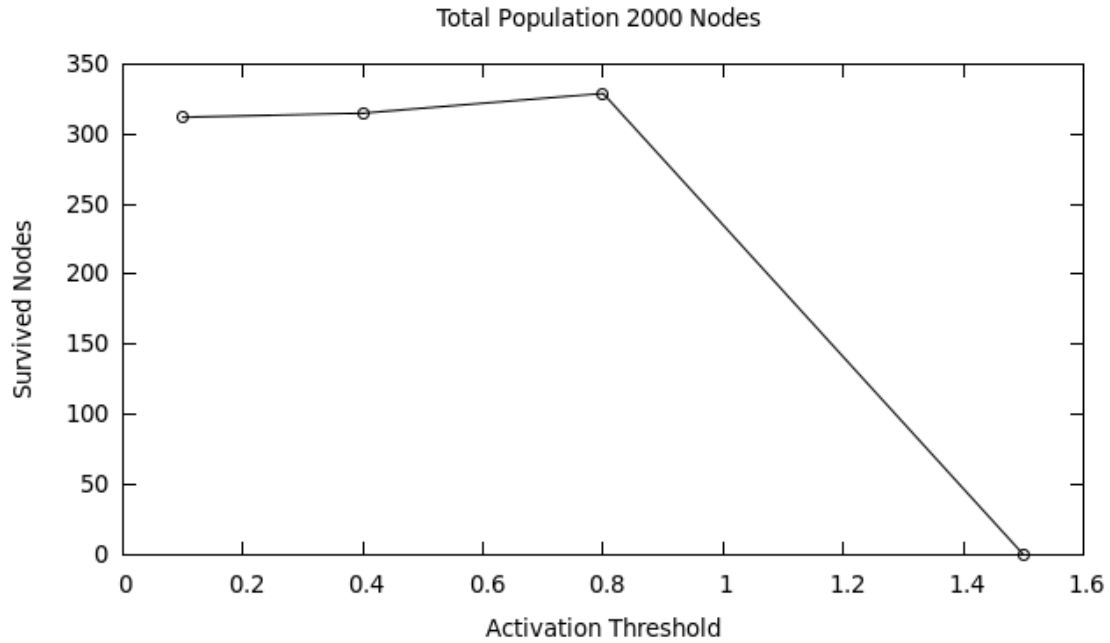


Figure 6. 21 Correlation between thresholds of security levels and the surviving nodes in a full scale free graph 2000 nodes

In this figure as we can notice the best value for optimal operation of PROMIS system is when Threshold is $L_{max} = 0.80$

Model type	<i>Full Scale Free Graph</i>
Environmental Parameters	
Birth Rate (β)	0.00241
Total Nodes	2000
Number of Pre-infected Nodes	10
System Parameters PROMIS	
Number of P2P Nodes	400
Maximum Iteration Number (t)	170
Threshold Security Level (L_{max})	-
Max number of communicating nodes (C_{max})	320
Minium number of communicating nodes (C_{min})	11%

