Crowdsourcing intelligence in security applications

CHRISTOFORIDIS CHRISTOFOROS

Master of Science in Network Computing

T.E.I. of LARISSA STAFFORDSHIRE UNIVERSITY

TECHNOLOGICAL EDUCATIONAL INSTITUTE OF LARISA

June 2014

TECHNOLOGICAL EDUCATIONAL INSTITUTE OF LARISA

Crowdsourcing intelligence in security applications

CHRISTOFORIDIS CHRISTOFOROS

Master of Science in Network Computing, 2014

Thesis Summary

Along with the Internet breakthrough, on November 2, 1988, Robert Tappan Morris unleased what became known as the Morris worm. Besides other known malware at that time, Morris worm managed to disrupt 6000 computer systems or else the one tenth of all computer systems connected to the Internet. Almost 26 years later worms are considered to be the most dangerous computers threats of all time. This thesis presents a survey on malware with emphasis on worms, some history, worm characteristics, related work on malware identification and protection. It further looks into peer-to-peer technology and JXTA peer-to-peer framework, Windows operation system specific modifications and vulnerabilities, and perform a deeper analysis on security and available security features that might support future JXTA applications. Ultimately an alternative collaborative security application is provided, which is built using Java and JXTA framework and specializing on worms. The final application uses the experiences of the many in a peer-to-peer network to detect and most probably avoid infections from worm attacks. The outcome of the Thesis could be used to inspire a new open source community, to support it with new ideas and allow it for further distribution among internet users.

Keywords: worm protection, JXTA, peer-to-peer, windows

Acknowledgements

In life knowledge enriches our existence with additional opportunities and choices waiting to be grabbed. Everyone's scope of seeing and perceiving things and therefore interacting with the World's entities is proportional to its experiences and knowledge. Some believe that living is more important than trading hours for knowledge. I believe that what doesn't kill you makes you stronger and more capable. Everything is knowledge after all.

For this journey in learning I would like to dedicate this thesis to my family for their constant love and support during difficult times in my life. Without them there would be nothing. I would like to express my sincere gratitude to my supervisor Dr Vasileios Vlachos for his support, motivation and enthusiasm during my MSc thesis study and research. I would also like to thank Johnny my friend for his help in testing and discussing parts of this thesis.

I would like to express my apologies to people i neglected during the composing of this thesis, a matter which i will try to redress upon finishing. Also, last but not least I would like to thank God for the everyday lessons He gives me, plus his support and guidance so as to be able to come along.

Contents

List of Figures

List of Tables

List of Symbols / Abbreviations

IDN	Intrusion Detection Network
IDS	Intrusion Detection System
P2P	Peer-to-Peer
S-I-R	Susceptible - Infectious - Removed
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
USA	United States of America
IIS	Internet Information Services
PC	Personal Computer
SQL	Structured Query Language
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
IP	Internet Protocol
LAN	Local Area Network
ICMP	Internet Control Message Protocol
NVD	National Vulnerability Database
NIST	National Institute of Standards and Technology
OSVDB	Open Source Vulnerability Database
CVE	Common Vulnerabilities and Exposure
LCS	Longest Common Substring
DHT	Distributed Hash Table
ABDIAS	Agent-based Distributed alert system
HBCIDS	Host-based Collaborative IDS
CDDHT	Cyber Disease Distributed Hash Table
SFC	Sensor Fusion Centers
URL	Uniform Resource Locator
FFCIDN	Fastflux service networks
XML	Extensible Markup Language
NAT	Network Address Translation
URN	Uniform Resource Name
ID	Identifier
QoS	Quality of Service
TTL	Time-to-live
PDA	Personal Digital Assistant
PRP	Peer Resolver Protocol
PDP	Peer Discovery Protocol
PIP	Peer Information Protocol

PBP	Pipe Binding Protocol
ERP	Endpoint Routing Protocol
RVP	Rendezvous Protocol
SRDI	Shared Resource Distributed Index
RPV	Rendezvous peerview
AES	Advanced Encryption Standard
MAC	Media Access Control
CPS	Certificate Practice Statement
CA	Certificate Authority
CRL	Certificate Revocation List
PSE	Personal Security Environment
SSL	Secure Socket Layer
TLS	Transport Security Layer
FTP	File Transfer Protocol
DSA	Digital Signature Algorithm
JKS	Java Keystore
DES	Data Encryption Standard
PKIX	Public-Key Infrastructure X.509
CBJX	Crypto-Based JXTA
PKCS	Public-Key Cryptography Standard
CBID	Crypto-Based Identifier
IDE	Integrated Development Environment
RDV	Rendezvous
RLE	Relay

Chapter 1

Introduction

1.1 Motivation

Malware pose a threat since the first constructed computer viruses from students undergoing their postgraduate studies. Their software was continuously replicating, consuming resources, and ultimately preventing host systems from functioning (Rabbit viruses). Today's computer world constantly suffers from new malware attacking and employs state of the art protection systems to defend against them. The biggest threats of all times, the Worm malware use the Internet to be able to spread as fast as possible to as many hosts as possible. When a new worm spread is identified, usually it is too late, and will most probably be followed by huge economic losses and important safety breaches in the computer world industry.

Antivirus and antimalware programs with signature analysis mechanisms on security files and/or on running procedures search for malware code that matches up to a point known prototypes stored on their appended databases. Heuristic algorithms that analyze the attitude of the programs (behavior-based analysis and algorithm-based analysis) [1] are trying to identify suspicious code sequences. Intrusion detection systems are trying to identify possible intrusions based on Internet traffic and suspend it. Although these and new approaches follow sophisticated techniques against Worms, they are not enough.

Internet worms most of the times are using simple techniques based more on social engineering, rather than computer systems intrusion mechanisms in order to

distract their victims. They are also targeting system vulnerabilities, in other words software weak points of entry or else backdoors that will allow them access to the system. If creating completely safe software is not an easy task so it is the protection from Internet worms. In addition, time is against the defenders. The time needed to update a system with the latest security software signatures and software patches is far greater than the 15 minutes a Warhol worm needs or the 1 minute a Flash [2, 3] worm requires to infect all of its targets.

No matter what the name is, if it is email attack malware like Melissa and ILoveYou [4, 5], server jamming like Code Red, Code Red 2[6, 7], multi-vector like Nimda [8], or other like Slammer [9], Blaster [10], Witty [11], Storm [12] or Conficker [13], etc. there will always be fast and their result will always translate to money losses, decreased reliability and anxiety towards digital communications. History has tough us that the consequences are tremendous when unprepared. Today computers are everywhere and our lives are influenced by them. For those reasons, and because there is no perfect system, ideas and mechanisms to shield against worms are now needed more than ever.

Because worms use the Internet to spread, a key point to have greater chances of immunizing hosts is collaborative intelligence. In other words, information such as e.g. shared probability of attack between peers on a common peer-to-peer network. Peer-to-peer approaches against centralized ones seam more promising and faster because each peer can have both client and server roles. Worms also use peer-to-peer technology networks to spread (e.g. VBS.Gnutella, Worm.P2P.Palevo.DP).

This thesis will provide a peer-to-peer (P2P) security system that based on traffic sequence (e.g. an IDS log file), a common worm attacking phenomenon will decide whether to enforce protection measures (close services, disable internet browser add-ons and restrict browsing experience) that will make the system less vulnerable with less chances to be infected. In more detail, there will be clients running on host machines forming a common P2P network that will exchange attack probability information based on a certain formula and calculate the average probability information from the messages received in order to enforce or withdraw measures according to certain threshold values.

1.2 Aims of the Thesis

1. A worm defender program based on an alternative way of defending [14] will be provided to enhance fighting against those most dangerous threats.
    a. JXTA latest edition peer-to-peer open source framework will be investigated, used and analyzed.
    b. Java programming language will be used which is open source, widely supported and used (computers, mobile devices, etc.). This means that the program will require minor modifications (e.g. on the measures part, stop Linux daemons etc.) to be able to run for Linux distributions.
    c. Windows probable security weak points of software will be looked into.
    d. Measures will be investigated and taken to avoid probable weak points that may compromise the Windows operating system.
        i. Registry modifications
        ii. Services to be stopped
2. A framework for evaluation of the performance of the end system will be built.
3. Clear statements of any constraints and restrictions of the worm security application will be made.
4. All in all, the target is to acquire the knowledge and skills around that subject and develop a piece of software using current tools that will be as close to ready as it can be for public use.

1.3 Research Methodology

1.3.1 Literature Review

The literature related to this Thesis includes malware, worm, security, windows registry, epidemiology, P2P, JXTA, related papers and books. Most of these are available online (Springerlink, ACM, IEEE) and can be accessed through the computing facilities of TEI of Larisa and the VPN account that is provided. Google Scholar and Microsoft Academic Search tools can also be used to assist in finding

suitable literature. There is also a library in the facilities of TEI, where useful resources can be found.

1.3.2 Analysis and Investigation

This thesis can be divided in three parts. The first part is a theoretical analysis around worms. The second part is a theoretical analysis of P2P, JXTA and the security measures. The third part, is a programming approach towards a scheme that will provide protection against worms.

Initially, malware in general is investigated, to have a first and broad view of the existing software threats in the digital world. Later, we focus on a certain category of malware named worms. History events regarding worm infections is surveyed and recalled. The purpose is to emphasize on how serious worm outbreaks can be. In addition worm characteristics are surveyed to acquire the knowledge of how to secure against them. It is obvious that knowing your enemy can assist to its confrontation. The significance of worm epidemic modeling is also important to that scope. S-I-R worm epidemic model is analyzed which is used by most of the worm modeling approaches out there. Such techniques are used either after an outbreak to determine what led to that result or for probable new such threats to determine their spread extent and consequences. Until now it is obvious that worms are highly infectious and defending against them is not an easy task. For that reason the need to analyze worms and their impact to digital world from the perspective of other scientific fields (e.g. biology) is also important. Furthermore besides local protection using antivirus and firewall programs, it would be more efficient to cooperate and share information in a network of peers (crowdsourcing intelligence). The key requirements of such systems are analyzed along with their necessary components. In addition related work is also presented.

A broad knowledge around worms is required to go even further and build a security application to defend against worm attacks. A classification of P2P networks is presented and the advantages of choosing peer-to-peer implementation in our application. A theoretical analysis on JXTA peer-to-peer framework, how it works, the protocols and services, JXTA security and implementation issues and details are also analyzed. The JXTA and Java APIs were studied, to determine what functions are

available and how these could be used in the best possible way for implementing the project. Also investigation was made for measures that could be enforced to Windows operating system without negatively influencing their good working order and/or striping important features. Interesting information was found on creating batch files that could run alongside with dos code, both called from inside the application using java code. In addition information is available regarding Windows services and their applications.

In order to achieve the goals of this Thesis, there are various topics that need to be studied in detail, which include:

- the Java programming language and its libraries that are of possible use,
- the JXTA P2P framework and its libraries,
- sqlite database,
- Windows operating system services, batch script files and DOS commands

1.3.3 Prototyping

The developed system uses current technologies, is based on [14] and is built in Java, using Java JXTA peer-to-peer framework libraries as a base of creating the P2P networks that the application will be based on. The program reads the inputs from a text file that saves information representing traffic to the host. The percentage of traffic increase or decrease is calculated according to a formula and sent to peers on the P2P network. Every peer calculates the average of the its received messages. If the average exceeds a certain threshold the application enforces measures (close services, disable browser add-ons, registry modifications) on the Windows operating system to protect it from getting infected. These measures limit the system to a constrained working order, closing services and allowing browsing using some of the available resources. If the sequence of the dropped packets is decreased under a certain threshold then the application returns to its previous state. The whole operation of the application is based on unusual, unexpected traffic increase to peers of the P2P network that is most probably caused by a worm scanning for targets mechanism. The peers communicate this knowledge and quickly take measure to evade infection.

1.3.4 System Evaluation

The developed system is tested on user machines across the Internet to verify its good working order. Messages between users are successfully exchanged and peers discover other peers and their advertisements. These are also confirmed via the testing framework. In addition, upon planting fake inputs to text files to simulate a probable worm outbreak, the measures are enforced successfully to the Windows operating systems. Again when a low threshold is reached the systems are restored to their previous state respectively. The time required for message exchanges in the low scale tests suffices and the application seems to be stable for long period of time, working without problems.

1.4 Novel Features of the Thesis

Worm attacks are evolving and have received much attention in recent years. Existing implementations are intricate, suffering from the client/server model, with scalability and fault tolerance issues, either constrained to LANs or not, even if applying layering techniques or other such techniques, they lack of potentials if not using the P2P model. Our application differs, is simple, easily expendable to include additional features and uses the powerful peer-to-peer technology. Although it focuses on worms that raise network traffic, those represent the majority of worms. It is built in Java which is a widely supported programming language and makes it easy to be transferred to other operating systems. Open source JXTA peer-to-peer framework is also used which is a powerful tool when building peer-to-peer networks. The application provides visualization of both the local percentage rate, and the combined knowledge coming from messages representing other peer experiences on the peer-to-peer network. It leverages peer-to-peer crowdsourcing intelligence or else the knowledge of a peer-to-peer network of peers communicating together and aims to avoid infections coming from software imperfections – vulnerabilities that are yet a major issue as far as security is concerned. It requires a minimum amount of time, which is very important on worm outbreaks and takes measures immediately until the threat is over and further measures are taken (e.g. updates – patches).

1.5 Outline of the Thesis

The thesis is organized as follows: Chapter 2 provides the necessary background on malware with emphasis on worms, worm characteristics, spread modeling, and related work. Chapter 3 provides the necessary background theory on P2P and JXTA P2P framework. It begins with a classification of peer-to-peer networks and their advantages, and continues with theoretical analysis of JXTA, JXTA framework security, and security issues. Chapter 4 presents the implementation details, design and analysis of the developed system. Chapter 5 consists of the evaluation, conclusions and observations, followed by suggestions for future work.

Chapter 2
Background

2.1 Malware

Malware term comes from two words malicious and software. The term refers to programs whose purpose is to violate computer security and cause damage in any form. Malware can be Viruses, Worms, Trojans, and Spyware. This thesis focuses on certain types of malware called Internet Worms. Worms are self-replicating malware that use computer networks to spread and exploit widely-used software [15,1]. Additional threats can be: Backdoor is a piece of code in an application programmed by its author to give access to the system no matter the circumstances, either for testing or for future use purposes [15,1]. Trapdoor is modifications on applications done by intruders to allow access after certain action sequences [15,1]. Logic bombs are malware activating when certain logic conditions are met.[15,1]. Time bombs are malware activating when certain time conditions are met [15,1]. Trojan horses [15,1] are software applications that run both a visible legal operation and an illegal hidden one. Rabbits or Bacteria are the continuously replicating programs that consume all host system resources. Bots [16,17] are a malware category equipped with a variety of functions such as communicating via Mirc protocol[16] or P2P, they may also use Http sequences of commands in order to evade detection and also can be upgraded to the host system[18]. Rootkits [19] are using low-level code functions to be detection difficult. There can be device drivers or kernel modules. Ransomware [20] uses cryptography to encrypt user data and demands ransom to decipher. Scareware [21] is a fraud malware known also as fraudware. It appears most of the times as pop-up

window to scare users in order to buy some other software that is supposed to protect their system. Many of the current threats are hybrids of the upper types of malware.

History stone craved moments of confusion due to worm outbreaks feature the epidemic worm called Morris at 1988, built by Robert Tappan Morris. Using state of the art techniques the Morris worm [22] managed to infect around 6000 systems or else 10% of the Internet. It also revealed security holes on USA's technical infrastructures on a time that the Cold War was in progress. 2001 Code Red [23] worm in 6 days time managed to infect 359.000 hosts at random attacking computers running Microsoft's IIS web server. A completely new worm Code Red 2[23] released two weeks after targeting again Microsoft's IIS web server software. This time it had no attack function but backdoor in order to allow attacks and targeting machines on the same subnet as the host infected one. Although the patch was available quickly, not everyone had patched their servers, including Microsoft themselves. Same year Nimda [24] worm affected workstations running Windows 95, 98, Me, NT, XP and servers running Windows NT and 2000. Nimda unlike the previous worms used five different infection vectors (email, open network shares, browsing of compromised web sites, exploitation of various Microsoft IIS 4/5 directory traversal vulnerabilities, backdoors left from previous worms). It was the most widespread worm within 22 minutes infecting in a 24-hour period 2.2 million servers and PCs and causing in 1 day 531 million dollars of losses due to downtime and clean-ups. 2003 SQL Slammer [25] caused denial of service on some internet hosts and managed to infect 75000 victims within 10 minutes. It exploited a buffer overflow vulnerability to random hosts running unpatched copies of Microsoft SQL server Resolution Service. These infected hosts became spaying the Internet with more copies of the worm. Same year Blaster worm infected systems running Windows XP and Windows 2000 in random causing restarts. 2004 Witty worm [26] managed to infect 12.000 computers in half an hour targeting firewalls and other security products written by a particular company IBM Internet Security Systems. Once infecting the host Witty worm launched attacks generating UDP traffic and attacking in pseudorandom subset of IP addresses as quickly as allowed by Internet access speed. Storm [27] worm in 2007 was a fast spreading email spammer that targeted Microsoft systems. Infected systems were 'zombies' added to a storm botnet. In a 5 month period it had infected 1.7 million computers. Conficker [28] worm first

discovered in November 2008 infected 9 to 15 million Microsoft servers running everything from windows 2000 to Windows 7 beta. It exploited flaws in those Microsoft products and performed dictionary attacks on administrator passwords to propagate while forming a net of bot machines. It was difficult to counter because it used many advanced malware techniques. All of the prior are some of the major worm threats in history that caution us of what is about to come.