Table 6. 9 Parameters of PROMISSim which simulations were performed

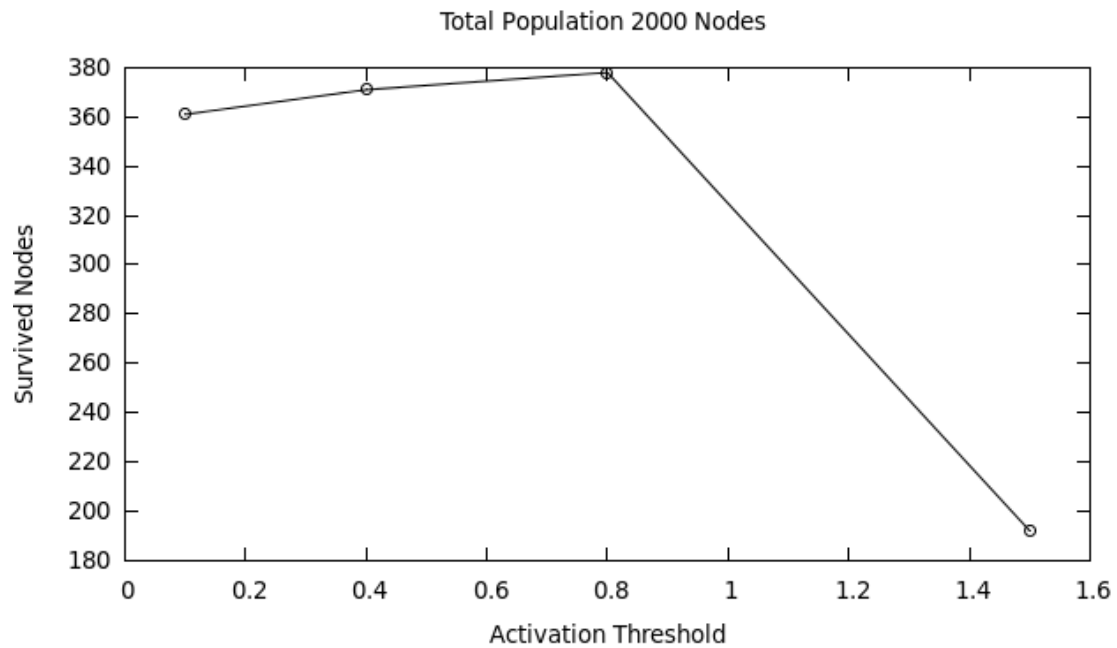


Figure 6. 22 Correlation between thresholds of security levels and the surviving nodes in a full scale free graph 2000 nodes

The figure above show us that the PROMIS system offers the optimal protection when the threshold is $L_{max} = 0.80$ and we have the most survived nodes.

Model type	<i>Full Scale Free Graph</i>
Environmental Parameters	
Birth Rate (β)	0.00141
Total Nodes	5000
Number of Pre-infected Nodes	10
System Parameters PROMIS	
Number of P2P Nodes	400
Maximum Iteration Number (t)	170
Threshold Security Level (L_{max})	-
Max number of communicating nodes (C_{max})	320
Minium number of communicating nodes (C_{min})	11%

Table 6. 10 Parameters of PROMISsim which simulations were performed

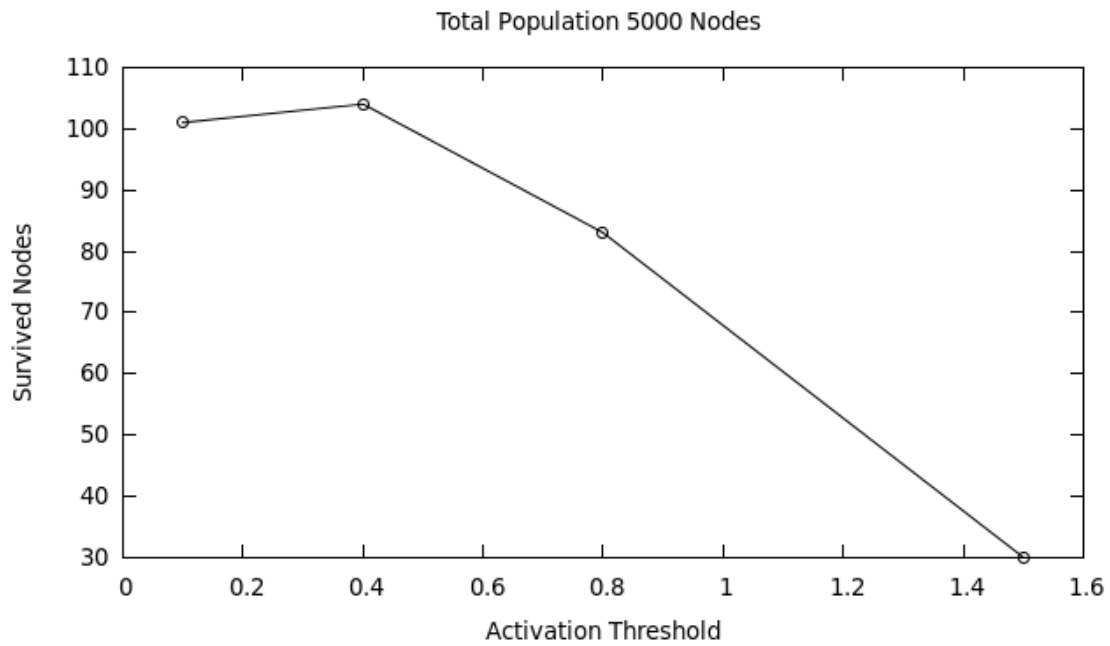


Figure 6. 23 Correlation between thresholds of security levels and the surviving nodes in a full scale free graph 5000 nodes

The optimal operation of the PROMIS algorithm is when we have configure the Lmax = 0.40 because we have the most survived nodes.

Model type	<i>Full Scale Free Graph</i>
Environmental Parameters	
Birth Rate (β)	0.00241
Total Nodes	5000
Number of Pre-infected Nodes	10
System Parameters PROMIS	
Number of P2P Nodes	1750
Maximum Iteration Number (t)	170
Threshold Security Level (Lmax)	-
Max number of communicating nodes (Cmax)	320
Minium number of communicating nodes (Cmin)	11%

Table 6. 11 Parameters of PROMISsim which simulations were performed

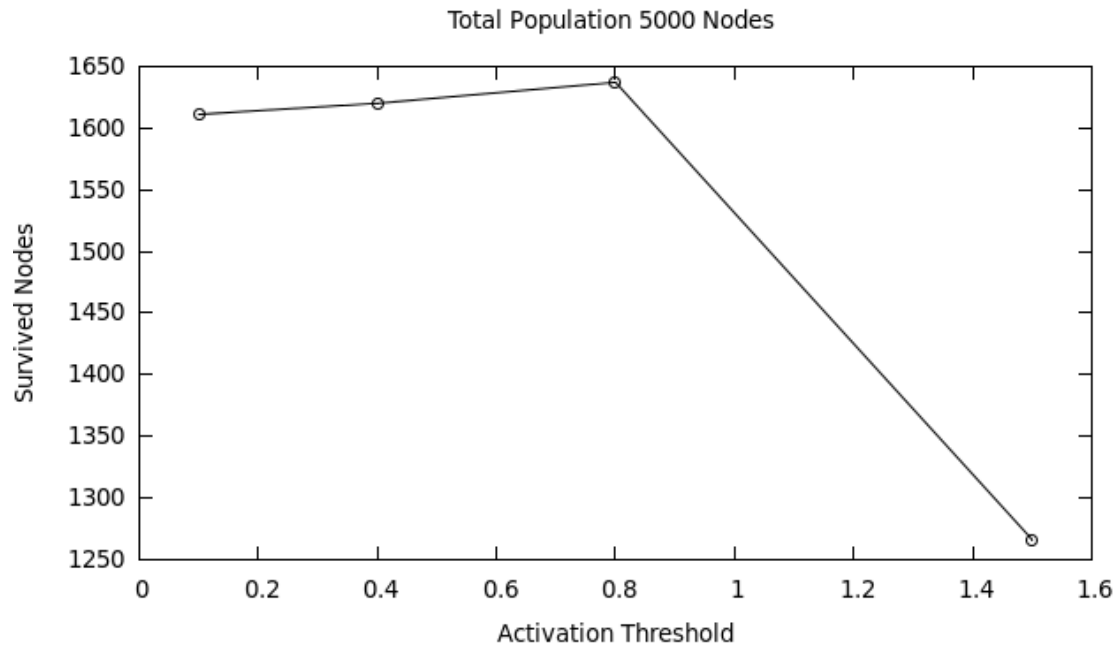


Figure 6. 24 Correlation between thresholds of security levels and the surviving nodes in a full scale free graph 5000 nodes

In this figure the most survived nodes, as we can see from the diagram is when the $L_{max} = 0.80$ and PROMIS system operates in the most optimal way to protect its users.