2.2 Worm characteristics

Researchers [29, 30] in the past have concluded that it is possible for a worm to infect more than 90% of the Internet population in 510 tenths of a second if using UDP and 1.3 seconds if using TCP. There are three factors [31] that determine if an Internet worm can do great damage and spread quickly or not. 1) Exploiting security holes of the system, 2) The way of choosing its victims, 3) Its infectiousness. When a software vulnerability is identified the company's programmers work hard to create a patch that will be distributed through Internet updates and seal that hole on every host running that software. Unfortunately this procedure requires time that might not be available in case of a worm outbreak. A worm's effectiveness in spreading highly depends on the target detection algorithm it uses and can be distinguished by 1) Random scanning[3,32], Local scanning[2], Using a List[3,32], Topological scanning – IPs that the host system may hold[3,32], Commutative scanning-divides IPs to smaller blocks for faster infection results in parallel[3,32], Using hosts as carriers to spread and also employing already available communication standards, Limited users ad hoc worms that can deactivate[33], Hybrid techniques[3,31]. Infectiousness of a worm depends on whether its purpose was to monitor hosts or do harm. It is obvious that a highly infectious worm would be faster known due to its nature [2].

2.2.1 Software vulnerabilities

Worms in order to be highly infectious are targeting security holes in software and try to exploit it. What mostly determines a worm's effectiveness are the publicity of the current security hole it exploits, the time required for this to be widely known and how easy it gets exploited.

Malware authors are trying to make the maximum damage to as many hosts as possible. In order to succeed they are targeting widely known security holes in the software most widely used and by the highest number of users [2]. This means that Microsoft products such as Windows and Office are the prime targets for any malware attacks [34]. On the other hand, Linux distributions that are preferred by followers under the 10% of market share are considered more secure to use. This asymmetry in operating system preference also draws attention of newcomers and script kiddies, skilled and unskilled with the users now being more cautious than ever. Operating system homogeneity although it benefits in points such as compatibility between applications, it increases the risk of getting infected by a malware.

As previously mentioned time favors the attackers. The time between the identification of a security hole to developing, testing for its good working order and distributing the patch that seals that vulnerability exceeds by far the available time in most of the cases [35]. Users also stale to install the patch even when it is available to system updates waiting to be installed. It is thus obvious that as new as the vulnerability might be the more chances it has to be exploited by a malware and infect more machines. Slammer worm continued to infect machines even 6 months after the vulnerability was known. Witty worm did that 1 day after the vulnerability was known to the public. The biggest threat to systems, are zero day exploits [36, 31], not known, not expected, with limited defensive measures to hold back and stay immune. It is also obvious that if an informed crowd cannot stay immune; the damage an unknown threat could produce greatly increases.

A solution to software security holes might be tools that are capable to scramble the code- mutate it [37] while keeping the original algorithm intact. The code changes itself each time it runs, but the function of the code (its semantics) will be intact. Such techniques are sometimes used by malware such as worms, to hide their presence. Now it might be used to favor the defenders and protect applications from being easily manipulated by malware. Software developing companies are now employing arc injections, pointer subterfuge and heap smashing [38] techniques to keep their programs safe from being exploited. Although security is now taken very seriously by software developing companies, attackers have evolved and publish their findings of vulnerable software so it can be used by others not such talented.

2.2.2 Target selection

An important part of a worm is its target selecting algorithm. [3] discusses the importance of having the right spreading approach. If this is so, infection and spreading time increases dramatically. In contrast, sometimes worms are build to monitor rather than cause trouble. In this approach, a more loosen way of spreading is chosen, defined by an algorithm that is selected to hide that worm outbreak tracks. This means that algorithms that increase Internet traffic by sending huge amount of data to find their targets are not preferred. Algorithms that do so are Random scanning, Commutative scanning. Algorithms that will most probably do so are those used by Ad hoc worms and Hybrid techniques. Also local scanning will raise the local traffic of a target LAN.

Random scanning is the easiest and fastest way of implementing who to attack. When a worm decides to attack a random IP if there is a holder machine of that IP then the attack is performed. The majority of current worms employ that approach (Slammer, Code Red, Code Red 2 etc.). A key point to implementing random scanning technique lies to its generator of pseudorandom numbers. A perfect generator could lead to faster attacks without sufficiently increasing Internet traffic.

In Local scanning [2], IP addresses are selected which belong to the same subnet of the infected host. This is done by selecting targets between /8 and /16 of address space beginning from the IP from which the attack started. This way targeted computers will most likely be easily accessed (no firewall) and of the same software specifications. Attacks will be executed easily and fast [39], the same way Code Red 2 and Nimda did.

Selecting targets to infect can highly determine a worm’s fate. If those targets are junction points with advanced communication capabilities, the worm will be able to spread even quicker. Also a worm might infect many targets, and then deactivate. Later when there will be enough infections to start the second phase of spreading, it will activate with perspectives to reach new outbreak limits. Although creating a list of selected targets takes from weeks to months to complete and results may grow stale, a sufficient number of non stale list targets would also suffice. There are many ways a list can be created:

- Proxy Servers that maintain and communicate the existence of other servers.
- Public searching might reveal important security evidence
- Stealth scanning technique will be a non detectable scanning technique to identify specifications - security vulnerabilities of probable future targets (e.g. open ports).
- Distributed scanning performs multiple scans in parallel using networks of cooperating computers [31, 3, 35], everyone assigned to search specific blocks of IPs. This technique is faster than the stealth scanning technique.

Topological scanning is using target information that is already saved in a victim's computer. When the worm infects a host, it searches for this information in e.g. email logs or other files certified to keep such information (e.g. hosts file in UNIX). This technique was used in Morris worm that led to the first worm outbreak of the history. It is not easily implemented and requires certain conditions to be successfully applied (e.g. information of target to exist in certain files and be populated).

Commutative scanning divides IP addresses to smaller IP blocks and every block is given to a worm to perform its scanning on that area. This way procedure of selecting and infecting targets is done in parallel and more sufficiently. This approach is better than random scanning because already scanned machines are not re-scanned and re-attacked. In addition, fewer packets are wired to Internet thus minimizing Internet traffic.

Another approach of infectious network worms is those that employ already existing communication standards to attack their victims [3, 35, 2]. Nimda worm infected systems vulnerable with Microsoft IIS security hole and Internet Explorer. When a user visited an infected Internet server, his system was infected and if the server was compromised so the user would be. It is obvious that worm was spreading according to visiting patterns of the users to servers and thus not as quickly as the other techniques.

Ad-hoc worms upon infecting a host they make it a bot. Bots are machines that are used according to malware author preferences, either to send spam mails or massive data in order to perform denial-of-service attacks to certain targets. One important goal of performing such actions would be Cyber terrorism probably to ask for

ransome. This approach does not require a huge number of infected machines to reveal the worm to security applications.

Hybrid worms [3, 31] include combinations of the prior techniques that theoretically can create the ultimate worm. This worm will be capable of infecting the majority of the Internet in the least time. It has not been created yet and is not an easy task. In the future, such attacks must be easily predicted and avoided because of the dependence of modern word to digital communications that in case of an attack will most probably lead to an economical chaos.

2.2.3 Worm Infectiousness

The most dangerous worms of all were not carrying any infectious load [40]. Those worms were quickly identified because of their tracks. If they were waiting on enough machines until the possible victims were identified then a massive worm outbreak would be launched and they would be even more destructive. On the other hand, the vulnerability would be identified and a patch would prevent them from causing any further trouble. This is a risk that malware authors must be willing to take to reach even further levels of infectiousness. The only fact is that the worm must first use the infected machine to infect others and then destroy them [31].

2.3 Modeling worm spreading

Modeling of malware spreading is an important factor that can provide visualization and make it easier for containment strategies. Also if some characteristics of the malware are known then a probable future infection extent can be distinguished. Usually modeling is used after an outbreak to determine what led to that result and why [3, 41]. It is also used for probable new malware threats to determine their spread extent and consequences [31, 30]. Most of the modeling approaches use epidemiological models originating mostly from biological organisms. [42] provides a general epidemic model known as S-I-R that can sufficiently predict the evolution of an epidemic using differential equations. S-I-R’s differential equations can only be under certain conditions. Those would be that the systems involved are linked together and can form a universal homogeneous graph [43, 44].

S is the number of vulnerable organisms, I is the number of infected in a population, R is the number of those that recovered or are quarantined or dead, β is the rate of infection per contact and γ is the removal rate of infected members due to treatment, quarantine or death.

$$\frac{dS}{dt} = -\beta SI$$

$$\frac{dI}{dt} = \beta SI - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

The prior differential equations require homogeneity of the population and population to be stable according the formula:

$$N = S(t) + I(t) + R(t)$$

Until 2000 researchers only focused on analyzing the malware code to be able to fight against it. Later and because that way was not adequate researchers focused on modeling malware infection using epidemiological models [3]. S-I-R general model was then transferred to fit computer science.

N is the complete population of the machines connected to Internet, S is the number of vulnerable systems having the same security hole, I is the number of infected systems, R is now the number of all protected systems without the security hole or the already infected, β is the rate of infection per contact and γ is the removal rate of infected members due to treatment, quarantine or death. Another parameter is calculating the relative removal rate and result from the above equations:

$$\rho \equiv \frac{\gamma}{\beta}$$

An infection outbreak may happen only if: $S_0 > \rho$.

Although improving antivirus and intrusion detection technologies can enhance security towards malware, this is not enough. In the past Witty exploited security holes in a whole suite of protection programs. The solution is both analysis of the

malware code and modeling of its probable infection extent. Firstly to protect machines as single units and secondly to evade infection of whole technological structures. Also additional collaborations on epidemiology of computer systems can be made for better results. And as it always has been the best way to defeat your enemy is to know how and where he moves.

2.4 Related Work

Local security mechanisms besides securing machines locally it would be more efficient to cooperate and share their information in a network of peers. Key requirements of such systems include:

- Resilience. No network can be free of congestion, attacks and system failures. Thus it is important for protection networks not to fall in the face of a worm outbreak or other threats.
- Collaborative attack detection and response. Collaboration against common enemies can greatly benefit the defenders. On the other hand this synergy should not significantly increase traffic.
- Decentralization. Central points of interest have always been a weakness on such systems. For that reason single points of failure must not exist.
- Low cost. Using a system like that should not increase cost due to resource demands.

Most of the research done the last years on attack detection and protection through defensive measures is on signature based IDSs and statistical anomaly based IDSs [45]. Signature based refers to Network packet monitoring and further comparison with known premade attack patterns, the so called signatures. Statistical anomaly-based IDSs look for any strange anomalous traffic on already known network activity (known used bandwidth, known protocols in use, connected ports and devices).

Additional separation categorizes IDSs to Host-based IDS (HIDS) that run on individual hosts in the network. Their purpose is to monitor inbound/outbound traffic to/from a computer as well as the internal activities such as system calls and system logs and lacks of the ability to be aware of the network activities. Network-based

IDSs (NIDS) monitor network traffic to/from the network system. They sniff packets, and process and correlate data triggering alarms on intrusion events. NIDS do not have the ability to know about the activities on individual computers.

Many existing centralized intrusion detection systems, process data in single point of failure nodes and even if some of them employ distributed techniques they cannot evade the inevitable. Emerald [46, 47] and NetStat [48] are trying to surpass their scalability and fault tolerance issues using hierarchical designs. Even if attack information passes through different processing layers, they still require centralized control. Those approaches are all subject to drawbacks stemming from the client/server architecture they follow. Their being dependent on specific, low number of servers makes them vulnerable to targeted attacks. Collaborative systems such as Netbait [49] and Cossack [50], cannot surpass network congestion problems and are not scalable. The first offers distributed worm detection services on a structured overlay network while the other is a collaborative system detecting denial-of-service attacks.

Indra [51] is an early Cooperative intrusion detection system proposal similar to [14] in which cooperating nodes share warnings about an intruder in a P2P network. If a node detects an attack from another node then, it multicasts messages about the attacker to the other trusted nodes. This way the informed nodes are able to protect themselves from the infected-attacker node. Indra targets LANs and is fully distributed.

NetShield [52] is an IDN that monitors epidemic worm and Denial-of-Service attacks. It uses the DHT Chord peer-to-peer system to load-balance the participating nodes. A threshold is defined to activate an alarm if the local prevalence of a content block is exceeded. Also it is efficient only on worms with fixed attacking traces and not on polymorphic worms. The system is fully distributed and assumes that all nodes are trusted.

Gossip-based Intrusion Detection [53] is a local epidemic worm monitoring system. When a threshold is exceeded based on the number of newly created connections a local detector raises an alert and further propagates it to neighbors for aggregation. A Bayesian network analysis system is used to correlate and aggregate alerts. It is fully distributed with no effect on UDP worms that do not require connections, a feature in which the system is based on.

Host-based Collaborative IDS (HBCIDS) [54] is a cooperative intrusion detection system where IDSs share detection experience with other hosts. Every host sends its alerts to its neighbors for analysis. Every host has a feedback which is aggregated based on its trust-worthiness using test messages. Feedback trust values are updated after every interaction experience. This is a fully distributed system.

ALPACAS [55] is a cooperative spam filtering system that aims to preserve the privacy of emails. A P2P system is used for the scalability of the system. Emails are divided into trunks and digested into fingerprints to enhance content privacy of emails. Email fingerprints are sent to agents to compare them with known Spam and Ham email fingerprints and determine the percentage of overlapping. Emails are labeled depending on a threshold of difference between Spam and Ham fingerprints.

Hummingbird [56] uses peer communications in order to exchange security information between networks or systems without the need of a central administration point. It is considered more intricate than ours, incorporating Manager-Hosts, Managed Hosts, Slave Hosts as well as Peer, Friend and Symbiotic relationships for the security information exchanges. It also detects malware code and uses advanced visualization tools for the manipulation of log files.

Domino [57] monitors internet outbreaks for large-scale networks. Its nodes are organized hierarchically with different roles assigned to each node. Each role's messages are distinguished by different trust levels which enables DOMINO to handle inter-administration zone cooperation. It is a scalable Global IDN, decentralized and with hierarchical structure.

Worminator [58] purpose was to allow information sharing between IDSs. Alert correlation is used for better detection accuracy. It enforces bloom filter to encode IP addresses and port numbers in the alerts to provide enhanced privacy between the collaborators. It can be either centralized or decentralized.

In Agent-based Distributed alert system (ABDIAS) [59] IDSs are grouped into communities sharing usually intra-community or inter-community communications only when a decision cannot be made from community members. A Bayesian network system is used to make the decisions. Early warnings are supported to earn time for administrators to fight attacks and a voting system decides whether a node is infected or not.

In Cyber Disease Distributed Hash Table (CDDHT) [60] each node detects intrusions locally and generates corresponding alerts, each with a disease key based on the intrusion. Each alert is sent to super nodes with increased resources called sensor fusion centers (SFCs) using a P2P network that supports DHT. CDDHT is a decentralized system which also supports load balancing through the categorizing of the alerts and forwarding them to the corresponding SFC.

Bakos and Bert [61] also described a system that is based on a central system that stores and processes messages that do not correspond to actual computer destinations. Those messages are the outcome of highly active worms that use scanning techniques in order to identify potential victims. Routers that intercept scanning messages in most of the cases reply with ICMP Destination Unreachable messages. A copy of every message is sent to a central collector system that forwards it to an analyzer system which is responsible for processing it and extracting any valuable information.

Dshield [62] is a centralized firewall log correlation system. Data is saved in the SANS internet storm center [63] which populated its collection with firewall logs received from worldwide volunteers. It is a centralized system which provides no real time analysis with often difficulties in attack classification due to removed data payload for enhanced privacy.

CRIM [64] is a centralized system that collects alerts from participating IDSs. Alert correlation rules are generated by humans offline that analyze attack descriptions. These rules are used to detect global-wide intrusions. It is a centralized system depending entirely on humans to provide the rules.