6.4 Maximum and minimum number of communicating nodes

In the early stages of a malware epidemic is reasonable to not become immediately noticeable the escalation of malicious activity of the PROMIS system, if it consists of a few members. Overcoming this issue may result from increased membership of promisgroup, but this means that will increase the volume of transferred data and may be scaling issues, since members of the P2P network communicate with each other constantly. To minimize the import load on the Internet, it was decided the members of the P2P network to communicate only when they have noticed a change in local malicious activity, reducing in this way the exchange of data, less burdening the operation of the Internet and allowing the further escalation of the system.

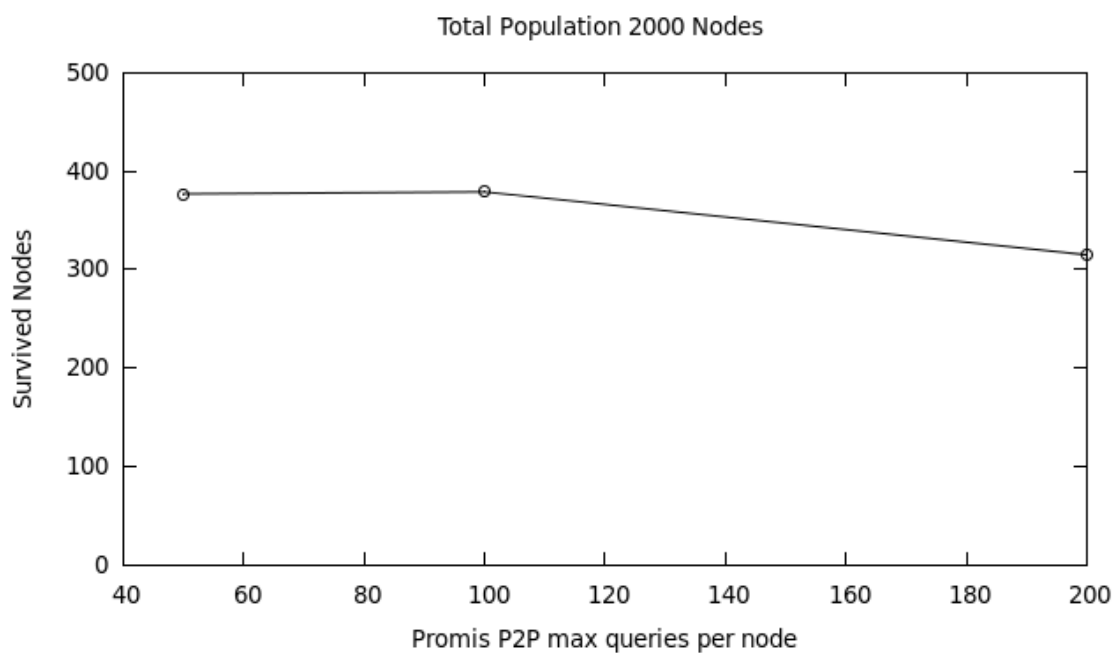


Figure 6. 25 Correlation between maximum - minimum number of communicating nodes and surviving nodes in the full scale free graph 2000 nodes

As we see from the above figure the algorithm the performance of the PROMIS algorithm decreases as the maximum number of communicating nodes increased.

Model type	<i>Full Scale Free Graph</i>
Environmental Parameters	
Birth Rate (β)	0.00141
Total Nodes	2000
Number of Pre-infected Nodes	10
System Parameters PROMIS	
Number of P2P Nodes	400
Maximum Iteration Number (t)	170
Threshold Security Level (Lmax)	0.80

Table 6. 12 Parameters of PROMISsim which simulations were performed

CHAPTER 7

The findings of the thesis are summarized in this chapter. They consist of evaluation, conclusions and observations, followed by suggestions for future research.