SmartScreen [65] is a phishing URL filtering system. It is used in Internet Explorer 8 to allow users to report phishing websites. A centralized decision system analyzes the collected data that came both from users and other trusted sources and generates a blacklist. Upon visiting a phishing website SmartScreen will warn the user.

Fastflux service networks (FFCIDN) are different fastflux domains or else botnets that handle compromised nodes. A collaborative intrusion detection network [66] is used to detect those networks and stop them from doing any further damage. The detection system collects and observes the length of IP addresses that were collected as query results on different locations and comes with the number of unique IP addresses and fastflux IP addresses. A threshold is derived based on queries and

results. This is a centralized system that requires the use of nodes to help on detection process.

There are a number of databases such as Dshield that include vulnerability information. The list includes NIST's National Vulnerability Database (NVD) [67], the Bugtraq security database [68], Symantec DeepSight [69], the Open Source Vulnerability Database (OSVDB) [70], and the Common Vulnerabilities and Exposure (CVE) referencing standard [71]. We therefore have knowledge gathered and stored for many years and from security researchers all around the world that can be used to add on this course.

Syzygy [72] is an epidemic detection application which uses the client/server model on a community of clients. Its prime purpose was to detect epidemics. It observes the operation of every client in order to determine any signal anomalies and builds a model with anomaly scores which is independent of its operation. While in monitor mode Syzygy periodically collects the most recent anomaly value from each client and compares it to community's anomaly value which is threshold V. If it is greater than V then Syzygy reports an epidemic.

Snort [73] just like Bro [74] which are intrusion detection systems based on rules are unable to handle state of the art malware. This happens due to their lack of knowledge of the way the new infection takes place which causes the spreading of the malware. Inevitably they have to wait until their signatures get up to date so that they can defend against it.

Microsoft Research's Shield [75] provides vulnerability type of signatures. This way the system knows the general resemblance of the exploit and any connections containing such data on packet datagrams is dropped. Then a pre-patch is manufactured which performs vulnerability-specific and exploit-generic modifications that shields the system from the specific malware as long as it is properly updated with that pre-patch. Again time favors the attackers.

A great number of researchers have considered that they would be well favored working on the streaming packets and additionally analyze their contents in order to generate content based signatures. Honeycomb [76] is a host-based IDS that uses longest common substring (LCS) on malware which is captured by a honeypot targeting dark space in order to create its signature. This substring is afterwards used as candidate worm signature. PAYL [77] has a choice on whether LCS will be used or

the longest common subsequence (LCSeq) on anomalous packets not always targeting a honeypot, but any victim in the protected LAN. Autograph [78] uses heuristics to classify traffic into two categories: a suspicious scanning activity flow pool and non-suspicious flow pool in the same way Polygraph [79] does. Suspicious flow pool traffic is TCP flow reassembled and the payload is partitioned to small blocks with the help of Rabin fingerprints. Then stemming from their count, the most frequent counted substrings from those blocks form a worm signature. Blacklisting is incorporated by the signature generator so as the number of false positives to be minimized. Assuming that signatures are not contiguous, and having multiple noncontiguous composed substring (token) signatures, Polygraph can accommodate polymorphic worms. Hamsa [80] is a more improved than Polygraph when comparing their accuracy, efficiency and toughness on attacks. Just like Polygraph, Anagram can identify multiple "tokens". According to its design Anagram doesn't incorporate an external flow classifier because it is one by itself. Earlybird [81] similar to Autograph can also detect new worms. There is a frequency count table where the substrings of every packet which are outcome of the Rabin fingerprint computations are inserted and numbered. Their order is based on rank which is determined by the frequency counts. Source and destination IPs are also stored. Then the system counts the number of IPs both source and destination so as to minimize the false positive rate. By its nature it is a centralized system. Also following the same philosophy [82], [83], [84], [85], [86] and [87]. But as though as impressive those techniques may seem worm signature approaches have little lack when they deal with encrypted or deformed worms.

Chapter 3

JXTA

3.1 P2P

Peer-to-peer networks are more capable than client-server networks. Their main advantage is that they utilize better the existing resources that are scattered all over the Internet or in a network alike. P2P networks are used to provide collaborative assistance on demanding tasks that require a number of machines [88, 89]. On the other hand client-server architectures require one or more powerful servers and a number of not so powerful clients. Every client has to contact a server to get its job done (e.g. file sharing, web browsing etc).

3.1.1Hybrid

It is a combination of client-server and P2P models but can still be distinguished as P2P. It is used primarily in file sharing applications such as Napster. There is a central server that holds records of all the files in the network of peers [90]. A user that wants a file firstly asks server which responds with the locations of that file. Then the user asks that file from the peers that have it. Hybrid model's features include: Searching for resources is fast and accurate because of the central server. Connecting and leaving of peers does not seriously affect the network, because only the central server needs to be informed. Although central server is a single point of failure, there can be additional servers that will share the load and enhance networks resilience. As far as security and trust management, all nodes are authenticated by central servers.

3.1.2Purely decentralized

In this approach every node can be both client and server at the same time with no single points of failure. Although it can be highly resilient to attacks, it has scaling problems. Gnutella was a true paradigm of such a network but as many users populated the network, scaling problems demanded its redesign in certain points [91, 92].

3.1.3Partially decentralized

This approach is similar to purely decentralized but also distinguishes peers to simple peers and super peers that have administration responsibilities. It is considered effective and for that reason it is used by P2P programming platforms such as JXTA. It is scalable and resilient to congestion, attacks and system failures.

Advantages of a P2P network

- The system is based on communications between peers.
- There is no reliance on central services or resources to operate.
- It is resilient to changes in network composition.
- It is highly scalable.
- It consists of peers coming from heterogeneous network environments.

3.2 JXTA framework

JXTA stands for Juxtaposed. It is an open source framework which contains various protocols that allow P2P interaction between hosts that are using it. It was first developed in 2001 by Sun Microsystems. XML messages are used by its protocols to allow communication regardless the network topology. JXTA's virtual overlay network can allow communication through firewalls and NATs. Every resource in a JXTA network is defined by a unique constant ID a 160 bit SHA-1 URN. In peers this ID is used for communication in a JXTA network.

3.2.1 JXTA concepts

JXTA networks consist of peers. [93] defines a peer as: *Peer is an entity capable of performing some useful work and communicating the results of that work to another entity over a network, either directly or indirectly.*

The definition of useful work depends on the type of peer. Four types of peers exist in a P2P network.

- Simple Peers
- Rendezvous Peers
- Relay Peers
- Rendezvous/Relay Peers (Combination of the two)

Each network peer can have one or many peer roles, and as a result a different set of responsibilities to the P2P network.

Simple peers:

- A simple peer serves a single end user and makes him capable of providing services from his machine or consumes services that others alike provide.
- Most of the cases a simple peer is located behind a firewall, isolated from the P2P network. Peers outside the firewall are not capable of communicating directly with that peer.
- Simple peers have the least amount of responsibility in any P2P network.
- Simple peers unlike the other two types are not responsible of handling communications between other peers or serving third party information to be consumed by others.

Rendezvous Peers:

- A rendezvous peer is a gathering place where other peers meet.
- Rendezvous peers provide peers with network locations to discover other peers and their services.

- Peers query rendezvous peers using discovery queries and rendezvous respond with information about other peers they know on the network
- Rendezvous peers extend their knowledge-services by forwarding discovery queries to other rendezvous peers and caching information on peers for later use.
- By extending their knowledge-services, rendezvous peers improve their QoS to simple peers and reduce network traffic.
- Rendezvous peers are usually placed outside private networks, but can also be on the inside if they are capable of traversing the firewall or with the help of a relay peer.

Relay Peers:

- Relay peers allow other peers to communicate with those that are located behind firewalls and/or Network Address Translation NAT equipment.
- They provide routes and mechanisms for isolated peers to be able to be a part of the P2P network and communicate in both directions.
- Peer messages must pass through the proper relay peer to reach destination.

Services provide peer functionality that might include e.g. transferring files, performing calculations, transferring calculations, performing status information and a whole lot of functions depending on preference. Services are the reason of creating a p2p network, and can influence its popularity and longevity. There are two service categories:

- Peer services: Functionality provided by a peer to other peers on the network. Depending on the peer status, its services might be unavailable if it is disconnected.
- Peer group services: Functionality provided by a peer group to its member peers. Those services are provided by members of the group and if at least one peer exists on that group and provides those services then the services will be available for member or joining peers.

Advertisements: Until recently P2P networks were using informal forms of advertisements. [93] also defines an advertisement as: *Structured representation of an entity, service, or resource made available by a peer or peer group as part of a P2P network* such as:

- Peers
- Peer groups
- Pipes
- Endpoints
- Services
- Content

JXTA advertisements are XML structured documents such as:

```
SocketService Costructor CALL
SocketService PipeADV: <?xml vers...
<!DOCTYPE jxta:PipeAdvertisement>
<jxta:PipeAdvertisement xmlns:jxt...
    <Id>
        urn:jxta:uuid-59616261646...
    </Id>
    <Type>
        JxtaPropagate
    </Type>
    <Name>
        test
    </Name>
</jxta:PipeAdvertisement>
```

Figure 3.1 XML Advertisement

Every item in a P2P network must carry a unique identifier information on that network. This information is necessary for peers to be able to communicate via third parties, for peer groups in order peers to be able to join, leave or query a group, for pipes so that peers can identify pipe endpoint connections, for content so that peers can mirror any content across network and later have access to that content wherever it is located.

A peer can discover an advertisement using passive discovery technique while not connected to network, and active Direct or Indirect techniques while connected. In passive technique previously discovered cached advertisements are used. Although it

reduces network traffic and is a good starting point for active peer discovery the cache list can grow stale. In active direct the discovery is made without any mediators such as rendezvous peers and usually takes place in a LAN. Unfortunately it is also limited to a LAN. In active Indirect a rendezvous is used to act as a source of known peers and advertisements while it performs discovering to serve the connected peers. This technique can also be used in a LAN or a private network to find other peers that do not support multicasting or broadcasting and peers located outside the LAN or private network. Rendezvous peers can use both propagation of advertisements to other known simple or rendezvous peers, or cached advertisements to respond to peers' discovery queries. Propagation of discovery queries without restrictions can cause indirect discovery propagation chaos and thus network congestion from queries looping through peers. TTL can alleviate this phenomenon by setting the number of allowed propagations between peers a message can have.

3.2.2 P2P Routing and resource discovery

Routing on P2P networks can be either centralized and statically configured (e.g. on hybrid P2P or on client/server) or implemented on top of a virtual overlay network on top of physical network topology. P2P overlay can be either structured [94] or unstructured. In unstructured overlay networks nodes can join, leave anytime dynamically and with no uptime guaranty (e.g. Gnutella, Kazaa). Resources are located by employing flooding techniques. It is clear that topology is highly dynamic with no standard valid routes. In structured overlay the overlay is tightly controlled and organized into a specific topology. Files or pointers to resources are placed to specific locations and variants of the distributed hash tables (DHTs) technology is used to assign ownership of each file to a specific peer. Any node can efficiently search the network for a file/resource, even if the resource is extremely rare. Routing and resource discovery especially in unstructured overlay P2P is very problematic [95].

Several overlay routing algorithms have been surveyed both for structured and unstructured overlay P2P networks [96]. Which of these algorithms suits best for the purpose depends on the network type, its application and its required functionalities and performance metrics, e.g. scalability, network routing performance, location

service, etc. Here for the purpose of this thesis, distributed hash tables (DHT) will be analyzed which is supported by JXTA (Loosely Consistent DHT). According to [96] DHT based systems are considered efficient in many of the tasks.

3.2.2.1Distributed Hash Tables (DHTs)

DHTs are tables containing numbers generated from variable length string of bytes. DHT algorithms such as Chord [97] hash identifiers and store them to a giant hash table that is distributed across the participating nodes. In addition, all these nodes use this function to hash their IP address and form a ring in ascending order of their hashed IP with the next node being successor(x).

- DHTs are used when sharing files and the hashed file identifier (generated key) together with the IP are send to the successor (key). This way the node saves locally this information further adding to the distributed hash table among nodes. If there are many nodes having that specific file or resource then the keys will be stored locally, giving requesting nodes choices.
- DHTs are used when requesting files. A node that wants a specific file or resource will hash its identifier and send a request to successor (key), and the last will answer with the IP of the node that has the data. When the IP is not known, but the key is, a finger table with the keys and their successor IPs is maintained.
- DHTs are used when searching for files. Each node is aware of the IP of the next real node in the ring. If the key exists between node k and the node then successor (key) is the next node. If not, the finger table is searched for the closest predecessor of the key. The request is forwarded to this node containing the IP of the requesting node and the process repeats until the key node is reached and reply to the contained IP.
- DHTs are used when joining or leaving a group. When joining, a node hashes its IP address and sends a request to any node on the network with purpose to find successor (number). Then it takes a certain amount of its successor's hash table and messages the predecessor to inform of the new successor. When leaving it sends its table to its successor and informs the predecessor that will

depart. In both situations finger tables retain wrong values and for that reason each node periodically sends messages to the ring to find new nodes.

3.2.3 JXTA Protocols and Architecture

JXTA is characterized by:

- Interoperability. Allows interconnected peers to locate and communicate, participate in community-based activities, and enjoy services offered by other peers in a variety of different P2P systems and communities.
- Platform independence**.** It is independent of programming languages, Operating systems, and networking platforms (e.g. TCP/IP, Wi-Fi).
- Ubiquity**.** It can be implemented on a wide variety of devices (e.g. sensors, PDAs, PCs, data-center servers, storage systems etc.)

Communication between peers uses a group of request/reply asynchronous protocols. JXTA protocols [98] provide the ways peers, discover each other, self-organize into peer groups, communicate with each other, monitor each other, advertise and discover network resources and are platform/operating system independent.

JXTA Protocols are:

- Peer Resolver Protocol (PRP) – Enables peers to send generic queries to one or many peers and be able to receive and manage responses. PRP associates a unique ID for every request and includes in each message. Protocols such as PDP make use of PRP to create their requests.
- Peer Discovery Protocol (PDP) – Used by peers to advertise their own resources and make them available to other peers.
- Peer Information Protocol (PIP) – Peers use this protocol to obtain status information of other peers, query peer capabilities or monitor its behavior.
- Pipe Binding Protocol (PBP) – Facilitate communication path between peers establishing virtual communication channels.

- Endpoint Routing Protocol (ERP) – Routes between peers are found using ERP to ask other peers for path information about a destination a message will be send to.
- Rendezvous Protocol (RVP) – For efficient propagation of messages in the network. PRP also uses RVP.

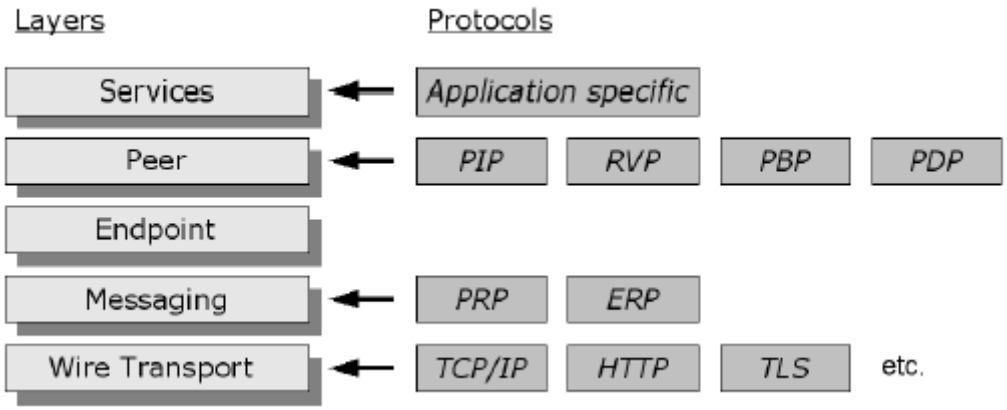

Figure 3.2: JXTA endpoint communication layers

Basic protocols services are:

- Rendezvous Service: is used for publishing messages outside the peer group
- Discovery Service: is used for Advertisement discovery e.g. peers, peer groups, pipes, services in general of a peer group
- Pipe Service: via pipes allows to send and receive data
- Endpoint Service: allows the transmission of simple messages
- Resolver Service: provides a generic mechanism for sending requests and receiving replies
- Peer Info Service: provides information about group nodes
- Membership Service: allows unique identities on peers in order to guarantee their presence in a group.

In JXTA implementations there is no need to deploy all core services, but the essential ones are PDP and ERP that provide addresses to peers and allow communication between endpoints. The rest of the protocols can be avoided but this may lead to possible interoperability issues and limits the degree of functionality.

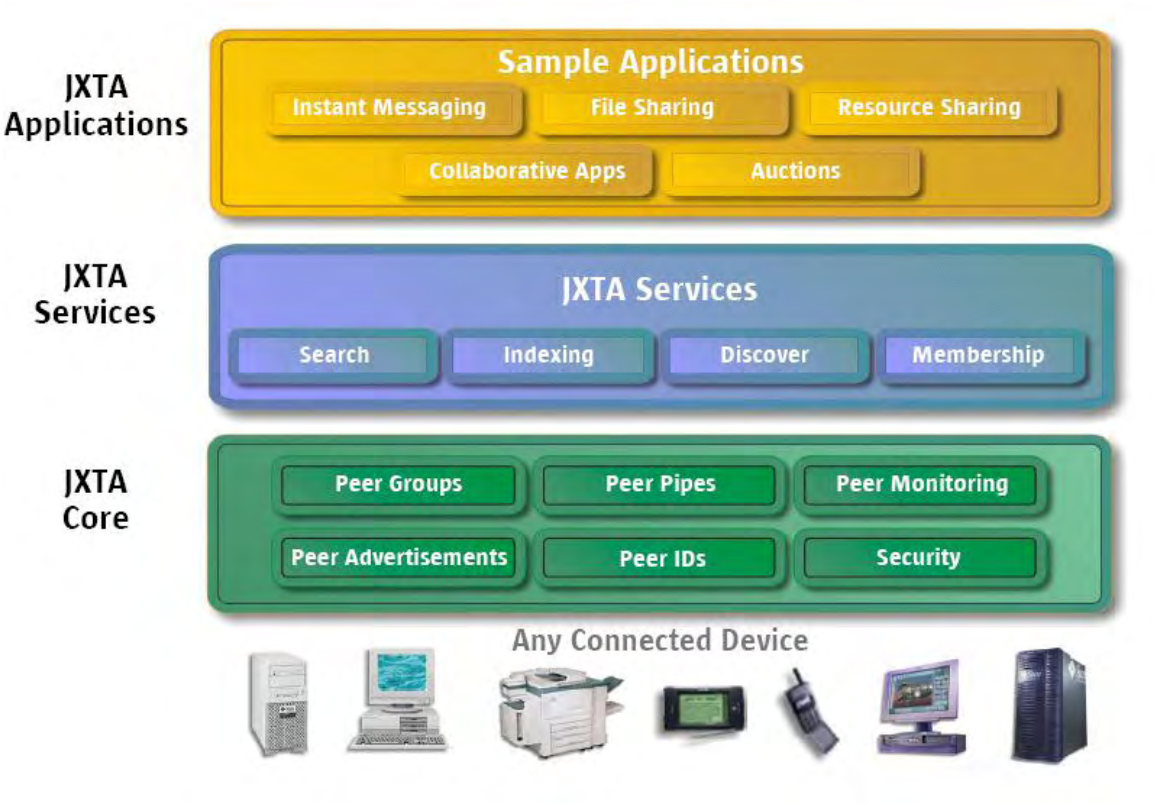

Figure 3.3: JXTA Architecture

A typical P2p software stack is divided into three layers:

a. JXTA Core layer (e.g. discovery, transport, peers and peer groups creation, adjusting security primitives)
b. JXTA Services layer (e.g. searching and indexing, file sharing etc.)
c. JXTA Applications layer (e.g. P2P instant messaging, document and resource sharing etc.)

Additional components of JXTA:

Shared Resource Distributed Index (SRDI):

In JXTA, metadata describes resources as XML documents - advertisements. SRDI's [99] goal is to index those documents in the network, using a set of specified attributes. The maintained distributed index is just as a hash table, with the indexed attributes as hash keys which map back to the source peer containing the actual advertisement. Queries can be made anywhere in the rendezvous network based on these attributes and thus SRDI is able to locate peers with the required advertisements

and respond to advertisement queries in the network. SRDI alleviates from the need of remotely publishing advertisements but only the advertisement's index on a peer is published. This index data is passed to the rendezvous super-peer network DHT via a single connected rendezvous.

Loosely Consistent DHT [100]:

In JXTA a distributed virtual hash table contains the indices of all the published advertisements in the entire JXTA group. Every edge peer can query the hash table on demand via supplying a set of attributes (table hash keys). In turn the rendezvous super-peer network resolves the query by hashing the key to the required value. The procedure that creates this DHT requires every peer to cache advertisements locally, which are then indexed. Rendezvous nodes receive indexes from edge peers. The rendezvous super-peer network is responsible for maintaining the DHT with the indices. Rendezvous peers are the only recipients of queries and if a rendezvous becomes unreachable, its index part that maintains also becomes unavailable until it gets published again by the responsible peers.

Rendezvous peerview (RPV) and RPV walker [100]:

The RPV is the list of known rendezvous to the peer, with their respective IDs in order. The DHT algorithm uses an identical hash function in every peer whose role is to determine the appropriate rendezvous to whom a locally irresolvable query request should be sent to. Any unreachable rendezvous is extracted from a peer's RPV. Every rendezvous in the network of super peers regularly sends to other rendezvous peers a subset of its RPV to a random selection of rendezvous in its RPV. This happens for two reasons, first to ensure convergence of RPV network and secondly to adapt to any physical network partitioning or merging.

While maintaining the DHT, SRDI using a fixed hashing function stores incoming index information to a selected rendezvous peer in the super-peer network. The index information is copied to adjacent rendezvous peers on the RPV to maximize the probability of successfully hashing of that index information in case of a rendezvous crash. A hashing function is performed against a rendezvous' own RPV when resolving queries. This happens because a rearrangement might have taken place and

multiple rendezvous may have disconnected or joined the super-peer network. If hashing does not immediately resolve the query, an optional customizable RPV walker might be used to forward the query to a limited number of additional rendezvous.

3.2.4 JXTA Security

Cryptography is a powerful tool that can almost eliminate access control and confidentiality issues faced by software applications during data exchange. Confidentiality between peers can be achieved using a public key system. Such based systems usually rely on asymmetric keys (RSA) for public exchanges and symmetric (Rijndael) for private exchanges. RSA allows a secure connection between parties, with identity verification and data origin certification. Although it provides security, it is not a complete solution due to its inefficiency to encrypt large quantities of data. For that reason, another solution, this time symmetric key encryption (Rijndael) was selected by the National Institute of Standards and technology (NIST) in 2001 as the advanced standard (AES) to be used by the U.S Government. Both solutions complement one another. Due to RSA inability to encrypt large quantities of data, after identity verification with the public key system, private keys are created between parties and use the more efficient private key system to exchange large quantities of data[101].

Both peers A and B create their RSA asymmetric key, which consists of two parts, a public (public key) and a private (private key). RSA public key encryptions can only be decrypted with the corresponding private key part and no other. Both peers publish their RSA public key part to authorities (certificate authorities) they trust which in turn are responsible for certifying its customers' keys. Certificates are delivered through established links between a peer and a public key. Others can later verify the validity of these certificates.

In other words peer A retrieves peer's B public key from the certificate authority and vice versa. They now verify the certificates and are able to use the public keys to encrypt messages and send them to the public key owner which has the private key to decrypt it. However public keys are available to anyone who asks for them from the

certificate authority. For that reason there must be a way of authentication to ensure messages are received from the expected peers and not malicious ones.

Peer A and peer B share a secret message encrypted by peer A with peer's B public key and attached to the encrypted messages sent to peer B that only the two know of (e.g. authentication code). This way upon receiving the message peer B decrypts the secret with its private key and ensures of the correctness of the secret and thus the authentication of the sender. On the other hand a malicious peer can substitute the encrypted message without interfering with the secret.

The solution to that problem lies to the secret's nature. If it is an authentication code based on not only the secret but also the message itself then the flaw is eliminated. This message authentication code (MAC) requires a sufficient hash function because it is not a simple operation. It must be guaranteed that finding the original message and secret by only knowing the MAC is impossible. Thus a malicious peer will not be able to create a proper MAC because it does not know the secret both peers share and does not have access to the message that is transmitted.

Again the problems of what secret and how should be shared must be taken into consideration. The secret must be as difficult as it could get, so as not to be guessed easily. The solution to those problems comes to give the Diffie-Hellman protocol. This protocol allows both peers to create a secret code from their remote locations using their RSA keys. A malicious peer in the middle won't be able to guess that secret. Also the secret can be used as a symmetric key between peers to encrypt bigger messages with faster algorithms such as AES. Secret stays valid only between the peers and no other.

Another important problem is a malicious peer pretending to be someone else and create a fake certificate. Certificate authority must be sure and thus verify that peer's identity through e.g. a license. Again this license would not guarantee that. A certificate invalidation trick might do the job. Again that mechanism is susceptible by peers that impersonate other peers. Certificate authorities register information such as unique emails and IP addresses of the peers making the request and in case of a problem they track down the user.

The truth is that we must have in mind all possible scenarios to avoid being hacked. There will always be a risk and malicious peers will try to perform exploits, impersonate fake identities, and/or corrupt communications. Remote and unnoticeable

secure communications involve a bit of risk to be efficient enough and avoid unbalanced tradeoffs to e.g. network traffic, user efficiency etc at the same time. There will always be users with bad intentions and systems that are not fully secured.

In JXTA the most efficient way to implement the certificates scenario is by requesting certificate authorities to certify public keys by providing peer identities in order peers to receive their certificates. Certificate authorities are authorities that create public key infrastructures – chains of certificates by certifying other users and authorities.

Every certificate authority provides a Certificate Practice Statement (CPS) – a document explaining how the authority issues and manages certificates, plus additional services it may provide. User identity verifications is a must mechanism that CAs should provide. They should also maintain a Certificate Revocation List (CRL), a list of all issued certificates that are no longer valid. CAs list is checked to make sure a peer's certificate is still valid and not revoked.

On the other hand on a Web of Trust where B peer knows C peer and can certify its certificate, A peer can make sure that B has certified C and no other and then A also trust's C peer's certificate. Through the same steps, C will do the same for A peer's certificate through B. Everyone creates its own certificates and have them certified by peers it trusts. Unfortunately when the number of users gets high then the long chains of certifications are very easy to get compromised. Peers from different chains certify a peer which is unnecessary and there is unclear how revocation can happen.

3.2.4.1 X.509

X.509 version 3 structure	**Description**	**Hashed field**
Version	X.509 version 1,2 or 3	Yes
serialNumber	Unique identifier of the certificate	Yes
Signature	Signature algorithm that should be used to certify this certificate	Yes
Issuer	Identity of the certificate	Yes

	authority issuing the certificate	
Validity	Validity period of the certificate	Yes
Subject	Identity of the user to certify	Yes
subjectPublicKeyInfo	Public key of the user	Yes
issuerUniqueID	Optional – Certificate authority unique ID	Yes
subjectUniqueID	Optional – User unique ID	Yes
Extensions	Optional – Additional information	Yes
signatureAlgorithm	Signature algorithm used by the certificate authority	No
Signature	Signature of the certificate itself	No

Table 3.1: X.509 version 3 structure

Table 1 describes X.509 version 3 of this certificate definition standard, first appeared in 1988. The procedure starts with the user sending its identity (subject), its ID (subjectUniqueID) and a request to use a specific signature algorithm to the certificate authority (signature). CA fills the rest of the fields and hashes them with a proper hash function. The resulting value is encrypted – signed with the private RSA key of CA which creates the signature of the certificate. Finally CA fills the signatureAlgorithm field with the type of the algorithm that was used and sends the certificate to the user.
The user distributes its certificate to other peers that in turn will check for its genuineness. This is performed by retrieving CA's certificate from the proper CA trace found on the user's certificate. CA's certificate contains its public key. The verification is happening through computing the hash value of the fields to be hashed in the first certificate using a method specified in signatureAlgorithm field. Peers also verify the content of signature field using the CA's public key found on its retrieved certificate. The value returned can be checked if it matches to the hash of the first in order to verify the validity of the certificate. Also validity period and CRL are the

final steps for a valid certificate. The procedure repeats for the root certificate authority to form a chain of trust.

3.2.4.2 X.500

X.500 is a standard for directory services. *It is a system of software applications collecting, organizing, storing, and providing information about network resources and users* [101]. JXTA uses the Java X500Principal to allow description of certificate authority peers when using cryptography. CA peer names and principal - identities are expressed using a X.500 standard format.

3.2.4.3Hash functions

Those functions are algorithms that map arbitrary length data to data of a fixed length. Such functions include lots of mathematics and rely on well known mathematical functions (e.g. SHA1, MD5). For example SHA1 returns a result of 160 bits long while MD5 a result of 128 bits long. This size is very important to cryptography because hackers are easier able to produce the same hash result using different inputs (collision issue) and successfully fake identities. To escape collision issue more or equal to 160 bits are required. The processing power required to create collisions in a range of 2^N bits increases leading to computationally intensive tasks that are currently not feasible (e.g. 2^256). Also good hash functions make it very hard for a hacker to guess the input value out of the output only.

3.2.4.4 Key Sizes

In addition key size is also important. Today's minimum acceptable size of a symmetric key is 128 bits. The size of an asymmetric key should be more than 1024 bits. RSA public key system relies on the product of two large prime numbers that cannot be decomposed easily. If someone finds an algorithm that achieves that, then suggested RSA key size will not be efficient anymore. This means that all of the systems alike are not proven to fully protect but on the other hand are not easily hacked also.

3.2.4.5 Personal Security Environment (PSE)

PSEmembership service is the most secure implementation of JXTA's membership services, using a public key infrastructure. PeerGroups can choose to use that service and elaborate on specified by code scenarios utilizing the tools it provides. A keystore is maintained by PSE membership service where the certificates and private keys are safely stored. X.509 is used for certificates and keystores in JXTA.

JXTA certificates on default JXTA are based on 1024 bits, which is not enough and believed to be breakable in the near future. The hashing method is SHA1 which returns 160 bit long results. A 2011 attack have managed to create SHA1 collisions with a complexity of 2^61 operations. Computer power to create collisions can be available by joining together networks of computers. In addition the password to encrypt the secret key corresponding to X.509 certificate should be long and complex enough to avoid being found easily.

3.2.4.6 Secure Socket Layer (SSL) & Transport Security Layer (TLS)

SSL first purpose was to provide secure HTTP communications between two endpoints over the World Wide Web. SSL is the base of the TLS protocol which tends to replace it. It uses TCP/IP to deliver data also employing cryptography methods of the data between the two parties established connection. It is independent of the end user application which means that it can provide secure transmissions over higher-level protocols such as HTTP, FTP, telnet, etc.

It provides communications privacy over the Internet and is designed to prevent eavesdropping, tampering or message forgery. Messages are private by using symmetric cryptography for data encryption (3DES, RC4, AES, etc.). In addition the connection is reliable because message transport includes a message integrity check using a keyed MAC. Also secure hash functions are used (MD5, SHA1).

- TLS has a handshake protocol followed by an application data protocol. Using TLS Handshake protocol peers authenticate each other with X509.v3 certificates. Also asymmetric or public key cryptography is used (RSA, DSA, etc.). Encryption algorithm and cryptographic keys negotiation is done before data is transmitted.

- TLS application data protocol permits the secure exchange of application data with symmetric cipher algorithms.

In JXTA peers can be both TLS clients and servers and authentication of both is required. JXTA uses TLS version 1.0 which sends data over encrypted communication channels and guarantees that any of the mediators will not be able to have access to the message content. Although TLS 1.0's cipher suites [102] are not all robust.

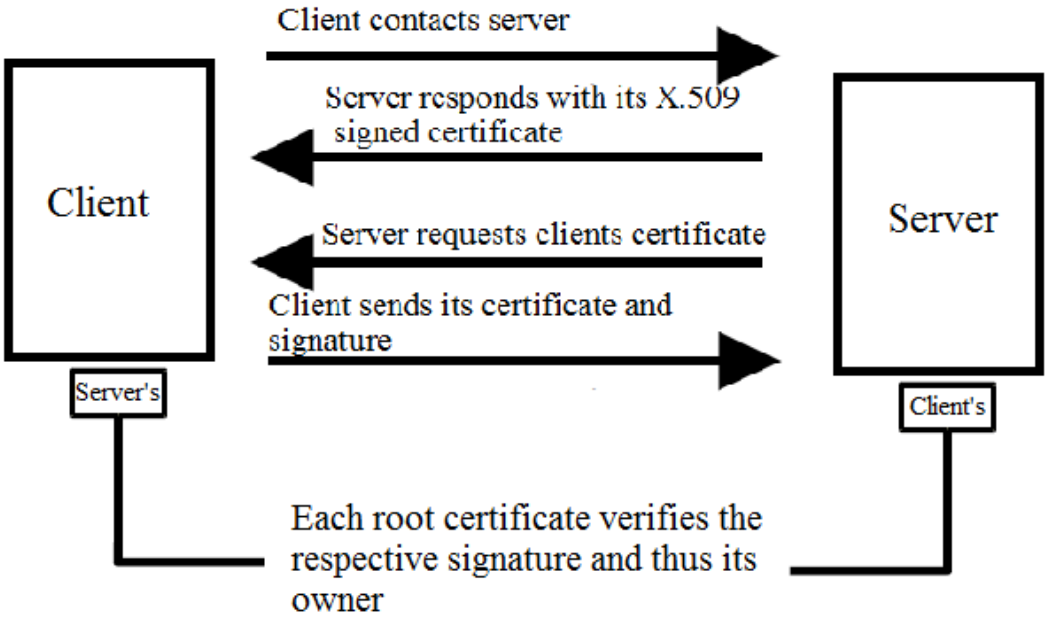

Figure 3.4: TLS Authentication

3.2.4.7 Private key & PSEConfig advertisement encryptions

JXTA automatically uses JKS keystore type but it also offers the possibility to create other type of keystores. The proper method to protect keystores and their keys is by first creating the keystore, then retrieve it from the PSEConfig and call the proper setKeyEntry() method after its private key has been encrypted. This protects the keys from unauthorized access to the keystores for e.g. if the PC falls to the hands of a foreign person.