7.1 The aim of the Thesis

The aim of this thesis was to design and deploy a peer-to-peer network and with the use of epidemiological models to collect significant data results in order to optimize the performance of the PROMIS algorithm under a malware epidemic and in what degree it manages to protect its own members.

7.2 Recommendations for Future Research

A technique that could contribute to a more accurate and realistic simulation of the operation of the PROMIS algorithm laying down the threshold limits of security levels on each node separately. In this case, each member of the P2P network promisgroup maintains a distinct security policy, and behaves differently against potential threats. But in order for the results obtained have the necessary theoretical background, relevant research are needed to analyze the ways in which users can choose to configure the security policy of their systems and include the appropriate statistical details.

7.3 Conclusions

Simulation results indicate that the PROMIS algorithm is able to protect effectively a significant percentage of P2P network nodes, against malware epidemics. This percentage depends on the configuration of the PROMIS algorithm. Aggressive setting of these parameters leads to strict security policies that can ensure almost all the members of the peer network, while milder settings still protect satisfactory percentage of users.

REFERENCES

- [1] M. Erbschloe. Trojans, worms and spyware. A computer security professional's guide to malicious code. Elsevier Butterworth–Heineman, Oxford, UK, 2005.
- [2] National Infrastructure Advisory Council (NIAC). The national strategy to secure cyberspace. Technical report, U.S. Department of Homeland Security, February 2003.
- [3] B. Schneier. *Secrets & Lies. Digital Security in a Networked World*. Wiley Publishing Inc, IN, USA, 2000.
- [4] W. Wang, Y. Yuan, and N. Archer. A contextual framework for combating identity theft. *IEEE Security and Privacy*, 4(2):30–38, March 2006.
- [5] P. Barford and V. Yegneswaran. A look inside botnets. In Springer, editor, to appear in Series: Advances in Information Security, 2006.
- [6] The HoneyNet Project & Research Alliance. Know your enemy: Tracking botnets. <http://www.honeynet.org/papers/bots/>, March 2005.
- [7] M. Erbschloe. Trojans, worms and spyware. A computer security professional's guide to malicious code. Elsevier Butterworth–Heineman, Oxford, UK, 2005.
- [8] M. Vatis. Cyber-attacks: Protecting america's security against digital threats. Technical Report ESDP Discussion Paper ESDP200204, John F. Kennedy School of Government, Harvard University, June 2002.
- [9] National Infrastructure Protection Center. Cyber protests: The threat to the U.S. Information infrastructure. Technical report, October 2001.
- [10] S. Staniford, V. Paxson, and N. Weaver. How to Own the internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, pages 149–167, August 2002.
- [11] J. Nazario. Defense and Detection Strategies against Internet Worms. Artech House computer security series, 2004.
- [12] N. Weaver, V. Paxson, and S. Staniford. A worst-case worm. In *Proceedings of the Third Annual Workshop on Economics and Information Security (WEIS04)*, May 2004.

[13] S. Staniford, D. Moore, V. Paxson, and N. Weaver. The top speed of flash worms.

In *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*, pages 33–42, New York, NY, USA, 2004. ACM Press.

[14] N. Weaver, V. Paxson, S. Staniford, and Robert Cunningham. A taxonomy of computer worms. In *First Workshop on Rapid Malcode (WORM)*, 2003

[15] P. Szor. *The Art of Computer Virus Research and Defense*. Addison Wesley, Upper Saddle River, NJ, February 2005.

[16] J. Kephart, G. Sorkin, M. Swimmer, and S. White. Blueprint for a computer immune system. In *Proceedings of the Virus Bulletin International Conference*, San Francisco, California, October 1-3, 1997.

[17] S. Sna, J. Brentano, G. Dias, T. Goan, T. Heberlein, C. Ho, K. Levitt, B. Mukherjee, S. Smaha, T. Grance, D. Teal, and D. Mansur. DIDS (distributed intrusion detection system) - motivation, architecture, and an early prototype. In *Proceedings of the 14th National Computer Security Conference*, pages 167–176, Washington, DC, 1991.