If a private key and an encryption password are set to the NetworkConfigurator, the save() method call will create a PSE configuration advertisement to the local configuration. The advertisement contains the encrypted key which was salted 500 times with MD5 and encrypted with DES together with the result of the MD5 hashing. Again this method is not sufficient enough to protect the private key. The encryption password protecting it should be hard enough to avoid being found by brute force attacks.

3.2.5 JXTA Security overview

Start platform	Load platform Join netPeerGroup Open network listeners Open local cache
Join a peer group	Locate group advertisements Authenticate Join or Create a group
Publish own resources	Create and publish advertisements, locally and remotely.
Discover other resource advertisements	Locate advertisements (peer, pipe, data or any other custom) Store advertisements in local cache
Exchange messages	Open pipe Send and/or receive messages Check Access Service Close pipe
Disconnect	Close connections Shutdown platform

Table 3.2: JXTA basic operation steps

Desired security requirements:

- Privacy (Data are kept secret)
- Authentication (You are authentic and not an imposter)
- Integrity (My messages are received without being tampered)
- Non-repudiation (I cannot undo what I have already done)

One of JXTA's core services, the Membership Service is responsible for providing group membership and identity management within a peer group. This requires each group member to have a credential, a proof of its membership to the group.

The choice of membership service defines the security measure in the group join operation. Peers must be on the same membership service in order to acquire identities and join the group that implements it. Besides the generic membership service which can be modified according to application needs, there are ready and available membership services that can be used (Nonemembership, Passwdmembership, and PSEmembership).

- The Nonemembership service provides no authentication and peers can claim any identity they decide. Applications that use this service do not require any security measures.
- The Passwdmembership service implements a Unix-like username and password pair for peer authentication. The correct credentials are required in order for a peer to claim an identity. A list that includes all the credential pairs is distributed to all peers in the group. This membership is insecure and was built for testing purposes. For that reason it should never be used by any serious applications.
- The PSEmembership service is the more acceptable choice. The PSE membership's credentials are based on PKIX [103] certificate chains. Root certificate is originated and possessed by the group creator, also having the role of a certification authority. A keystore password is required to allow access to the keystore which holds the private key for that certificate chain. PSE's credentials are chosen to provide asymmetric key management messaging security services.

```
<xs:complexType name="jxta:PSECred">
  <xs:sequence>
    <xs:element name="PeerGroupID" type="jxta:JXTAID"/>
    <xs:element name="PeerID" type="jxta:JXTAID"/>
    <xs:element name="Certificate" type="base64binary"
             minOccurs="1" maxOccurs="unbounded"/>
    <xs:element name="Signature" type="base64binary"/>
  </xs:sequence>
</xs:complexType>
```

PSE Credential XML schema

The PSE Membership Service can provide security to JXTA messages at two different layers [104]: a) the messaging layer using the CBJX [105] protocol, b) the wire transport layer using TLS [102].

A. Messaging layer security

CBJX (Crypto-Based JXTA) is a lightweight JXTA-specific protocol that implements basic secure message source verification. This is accomplished by adding its own signature element into messages. This way it provides to messages data integrity and authentication.

CBJX [104] cannot directly send messages between endpoints. It pre-processes messages so as to securely encapsulate them. The new message is then relayed to an underlying wire transport protocol. Although CBJX is specified as a transport protocol the prior traits better define it as a message layer protocol.

The secure message contains besides the digital signature of the original message, a CbJxMessageInfo element with the source peer credential or else the peer PSE certificate, the source and destination addresses, and the source peer ID.

A digital signature is also produced from this cryptographic information block resulting to a final CBJX message with two signatures. The certificate inside validates the two signatures. CBJX encapsulates signatures by using a single Signature element containing a Base64-encoded PKCS#7 [106] binary signature.

```
<xs:complexType name="cbjx:CbJxMessageInfo">
  <xs:sequence>
    <xs:element name="PeerCert" type="xs:base64binary"/>
    <xs:element name="DestinationAddress" type="xs:string"/>
    <xs:element name="SourceAddress" type="xs:string"/>
    <xs:element name="SourceID" type="jxta:JXTAID"/>
  </xs:sequence>
</xs:complexType>
```

CBJX crypto-information XML schema

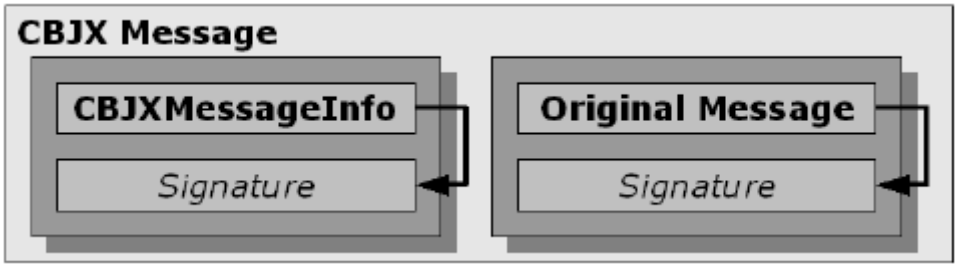

Figure 3.5: CBJX secure encapsulation

CBJX also provides some level of authenticity by using Crypto-Based Identifiers (CBIDs [107]). This method provides authentic messaging based on Self-signed certificates and not on certificates generated by a Trusted Third Party.

B. Wire transport layer security

TLS used on JXTA handles the streaming communications to be private, reliable, and authenticated in both ends. It both encodes and sends data through the network, and protects from both passive and active attacks [108].

TLS provides privacy via using symmetric cryptography for data encryption. Keys are generated for each connection respectively based on the Handshake protocol secret negotiations. TLS also provides integrity via using keyed message integrity checks (MACs) that in turn employ secure hash functions for MAC computations.

TLS is a binary protocol, but JXTA employs XML elements to transfer some of the binary content. The three element responsible for this task are: TLS Content is used to encapsulate secure data for transmission, Acknowledgements for data reception acknowledgements, and retries to resend messages due to failed transmission.

TLS and CBJX, provide information security only by protecting the JXTA transport protocol during transit. Once information is decapsulated and stored in the local peer, security is no longer provided. Also both transport methods only support end-to-end communications. In other words, applications based on multicast cannot use them.

CBJX authentic messaging method has no need to be under PSEmembership service. On the other hand TLS requires PSEmembership because its authentication handshake relies on PSE credentials. In addition messages are sent through TLS sockets. The combination of the two can provide data privacy, integrity and authenticity.

The authentication procedure in order to join a PSE PeerGroup can be summarized as follows:

1) User gives a personal password.
2) A keystore is created by writing the proper code lines and supplying a keystore password and provider credentials. The keystore password encrypts the place (keystore) where the certificate and password are saved.
3) Using the keystore password and the given PeerID, a Peer initializes Peer Group Authenticator.
4) Again with this information, the encrypted keystore in the local cache is located and opened.
5) User's certificate chain is retrieved.
6) That certificate chain allows the peer to join the PSEmembership using PeerGroup and becomes the group credential for that Peer.
7) Now the peer is able to interact with other peers in that same PeerGroup.
8) The password of the first step or private key and the certificate in the keystore are used when additional security is implemented (e.g. using TLS by choosing UnicastSecure type of pipes).

The authentication procedure for the PSE Membership Service follows local calls to JXTA libraries. For that reason network-based attacks (eavesdropping, traffic analysis, man-in-the-middle, and replay) are not considered threats to the PeerGroup joining procedure. A probable threat could be a local attacker.

In the procedure the participants are a) the user or some agent, b) the peer – application, c) the PeerGroup or other JXTA libraries which control group access. For that reason spoofing is limited to: a) user impersonation with unauthorized access to the keystore – the rest of the user's peers are in trouble, b) peer impersonation with unauthorized identity claim and credential generation – any PeerGroup identity can be claimed.

In the first case every peer's keystore is stored to its respective machine, and can be easily copied and distributed. Also access to it relies on the encryption mechanism and the password that were used.

In the second case peers under a PSEmembership service are authenticated only if they are able to access the local keystore. The validity of certificate and the certificate content is never checked. Peers can join and claim an identity for any group by only having the keystore password and a certificate. For those reasons the PSEmembership scenario is vulnerable to spoofing attacks and represents no real security on peer identity claims.

PSEmembership is more or less a base where developers can program their own preferred model based on securing identities via digital certificates. By its own the service is not a fully and trusted working model for the purpose. There must be additions such as e.g. perhaps a centralized CA and a method to guarantee private key authenticity.

When every peer can generate a certificate, a way to guarantee private key authenticity is essential. Even when a trusted centralized CA is in place this represents a central point of failure and further CA plans should be made with caution as not to threaten the p2p nature of the application and equality between the peers. Also man-in-the middle attacks can easily compromise such system targeting solely certificate generation procedures. Again there might be peers that claim identities and join the PeerGroup but will not have a properly signed certificate. These peers can be easily identified.

Typical user certificate in text format:

```
    Version: 3
    IssuerDN: O=www.jxta.org, L=SF, C=US, CN=secure-CA, OU=98FD486C4FBA4E9588C2
  Start Date: Wed Nov 28 13:55:54 PST 2001
  Final Date: Mon Nov 28 13:55:54 PST 2011
  SubjectDN: O=www.jxta.org, L=SF, C=US, CN=secure,OU=6FC145F38A4F70A89C02
  Public Key: RSA Public Key
    modulus:
843d01cc08ac4f024419870d2cdb769f46cb91d9222236cfce360f636a6b160edfc993150ded0737a
45b31835b09c2ae1767bd5b8a9ef5b95ec923d3a091775c4f60f037a67af55262bf6e05fe2062ea05
194a6e8ed73a78b2966fe49858d66abda1fe155dea2248b891ef8311b52926154d3a2ce4484dd0eb9
cd51eb797a0a1
public exponent: 11
  Signature Algorithm: SHA1WithRSAEncryption
          Signature: 8777029a34f42990226cfc94dc91c263111f354b
                     5a1efab5debf1e421f32b04c6f637a25d47752d5
                     a970e5126dbeda7f335ba40e65e3ff019b2775de
                     b8141dac322271fa1c296afac26bc1a1d0dba9cb
                     6cacfa06430a7f4eae508f46ee3a4416bdb3304a
                     b4f831c66b79338b3e83c57e9bf52bb498ca7b77
           3513409e74ba0ede
```

User Certificate as it appears in the peer advertisement (ASN.1 DER Encoded and in Base64)

```
-----BEGIN CERTIFICATE-----
MIICNTCCAZ6gAwIBAgIBATANBgkqhkiG9w0BAQUFADBkMRUwEwYDVQQKEwx3d3cu
anh0YS5vcmcxCzAJBgNVBAcTAlNGMQswCQYDVQQGEwJVUzESMBAGA1UEAxMJc2Vj
dXJlLUNBMR0wGwYDVQQLExQ5OEZENDg2QzRGQkE0RTk1ODhDMjAeFw0wMTExMjgy
MTU1NTRaFw0xMTExMjgyMTU1NTRaMGExFTATBgNVBAoTDHd3dy5qeHRhLm9yZzEL
MakGA1UEBxMCU0YxCzAJBgNVBAYTAlVTMQ8wDQYDVQQDEwZzZWN1cmUxHTAbBgNV
BAsTFDZGQzE0NUYzOEE0RjcwQTg5QzAyMIGbMAsGCSqGSIb3DQEBAQOBiwAwgYcC
gYEAhD0BzAisTwJEGYcNLNt2n0bLkdkiIjbPzjYPY2prFg7fyZMVDe0HN6RbMYNb
CcKuF2e9W4qe9bleySPToJF3XE9g8DemevVSYr9uBf4gYuoFGUpujtc6eLKWb+SY
WNZqvaH+FV3qIki4ke+DEbUpJhVNOizkSE3Q65zVHreXoKECAREwDQYJKoZIhvcN
AQEFBQADgYEAh3cCmjT0KZAibPyU3JHCYxEfNUtaHvq13r8eQh8ysExvY3ol1HdS
1alw5RJtvtp/M1ukDmXj/wGbJ3XeuBQdrDIicfocKWr6wmvBodDbqctsrPoGQwp/
Tq5Qj0buOkQWvbMwSrT4McZreTOLPoPFfpv1K7SYynt3NRNAnnS6Dt4=
-----END CERTIFICATE-----
```

In current JXTA, advertisements may be secured by employing digitally signing at application level to create Signed Advertisements. Digital signature directly on the advertisement, may support both local and remote publication via propagation to multiple peers.

The PSE Membership Service must be the membership service on the group to allow usage of Signed Advertisements, since information such as cryptographic keys that generate and validate the signature are obtained from its associated keystore and

credential. Signed advertisements will only be sent to members of that group and only inside the group's boundaries.

```
<xs:element name="SA" type="jxta:SA"/>
  <xs:complexType name="jxta:SA">
  <xs:sequence>
    <xs:element name="PSECred" type="jxta:PSECred"/>
    <xs:element name="jxta:Signature" type="base64binary"/>
    <xs:element name="jxta:Advertisement" type="base64binary"/>
  </xs:sequence>
</xs:complexType>
```

Signed Advertisement XML schema

As far as Access Control mechanisms the current JXTA reference implementation offers three kinds of access control, each one bound to each different membership service credential type:

- The Always Access Service, which is the default Access service for peer groups and allows any operation without checking access control.
- The simpleACL Access Service uses Access Control Lists to distinguish which identities are allowed to perform which operations. The Access Control Lists are distributed as parameters within the peer group advertisement.
- The PSE Access Service validates all credentials against a previous set trust anchor to decide whether to permit the operation or not. It provides an interface to PKIX certificate path validation.

Current Access Service approaches do not check whether an identity belongs to the peer group but only if that identity is permitted to access some service. Since the membership service does not do that either and any peer may claim any identity, this is left on JXTA developers to manage.

In addition, the access service just checks credential content which is not sufficient to guarantee protection against spoofing, since credentials are publicly exchanged across the network. A method such as TLS or CBJX which will test credential authenticity must exist.

Chapter 4
Design and Analysis

4.1 NetBuckler

The current implementation is based on [14] and is named NetBuckler. The application's architecture follows:

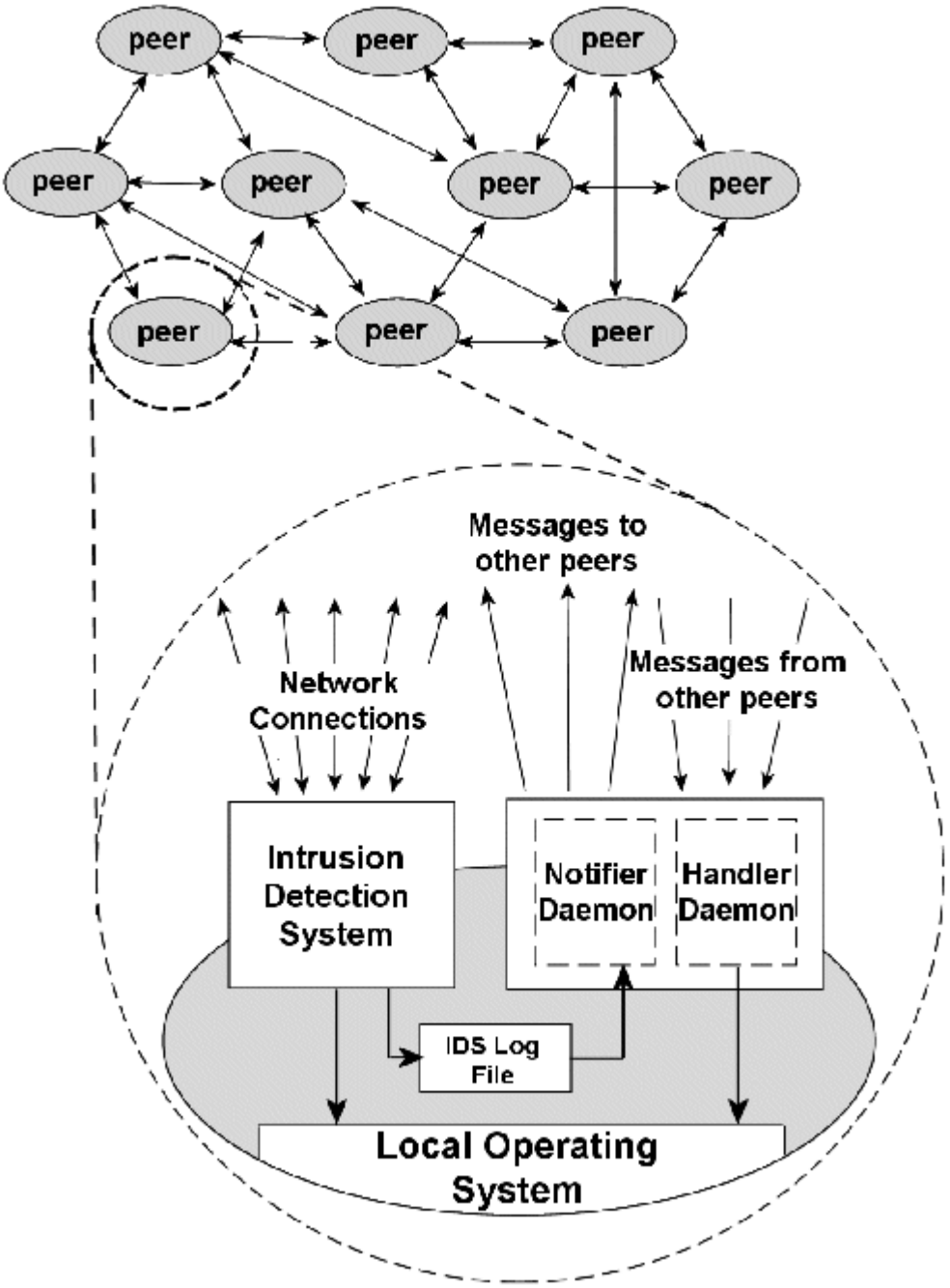

Figure 4.1: Architecture of the implementation system [14].

As in the figure 4.1 there is a P2P network of peers in which every peer runs the program. There is a log file in every peer with Internet traffic information to that peer. The application utilizes that log file in order to make decisions as if to enforce or withdraw protection measures to the host Operating System. Also there are two daemons that handle communication operation. The Notifier daemon sends

information to other peers and reads the log file while Handler daemon receives information from others.

More analytically:

A. The Notifier daemon monitors the peer where it runs. What really does, is reading a log file originating from the operating system or other source (ISD, firewall, antivirus) with traffic information. At regular fixed times it checks network traffic destined to the peer from the log file. This information may reveal (e.g. if inexplicably high) a probable malware outbreak to the Internet, and therefore a probable imminent security breach to the system.

In other words, the node's n Notifier at regular fixed time intervals t will record the number of attack hits (h^n_t) that happened to the node the past interval. The p^n_t will be then calculated and send. This is the percentage of increase or decrease in attacks during the current interval t on node n. In addition, k(> 0) is the size of the 'time window' used in the number of t time intervals which the malicious activity rate is calculated as described in[14] according to the formula:

$$p_t^n = \frac{h_t^n - \frac{\sum_{i=t-k}^{t-1} h_i^n}{k}}{\frac{\sum_{i=t-k}^{t-1} h_i^n}{k}}$$

The fitting value of time interval t will be set to 10 minutes if we consider the time needed for the network traffic to be increased and the time spent in order the malware outbreak to be noticed. A value of p^n_t that exceeds by far 1.0 means that node n is under siege and it is probably threatened by a worm during the time interval t. Threshold value can have a value that best suits the node that sets it. Having a small value will probably false interpret a slight increase in malware activity. If this is introduced by a lot of peers, it may lead to wrong taken decisions on activating the countermeasures. In turn this may cause frustration to the users because of the enforced limitations to every system and to applications on those systems. Increasing it to a high value, it will most probably fail to detect a worm outbreak even if it really

happens. Selecting a large threshold would be a great option because malware epidemics cause high traffic on most of the cases that is difficult to be unnoticeable.

B. The Handler daemon, receives the messages from other nodes (Notifier sends them), and takes the countermeasures when the time calls. According to the formula [14]:

$$p_{avg} = \frac{\sum_{i=1}^{n} p_t^i}{n}$$

Handler produces the overall difference in arrack rate coming from the average attack rate of all n nodes of the peer group that have sent a message the last interval. If p_{avg} surpasses a predefined threshold top value, the program triggers the rmeasures to be taken on the system. If there is the phenomenon where it drops low then the system is reconfigured back to its former state.

The correct threshold is very significant to the process because security level fluctuations need to be made when needed most and not left or set earlier or after a long period of time. We target to immunize our system so as not to be victims, and for that reason selecting the appropriate threshold values T_{high} and T_{low} is crucial.

- If $p_{avg} > T_{high}$, then increase security policy.
- If $p_{avg} < T_{low}$, then decrease security policy.
- If $T_{low} \leq p_{avg} \leq T_{high}$, do nothing.

4.2 The tools and libraries

NetBeans (Version 7.4) Integrated Development Environment was used along with JDK Version 1.7.0_25. The JXTA 2.6 P2P framework was also used as opposed to JXTA 2.7 due to problems with the latest while loading PSEmembership service for the PeerGroup. For the measures part, batch files were created and custom code was written.

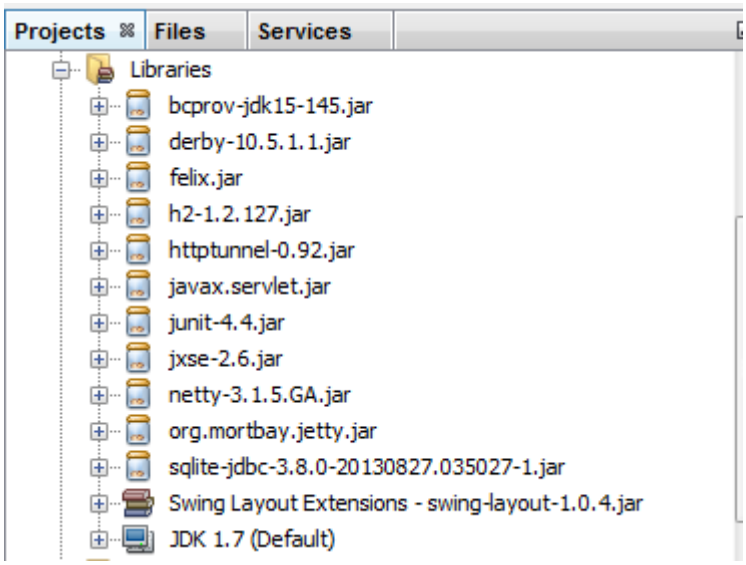

Figure 4.2: Project libraries

- The Bouncy Castle Crypto package (bcprov-jdk15-145.jar) [109] is a Java implementation of cryptographic algorithms. It contains a light-weight API suitable for use in any environment.
- Derby (derby-10.1.1.jar) [110] is a pure Java relational database engine using standard SQL and JDBC as its APIs.
- Apache felix (felix.jar)[111] is an open source implementation of the OSGi core framework specification. In other words, it is a module system and service platform for Java that implements a complete and dynamic component model, that was not available in standalone/Virtual Machine environments until now.
- H2 (h2-1.2.127)[112] is an open source java SQL database.
- JXTA 2.6 (jxse-2.6)[113] is the core library of the P2P framework.
- JUnit (junit-4.4.jar) [114] is a unit testing framework for Java

- Netty (netty-3.1.5.GA.jar) [115] provides asynchronous event-driven network application and tools for rapid development of maintainable high performance and high scalability protocols servers and clients.
- The Jetty Web (org.mortbay.jetty.jar) [116] Server provides an HTTP server and Servlet container capable of serving static and dynamic content.
- SQLite (sqlite-jdbc-3.8.0-20130827.035027-1.jar) [117] implements a self-contained, serverless, zero-configuration, transactional SQL database engine.
- Swing Layout Extensions (swing-layout-1.0.4.jar) extends the layout capabilities of Swing. The generated GUI code needs its classes to execute.

Although more than needed libraries are included, those used by the application are listed to the table:

Library	Description	Dependencies
JXSE_2.6.jar	JXTA 2.6 core library http://jxta.kenai.com/	bcprov-jdk15-145.jar; h2-1.2.127.jar; org.mortbay.jetty.jar; netty-3.1.5.GA.jar; httptunnel-0.92.jar
bcprov-jdk16-145.jar	Used by JXTA for encryption	
h2-1.2.127.jar	Used by JXTA for H2 implementation of the cache manager	
org.mortbay.jetty.jar	Required by JXTA HTTP transport	javax.servlet.jar
javax.servlet.jar	Required by Jetty	
netty-3.1.5.GA.jar	Required by JXTA TCP transport	
httptunnel-0.92.jar	Required by JXTA	
sqlite-jdbc-3.8.0-20130827.035027-1.jar	Used for manipulating the database	

Table 4.1: Project required libraries

4.3 Class diagrams

Below are the class diagrams of the implemented application. The classes in the diagrams are described above in 3.2.1 and 3.3.1. The UML diagrams were created and exported using JUG (Java UML Generator) a freeware UML generator designed for Java downloaded from SourceForge.net [118]. The results for all the classes that were used are the following:

Package netBuckler

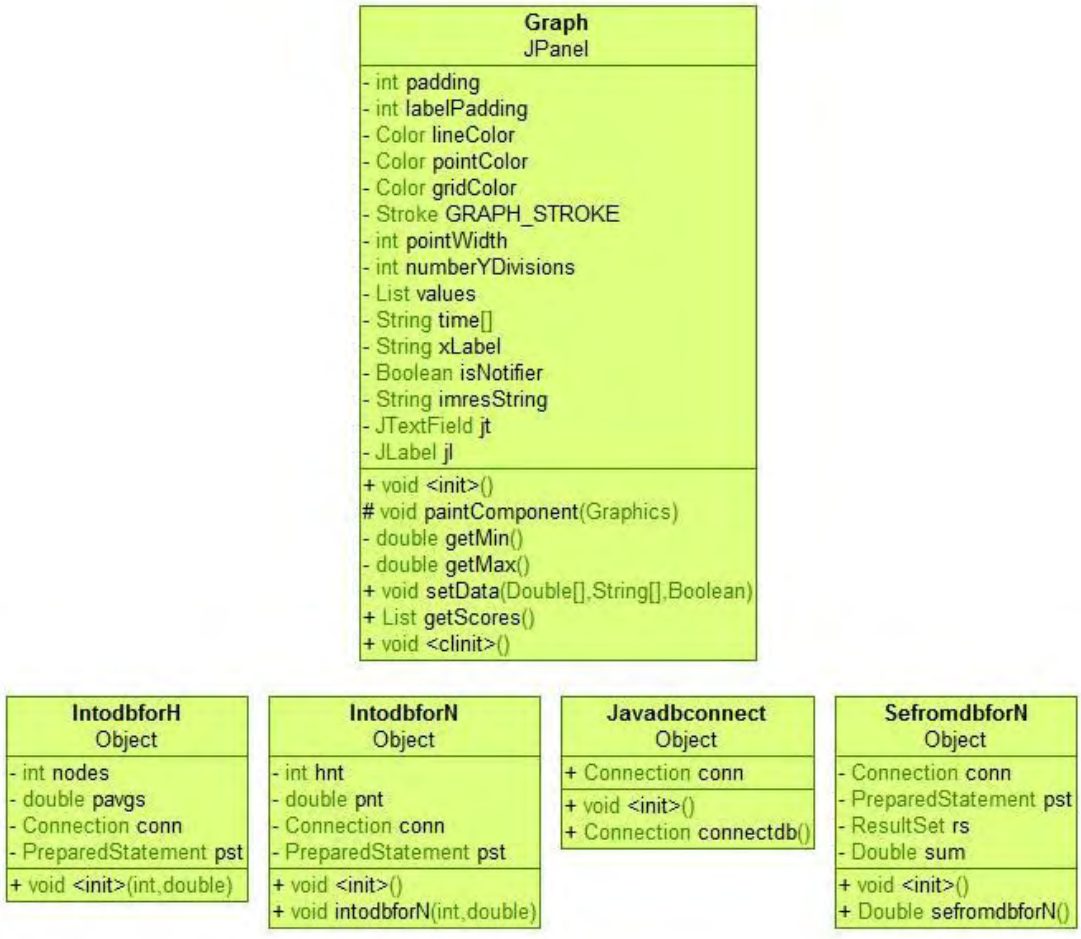

Figure 4.3: UML diagrams of Graph, IntodbforH, IntodbforN, Javadbconnect, SefromdbforN

Figure 4.4: UML diagrams of Handler, Notifier, ListPeers

Figure 4.5: UML diagram of MainForm

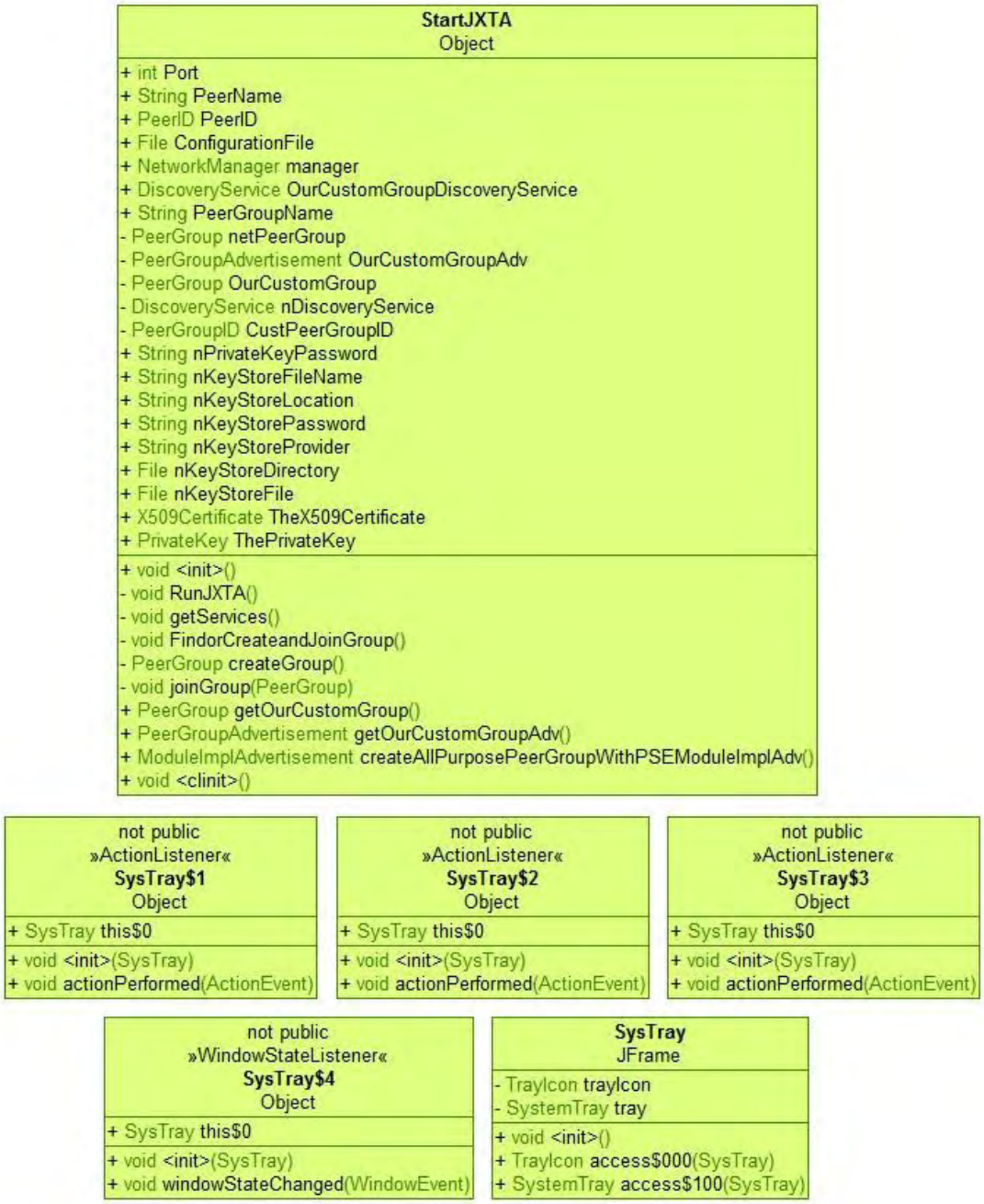

Figure 4.6: UML diagrams of startJXTA, SysTray

Figure 4.7: UML diagrams of Measures, Tools

4.4 Technical details

4.4.1 JXTA Specific

JXTA has all of its services available to all peers connecting to JXTA network via WorldPeerGroup and NetPeerGroup. Although this does not mean that those services are also activated. The services that are mentioned to the module implementation advertisement are those that can be used by peers that call newGroup(…).