[18] D. Moore, C. Shannon, G. Voelker, and S. Savage. Internet quarantine: requirements for containing self-propagating code. In *Proceedings of 22nd Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM 2003)*, April 2003.

[19] C. Zou, W. Gong, and D. Owsley. Code red worm propagation modeling and analysis. In *Proceedings of the 9th ACM Conference on Computer and Communication Security (CCS)*, Washington DC, USA, November 2002.

[20] D. Moore, C. Shannon, and J. Brown. Code-Red: a case study on the spread and victims of an internet worm. In *Proceedings of the Internet Measurement Workshop*, 2002.

[21] Eye Digital Security. Code Red II worm analysis AL20010804 Current on-line (June 2005): <http://www.eeye.com/html/research/advisories/al20010804.html>

- [22] D. Moore and C. Shannon. The spread of the code-red worm (crv2) Current online (June 2005): http://www.caida.org/analysis/security/codered/coderedv2_analysis.xml
- [23] A. Mackie, J. Roculan, R. Russell, and M. VanVelzen. Nimda worm analysis incident analysis report version ii. Technical report, Security Focus, September 2001.
- [24] NIMDA. Nomad worm/ virus report–final. Current on-line (January 2006): <http://www.securitymanagement.com/library/nimda1101.pdf>, October 2001.
- [25] F-secure virus descriptions: Nimda, Current on-line (June 2003): <http://www.fsecure.com/v-descs/nimda.shtml>.
- [26] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. *IEEE Security & Privacy*, pages 33–39, July 2003.
- [27] M. Bailey, E. Cooke, F. Jahanian, D. Watson, and J. Nazario. The blaster worm: Then and now. *IEEE Security & Privacy*, 3(4):26–31, July 2005.
- [28] C. Shannon and D. Moore. The spread of the witty worm. *IEEE Security & Privacy*, 2(4):46–50, July 2004.
- [29] S. Gritzalis. Enhancing web privacy and anonymity in the digital era. *Information Management and Computer Security*, 12(3):255–288, 2004.
- [30] S. Gritzalis and D. Spinellis. Addressing threats and security issues in World Wide Web technology. In *Proceedings CMS '97 3rd IFIP TC6/TC11 International joint working Conference on Communications and Multimedia Security*, pages 33–46. IFIP, Chapman & Hall, September 1997.
- [31] A. Doumas, K. Mavrouidakis, D. Gritzalis, and S. Katsikas. Design of a neural network for recognition and classification of computer viruses. *Computers & Security*, 14(5):435–448, 1995.
- [32] S. Katsikas, T. Spyrou, D. Gritzalis, and J. Darzentas. Model for network behaviour under viral attack. *Computer Communications*, 19(2):124–132, 1996.
- [33] Bollobás, B. *Random Graphs*. London: Academic Press, 1985.
- [34] Law, R., and Dieckmann, U. Symbiosis through exploitation and the merger of lineages in evolution. *Proc. R. Soc. London (B)* 265: 1-9, 1998.