As far as network scope, relay peers are needed to overcome artificial network boundaries. Peers with no unique IP addresses and/or located behind NATs and/or probable HTTP use only communication limitations urge for the use of such super peers that have unique public IP addresses. Those peers will be set as seeds for edge peers that run the JXTA application.

Equally important are also the rendezvous peers. They help with the propagation of messages and queries between peers in a PeerGroup that cannot be reached by multicasting. The number of rendezvous peers in a PeerGroup is defined by the following three factors: a) There must be peers that help with the propagation to improve QoS, b) there may be peers that decide to became rendezvous from edge peers to the group they are connected, c) there must be a certain number of connections to the rendezvous peers they can accept to increase QoS. We must recall that rendezvous peers regularly send random rendezvous lists of those they are aware of, to a set of those known rendezvous peers. Those that they do not respond are removed from the lists.

The need for rendezvous and relay seeds is crucial. They both facilitate connectivity of JXTA-enabled peers over the internet. Such seeds require a static ip and NAT translation must be avoided to packets that reach and leave those seeds. In addition seeds allow connectivity to peers even from remote locations where proper configuration to new routers, NATs and firewalls may not exist.

There is no such restriction that prevents a peer from participating to more than one PeerGroups. Also it is important that there is a proper mechanism that prevents rendezvous peers from running out of leases to accept connections. A probable solution would be to create a set of rendezvous for edge peers and a set for rendezvous peers.

As far as connectivity of JXTA applications, default JXTA IP ports range from 9701 – 9799. If there are more than one application running on the same machine, more than one of these ports will be used. Firewalls should be configured to allow communication over these ports. Peers should be able to communicate with relay peers to reach others located behind NATs and firewalls if not on the same LAN. Also NetworkConfigurator can be used to allow other ports and port ranges.

Multicasting is performed to 224.0.1.85 and 1234 port for JXTA-enabled peers. If there are additional firewalls on every machine, there should be configured to allow multicasting. In a case where there are subnets in a network, there must be proper configuration to routers to allow peers from one subnet to communicate with those on the other with the use of their local IP addresses. Also multicast should be enabled to prevent the need for rendezvous peers.

4.4.2 Application Specific

The Graph.java code that prints the Graphs was taken from [119] and is slightly modified. In addition, in the same way the SysTray.java code [120] which sends the application to system tray when minimized. Tools.java was kept from the code that accompanies [101] to mostly help in popping up messages. IntodbforH.java IntodbforN.java are used to save information into the database (NetBuckler) for the Handler and Notifier respectively. SefromdbforN.java queries the database for information. Furthermore Javadbconnect.java contains java connection information and code. Handler.java and Notifier.java both contain code for the programs daemons and also implement the formulas respectively. ListPeers.java contains code that searches for peer advertisements and lists the names of those found to a Jlist on the GUI. Also importCertificate() code from [121] can be used to extend PSEMembership service's security features. MainForm.java initializes the program's GUI and its features when the start button at log tab is pressed. Measures.java finds windows version, displays the correct service names on the GUI, and executes command prompt commands through java code. Also it runs the batch files. StartJXTA.java starts the JXTA network with all the specifications in the written code.

The application comprises of four tabs (Program Configuration, Log, List Of Peers, About). The user is supposed to fill the application's parameters in the Program Configuration tab and decide which services to stop in case of a worm spread by

reading every services description and according to his preferences. The displayed services are operating system specific which means that are based on the host operating system that the application is running. The application automatically detects the host operating system and fills the service list from a .txt file. As far as the rest configuration, the user may decide to change the log file default path which by default is configured to read the windows firewall pfirewall.log file. Also the name of the peer which takes the name of the current user, the time window k default value which is 4 time intervals, the default threshold values Thigh and Tlow with 1 and 0 values respectively, the port number of accepting messages 9726 and the default Group Name. The user may only decide to configure the peer type and fill the local IP and global IP fields if the peer choice is a super peer, inside a LAN and behind a NAT. The threshold values configuration is of high importance because they affect the application's efficiency. If the peer usually has high fluctuations in traffic rate and a low Thigh value is configured, then false information will be sent to peers on the custom group. Also the port number should be changed for peer running inside the same LAN. From the log tab, user may decide to start or stop the application and even stop and change configuration and start over the application. The communication between peers, the messages sent and received, as long as the peer names that communicate the messages are also displayed. The list of peers tab displays the names of the peers in the custom group. The about tab displays information about the application and instructions how to run it.

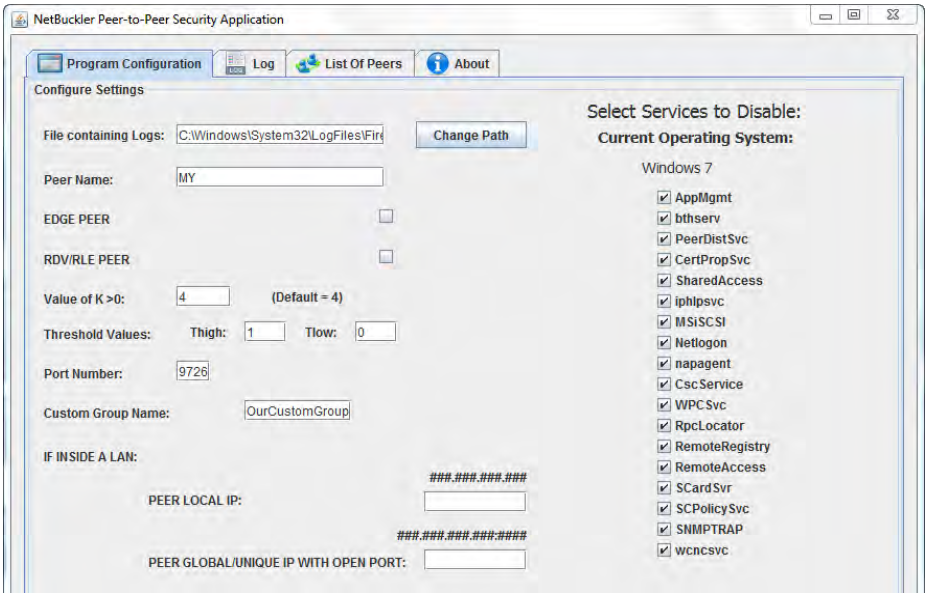

Figure 4.8: NetBuckler Program Configuration tab

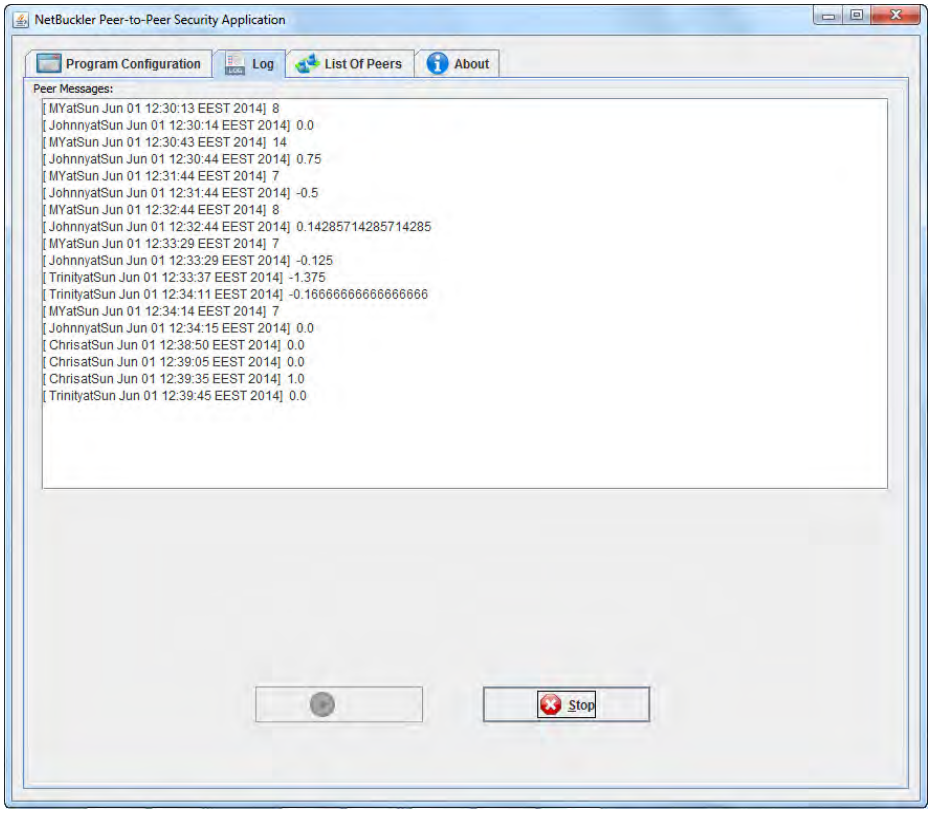

Figure 4.9: NetBuckler Log tab

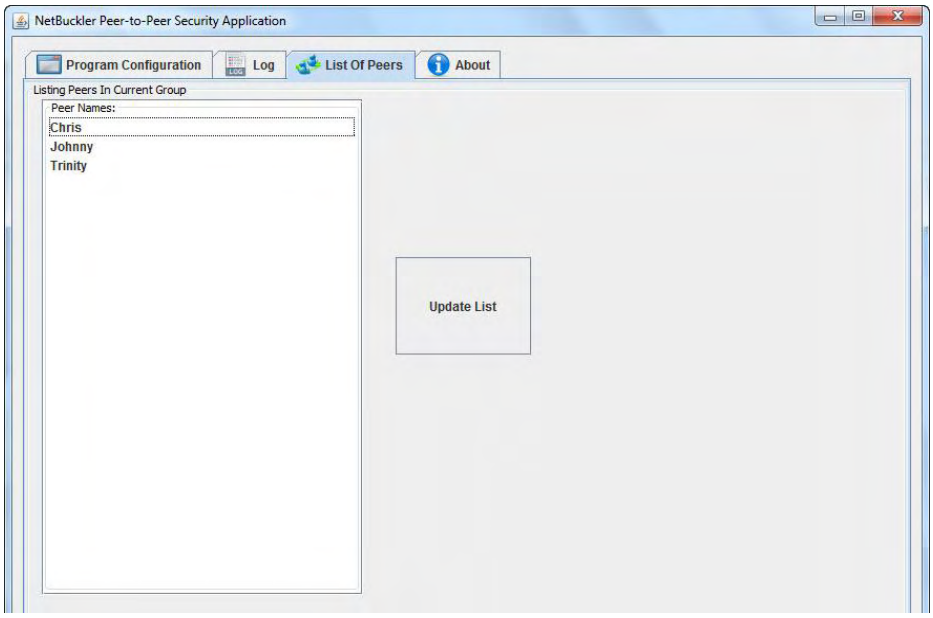

Figure 4.10: NetBuckler List Of Peers tab

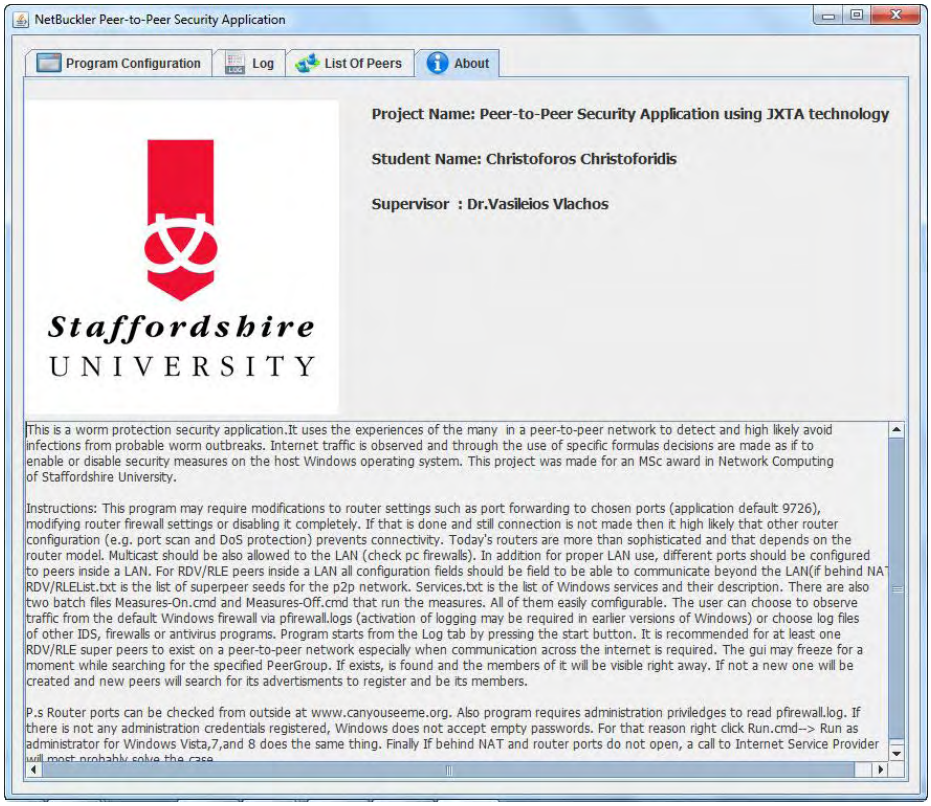

Figure 4.11: NetBuckler About tab

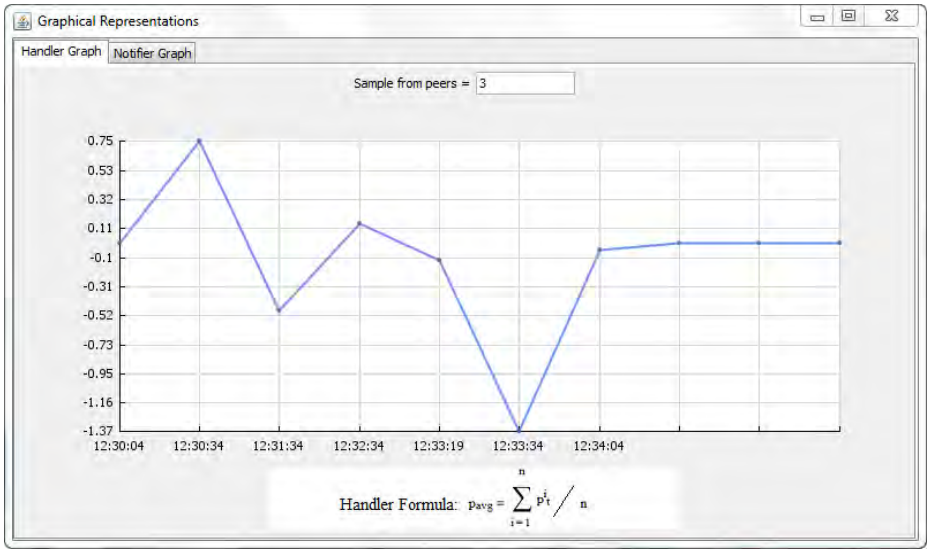

Figure 4.12: Handler Graph

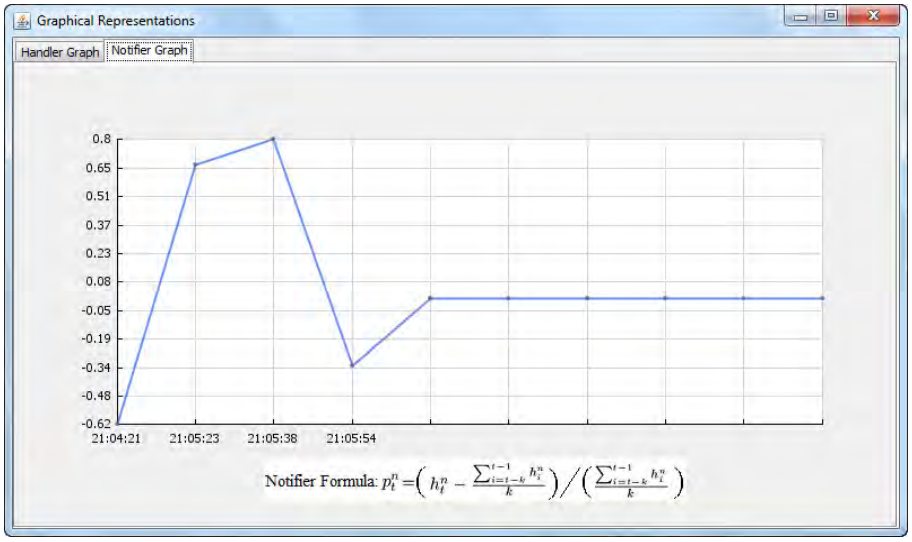

Figure 4.13: Notifier Graph

The user may choose to read traffic from the default Windows firewall via pfirewall.log (activation of windows firewall logging may be required in earlier versions of Windows) or choose log files of other IDS, firewall or antivirus programs. The application searches for the specified PeerGroup and if exists, it is found and the members of it will be visible right away especially when connected to a RDV/RLE peer. If not a new one will be created and new peers will search for its advertisements to register and be its members. There are graphical representations for both Notifier and Handler values.