- [35] Barbour, A.D. & Mollison, D. Epidemics and random graphs. In: Stochastic processes in epidemic theory (J.P. Gabriel, C. Lefevre & P. Picard, Eds), *Lecture Notes in Biomathematics* 86: 86-89, Springer, 1990.
- [36] Eames and Keeling, Modeling dynamic and network heterogeneities in the spread of sexually transmitted diseases, *PNAS* 99 13330-13335, 2002.
- [37] Read and Keeling, Disease evolution on networks: the role of contact structure *Proc. Roy. Soc. Lond. B* 270 699-708, 2003.
- [38] R. Albert, H. Jeong, and Barabasi. Diameter of the world-wide web. *Nature*, 401:130–131, September 1999.
- [39] R. Albert and A. Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, January 2002.
- [40] Liljeros, F., Edling, C. R., Amaral, L. A. N., Stanley, H. E., and Aberg, Y. The web of human sexual contacts. *Nature* 411(6840), 907–908, 2001.
- [41] Frank, O.; Strauss, D., "Markov Graphs". *Journal of the American Statistical Association* 81: 832–842., 1986
- [42] W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. In *Proceedings of the Royal Society of London. Series A*, volume 115, pages 700–721, 1927.
- [43] M. Cai, K. Hwang, Y. Kwok, S. Song, and Y. Chen. Collaborative internet worm containment. *IEEE Security & Privacy*, 3(3):25–33, May 2005.
- [44] A. Kalafut, A. Acharya, and M. Gupta. A study of malware in peer-to-peer networks. In *Internet Measurement Conference*, 2006.
- [45] S. Shin, J. Jung, and H. Balakrishnan. Malware prevalence in the kazaa file-sharing network. In *Internet Measurement Conference*, 2006.
- [46] R. Arora. Detecting worms through de-centralized monitoring. <http://homepages.cae.wisc.edu/raman/Projects/WormDet.pdf>, May 2004.
- [47] M. Locasto, J. Parekh, A. Keromytis, and S. Stolfo. Towards collaborative security and p2p intrusion detection. In *Systems, Man and Cybernetics (SMC)*

Information Assurance Workshop, 2005. *Proceedings from the Sixth Annual IEEE*, pages 333–339, June 2005.

[48] R. Janakiraman, M. Waldvogel, and Q. Zhang. Indra: A peer-to-peer approach to network intrusion detection and prevention. In *Proceedings of 2003 IEEE WET ICE Workshop on Enterprise Security*, Linz, Austria, June 2003.

[49] K. Anagnostakis, M. Greenwald, S. Ioannidis, and A. Keromytis. Robust reactions to potential day-zero worms through cooperation and validation. To appear in the *Springer International Journal of Information Security (IJIS)*, ISC'06 Special Issue, 2007.

[50] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In *OSDI*, pages 45–60, 2004.

[51] P. Chhabra, A. John, and H. Saran. PISA: Automatic extraction of traffic signatures. *Lecture Notes in Computer Science*, 3462:730–742, May 2005.

[52] T. Toth and C. Kruegel. Connection-history based anomaly detection. 2002.

[53] S. Cheetancheri, J. Agosta, D. Dash, K. Levitt, J. Rowe, and E. Schooler. A distributed host-based worm detection system. In *SIGCOMM'06 Workshop, Pisa, Italy*, September 2006. ACM.

[54] DShield. Distributed intrusion detection system. : Current on line (december 2006): <http://www.dshield.org/>.

[55] DeepSight. Symantec DeepSight Threat Management System current on-line (november 2007): <http://tms.symantec.com/>

[56] J. Yang, P. Ning, X. Wang, and S. Jajodia. CARDS: A distributed system for detecting coordinated attacks. In *Proceedings of the IFIP TC11 Fifteenth Annual Working Conference on Information Security for Global Information Infrastructures*, pages 171–180, Deventer, The Netherlands, The Netherlands, 2000. Kluwer, B.V.

[57] J. B. Michael, M. Auguston, N. Rowe, and R. Riehle. Software decoys: Intrusion detection and countermeasures. In *Proceedings of the 2002 IEEE Workshop on Information Assurance*, pages 130–138, West Point, NY, USA, June 2002. United States Military Academy.

- [58] F. Raynal, Y. Berthier, P. Biondi, and D. Kaminsky. Honeypots forensics, part ii: Analyzing the compromised host. *IEEE Security & Privacy*, 2(5):77–80, September 2004.
- [59] P. Reiher, J. Li, and G. Kuenning. Midgard worms: Sudden nasty surprises from a large resilient zombie army. Technical Report CLA -CSD-040019, UCLA Computer Science Department, April 2004.
- [60] N. Ye and T. Farley. A scientific approach to cyber-attack detection. *IEEE Computer*, 38(11):55–61, November 2005.
- [61] M. Webster and G. Malcolm. Detection of metamorphic computer viruses using algebraic specification. *Journal in Computer Virology*, 2(3):149–161, December 2006.
- [62] I. Yoo and U. Ultes-Nitsche. Non-signature based virus detection. Towards establishing an unknown virus detection technique using SOM. *Journal in Computer Virology*, 2(3):163–186, December 2006.
- [63] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the twelfth international conference on World Wide Web*, pages 640–651. ACM Press, 2003.
- [64] S. Shin, J. Jung, and H. Balakrishnan. Malware prevalence in the kazaa file-sharing network. In *Internet Measurement Conference*, 2006
- [65] W. Yu, C. Boyer, and D. Xuan. Analyzing impacts of peer-to-peer systems on propagation of active worm attacks. Technical report, Computer Science Dept., Texas A&M Univ, 2004.
- [66] J. Kephart and S. White. Directed-graph epidemiological models of computer viruses. In *Proceedings of the 1991 Computer Society Symposium on Research in Security and Privacy*, California USA, pages 343–361, 1991.
- [67] J. Kephart. How topology affects population dynamics. In *Proceedings of Artificial Life 3*, New Mexico, USA, June 1992.
- [68] Kephart and S. White. Measuring and modeling computer virus prevalence. In *Proceedings of the 1999 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 2–14, Oakland, California, May 1999.

- [69] Kephart, D. Chess, and S. White. Computers and epidemiology. *IEEE Spectrum*, 30(20), May 1993.
- [70] V. Vlachos, A. Raptis, and D. Spinellis. PROMISing steps towards computer hygiene. In Steven Furnel, editor, International Network Conference (INC2006), pages 229–236, Plymouth, UK, July 2006.
- [71] A. Law and W. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, third edition, 2000.
- [72] R. Bagni, R. Berchi, and P. Carello. A comparison of simulation models allied to epidemics. *Journal of Artificial Societies and Social Simulation*, 5(3), June 2002.
- [74] Kermack, W; Kendrick, "Contributions to the mathematical theory of epidemics—II. The problem of endemicity". *Bulletin of Mathematical Biology* 53 (1–2), 1991.
- [75] D. Daley and J. Gani. *Epidemic Modelling*. Cambridge University Press, Cambridge, UK, 1999.
- [76] E. Filiol, M. Helenius, and S. Zanero, "Open problems in computer virology," *Journal in Computer Virology*, vol. 1, no. 3, pp. 55–66, 2006.
- [77] J. Aycock, *Computer viruses and malware*. Springer Publishing Company, Incorporated, 2010.
- [78] P. Szor, *The art of computer virus research and defense*. Addison-Wesley Professional, 2005.
- [79] M. Ludwig, *The giant black book of computer viruses*. American Eagle Publications, 1998.
- [80] V. Vlachos, V. Vouzi, D. Chatziantoniou, and D. Spinellis. NGCE – Network Graphs for Computer Epidemiologists. In Panagiotis Bozanis and Elias N. Houstis, editors, *In Advances in Informatics: 10th Panhellenic Conference on Informatics, PCI 2005*, Lecture Notes in Computer Science 3746, pages 672–683. Springer-Verlag, November 2005.
- [81] L. Li. *Java Data Structures and Programming*. Springer-Verlag, Berlin, DE, 1998.

- [82] V. Vlachos, and D. Spinellis, A PProactive Malware Identification System based on the Computer Hygiene Principles, *Information Management & Computer Security (Emerald)*, August 2007
- [83] A. Barabasi, R. Albert, and H. Jeong. Mean-field theory for scale-free random networks. *Physica A*, 272:173–187, 1999.
- [84] S. Virtanen. Properties of nonuniform random graph models. Research Report HUT-TCS-A77, Helsinki University of Technology, Laboratory for Theoretical Computer Science, 2003.
- [85] <http://www.ossec.net/>
- [86] <http://www.snort.org/>
- [87] <http://suricata-ids.org/>
- [88] <http://www.sourcefire.com/>
- [89] <http://www.cisco.com/c/en/us/products/security/intrusion-prevention-system-ips/index.html>

APPENDIX