Inside the application folder besides Services.txt which contains the services and information about them for the GUI, there is RDVRLEList.txt. This .txt file contains a list of super peers IPs which will be used by the client to connect to. Both files are easily modified to include super peer IPs, change services and information about them. Also there are two batch files, Measures-On and Measures-Off, containing commands on enforcing measures and withdrawing them respectively. Measures-On removes installed browser specific links (Mozilla firefox, Google Chrome, Internet Explorer) from both desktop and taskbar and places new ones with browsing restrictions. The new .lnk files are copied from 86lnks and lnks folders in the same directory both for 64bit and 32 bit systems while the old are moved to destemp (desktop moved .lnks) and tastemp (taskbar moved lnks) folders in C:/ directory. There are also additional commands to backup registry, edit it with .reg files from regfiles folder for both windows 7 and windows 8 and kill explorer, all the upper after

an operating system check is made. On the other hand Measure-Off ensures that the previous enforced measure are taken back. Both batch files are not affecting any services which are handled from a java class separately.

The application has the choice of making super peers (RDV/RLE) to assist communications. It is recommended for at least one RDV/RLE super peers to exist on a peer-to-peer network especially when communication across the internet is required. A peer's choice to change to rendezvous is also available by default. In addition, there is no restriction to the connections a peer can accept. Multicast is by default enabled. As far as security we have joined a custom group that runs under the PSE membership implementation after following the steps discussed earlier to connect. This does not provide the proper security but is a step for further implementation on the scheme. Messages are exchanged using a single pipe both for sending and receiving that is activated from a worker thread every certain time defined in its sleep duration. Also messages are sent in a propagate manner defined in the pipe advertisement which allows no security for those messages.

For internet and LAN connectivity configuration may be required to router settings such as port forwarding to chosen ports (application default 9726), modifying router firewall settings or disabling it completely. If these are done and still connection is not made then it high likely that other router configuration (e.g. port scan and DoS protection) prevents connectivity. Today's routers are more than sophisticated and that depends on the router model. Multicast should be also allowed to the LAN (check pc firewall).

This version of the application is compatible only with Microsoft Windows operating systems. The major parts of the program consist of the P2P network, and the body (Handler, Notifier). These parts are independent of operating systems because they are built in java language. So we understand that in order this program to be compatible with other operating systems besides Microsoft's Windows minor modifications are required.

4.5 Testing and results

The application was tested in order to get feedback of proper function. The prime objectives of communication inside and outside a LAN have been confirmed. In addition messages are both sent and received without problems. The results coming from the formulas are correct and measures are correctly enforced upon exceeding Threshold upper or lower values. Graphs for both daemons are correctly displaying results. Pfirewall.log windows log.txt can be accessed if the program is "run as administrator".

Scenario 1: LAN – EDGE and RDV/RLE

With disabled firewalls on either PCs, or allowed multicast via configuration we run the program with the following parameters:

PEER 1:

File containing Logs:	We create a log.txt and select it
Name:	Trinity
EDGE PEER	
RDV/RLE PEER	√
Value of K>0	4
Threshold Values	1-0
Port Number:	9726
Custom Group Name:	OurCustomGroup
IF INSIDE A LAN:	Both empty

Table 4.2: Scenario1 peer1

In the command prompt →ipconfig →IPv4 Address

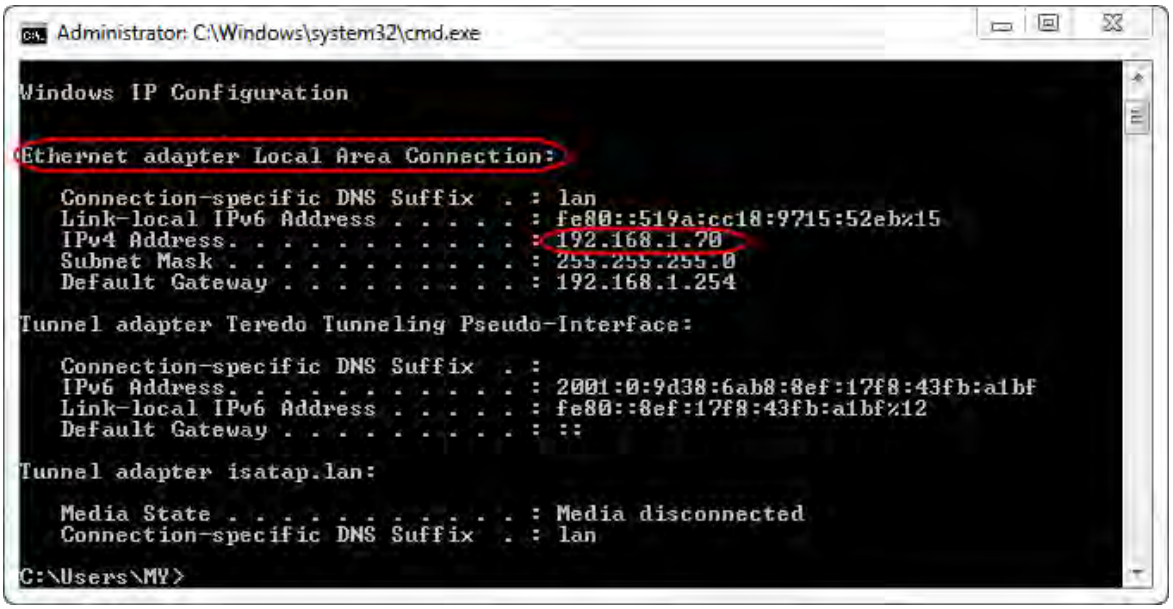

Figure 4.14: Get local IP

PEER 2:

File containing Logs:	We create a log.txt and select it
Name:	MY
EDGE PEER	√
RDV/RLE PEER	
Value of K>0	4
Threshold Values	1-0
Port Number:	9727
Custom Group Name:	OurCustomGroup
IF INSIDE A LAN:	Both empty

Table 4.3: Scenario1 peer2

Open RDV/RLEList.txt →192.168.1.70:9726

Scenario 2: LAN - 2 EDGEs

With disabled firewalls on either PCs, or allowed multicast via configuration we run the program with the following parameters:

PEER 1:

File containing Logs:	We create a log.txt and select it
Name:	Trinity
EDGE PEER	√
RDV/RLE PEER	
Value of K>0	4
Threshold Values	1-0
Port Number:	9726
Custom Group Name:	OurCustomGroup
IF INSIDE A LAN:	Both empty

Table 4.4: Scenario2 peer1

PEER 2:

File containing Logs:	We create a log.txt and select it
Name:	MY
EDGE PEER	√
RDV/RLE PEER	
Value of K>0	4
Threshold Values	1-0
Port Number:	9727
Custom Group Name:	OurCustomGroup
IF INSIDE A LAN:	Both empty

Table 4.5: Scenario2 peer2

It should be mentioned that communication of two EDGE peers may not always be possible if both attempt it wirelessly. The existence of at least on super peer is recommended to facilitate communication inside and beyond a LAN that it wouldn't be possible otherwise.

Scenario 3: Internet – 2 EDGEs on the same LAN and 1 RDV/RLE

With disabled firewalls on both PCs, or allowed multicast via configuration LAN Peers run the program while the Internet Peer must configure his router first (open his preference port 9726 and disable firewall):

LAN PEER 1:

File containing Logs:	We create a log.txt and select it
Name:	Trinity
EDGE PEER	√
RDV/RLE PEER	
Value of K>0	4
Threshold Values	1-0
Port Number:	9726
Custom Group Name:	OurCustomGroup
IF INSIDE A LAN:	Both empty

Table 4.6: Scenario3 peer1

Open RDV/RLEList.txt →178.59.126.201:9726

LAN PEER 2:

File containing Logs:	We create a log.txt and select it
Name:	MY
EDGE PEER	√
RDV/RLE PEER	
Value of K>0	4
Threshold Values	1-0
Port Number:	9727
Custom Group Name:	OurCustomGroup
IF INSIDE A LAN:	Both empty

Table 4.7: Scenario3 peer2

Open RDV/RLEList.txt →178.59.126.201:9726

Peers 1 & 2 are on the same LAN while Peer 3 is somewhere in the internet.

Internet PEER 3:

File containing Logs:	We create a log.txt and select it
Name:	Man
EDGE PEER	
RDV/RLE PEER	√
Value of K>0	4
Threshold Values	1-0
Port Number:	9726
Custom Group Name:	OurCustomGroup
IF INSIDE A LAN:	PEER LOCAL IP: 192.168.1.70 PEER GLOBAL WITH PORT : 178.59.126.201:9726

Table 4.8: Scenario3 peer3

In the command prompt →ipconfig →IP address (the same as in Figure 4.12).

Visit e.g. www.whatismyip.com →e.g. result: 178.59.126.201

Scenario 4: Internet – 2 EDGEs on different LANs and 1 RDV/RLE

Internet Peer must configure his router first (open his preference port 9726 and disable firewall):

LAN PEER 1:

File containing Logs:	We create a log.txt and select it
Name:	Trinity
EDGE PEER	√
RDV/RLE PEER	
Value of K>0	4
Threshold Values	1-0
Port Number:	9726
Custom Group Name:	OurCustomGroup
IF INSIDE A LAN:	Both empty

Table 4.9: Scenario4 peer1

Open RDV/RLEList.txt →178.59.126.201:9726

LAN PEER 2:

File containing Logs:	We create a log.txt and select it
Name:	MY
EDGE PEER	√
RDV/RLE PEER	
Value of K>0	4
Threshold Values	1-0
Port Number:	9726
Custom Group Name:	OurCustomGroup
IF INSIDE A LAN:	Both empty

Table 4.10: Scenario4 peer2

Open RDV/RLEList.txt →178.59.126.201:9726

Peers 1 & 2 are not on the same LAN and Peer 3 is somewhere in the internet.

Internet PEER 3:

File containing Logs:	We create a log.txt and select it
Name:	Man
EDGE PEER	
RDV/RLE PEER	√
Value of K>0	4
Threshold Values	1-0
Port Number:	9726
Custom Group Name:	OurCustomGroup
IF INSIDE A LAN:	PEER LOCAL IP: 192.168.1.70 PEER GLOBAL WITH PORT : 178.59.126.201:9726

Table 4.11: Scenario4 peer3

In the command prompt →ipconfig →IP address (the same as in Figure 4.12).

Visit e.g. www.whatismyip.com →e.g. result: 178.59.126.201

Scenario 5: Internet – 4 EDGEs on the same LAN, 1 RDV/RLE, 2 Internet EDGEs

EDGE peers on the same LAN have different port numbers while Internet EDGEs can be configured with the same 9726 default port. There is one RDV/RLE peer and all other peers are connected to that super peer.

Our application can be configured to work with windows firewall and every other firewall that exports log files. A simple test was made locally that proved our claims. [122] is a configuration guide for the windows firewall log.

Chapter 5

Conclusions

5.1 Aims of the Thesis

1. A worm defender program based on an alternative way of defending [14] will be provided to enhance fighting against those most dangerous threats.
    a. JXTA latest edition peer-to-peer open source framework will be investigated, used and analyzed.
    b. Java programming language will be used which is open source, widely supported and used (computers, mobile devices, etc.). This means that the program will require minor modifications (e.g. on the measures part, stop Linux daemons etc.) to be able to run for Linux distributions.
    c. Windows probable security weak points of software will be looked into.
    d. Measures will be investigated and taken to avoid probable weak points that may compromise the Windows operating system.
        i. Registry modifications
        ii. Services to be stopped
2. A framework for evaluation of the performance of the end system will be built.
3. Clear statements of any constraints and restrictions of the worm security application will be made.

4. All in all, the target is to acquire the knowledge and skills around that subject and develop a piece of software using current tools that will as close to ready as it can be for public use.

5.2 Evaluation

Overall the program is a working solution for defending against worms. But this does not make it a good solution at this stage. Security of the implementation result may be enhanced to the point where there will be user authentication, either by using signed certificates by a trusted CA, or by requiring credentials alone. Data privacy is also important and should be carefully planned and implemented in order to preserve efficiency. This first step on this could be to enable TLS (UnicastSecure pipes) under the PSE membership service and create a user authentication mechanism based on certificates. PSE membership service is currently used for the created groups. In addition, the system must be also tested on a large scale and decisions must be made on the number of connections a peer must accept.

5.2.1 Literature Review

For the implementation of this Thesis there were various topics that needed to be studied, discussed and brought together. At first malware and specifically worms were studied. Worm characteristics were analyzed in order to better understand them and get the motivation and knowledge to implement an application based on them. The next step was to acquire better understanding and knowledge of the JXTA P2P framework that would be used through many books, programming guides and papers that were read. The JXTA puzzle had to be solved both on theory and on coding. Security of the application was very important because a future approach on security and data privacy may compromise efficiency. The application must be efficient, robust and secure. Windows issues and measures had to be surveyed for information on the Internet and on papers to conclude to a lightweight solution on the measures part.

5.2.2 Proposed System

The developed system uses current technologies, is based on [14] and is built in Java, using Java JXTA peer-to-peer framework libraries as a base of creating the P2P networks that the application will be based on. The program reads the inputs from a text file that saves information representing traffic to the host. The percentage of traffic increase or decrease is calculated according to a formula and sent to peers on the P2P network. Every peer calculates the average of the its received messages. If the average surpasses a certain threshold the application enforces measures (close services, disable browser add-ons, registry modifications) on the Windows operating system to protect it from getting infected. These measures limit the system to a constrained working order, closing services and allowing browsing using some of the available resources. If the sequence of the dropped packets is decreased under a certain threshold then the application returns to its previous state. The whole operation of the application is based on unusual, unexpected traffic increase to peers of the P2P network that is most probably caused by a worm scanning for targets mechanism. The peers communicate this knowledge and quickly take measure to evade infection.

The developed system is tested and evaluated according to several criteria. The program works and responds well in small scale tests. It can communicate both on LAN and over the Internet and print results to the Graphs. Measures are not affecting much user's experience and both execution time and transmission times of the messages suffice. There is no real security on the user authentication and on communicated messages but this can be handled on later versions of the program. The base of the current application is a great foundation and the potential with java and JXTA are many.

5.3 Recommendations for Future Research

This thesis can be the basis for a new open source project that would intrigue many developers to use their skill and ideas as it is simple, easily expendable and portable to fit other operating systems. This is not a client/server system but is based on powerful P2P JXTA technology. Every user will have a visualization of traffic activity to the host machine and a sense of greater security towards unknown threats. Those traits will make it a fast success. In addition the security of exchanged messages and

especially user authentication mechanisms must be the topic for future research to increase application's security and efficiency. Additional enhancements could be techniques such as load balancing which helps distribute workload to multiple servers to speed up detection, or trust management which helps identify dishonest nodes by monitoring their past behavior.

5.4 Conclusions

Protection against worms has received much attention in the recent years. Existing implementations are intricate, suffering from the client/server model, with scalability and fault tolerance issues, either constrained to LANs or not, even if applying layering techniques or other such techniques, they lack of potentials if not using the P2P model. This thesis has provided an implementation of a worm security application based on [14]. The produced system is functional but not secure. Additional testing must be done on large scale and application's user authentication and security must be looked upon. However, the outcome of this thesis can be used for further investigation and testing on securing applications using JXTA platform. It can be continued as an open source project with high potential. Finally, the use of Java and JXTA allow implementations on other computing devices such as mobile phones and tablets where security lacks of solutions and needs more attention nowadays.

Bibliography

[1] Szor, P. (2005), *The Art of Computer Virus Research and Defense*, Upper Saddle River, New Jersey: AddisonWesley.

[2] Nazario, J. (2004), *Defense and Detection Strategies against Internet Worms*. Artech House computer security series.

[3] Staniford, S., Paxson, V., and Weaver, N. (2002), 'How to own the internet in your spare time', in *Proceedings of. 11th USENIX Security Symposium*, pp. 149–167.

[4] Erbschloe, M. (2005), *Trojans, worms and spyware. A computer security professional's guide to malicious code,* Oxford, UK: Elsevier Butterworth–Heineman press.

[5] Grier, D. (2006), 'The benefits of being different', *IEEE Computer* **39**(11), 6–8.

[6] Moore, D., Shannon, C., and Brown, J. (2002), 'CodeRed: a case study on the spread and victims of an internet worm', in *Proc. Internet Measurement Workshop*.

[7] Moore, D. and Shannon, C. 'The spread of the codered worm (crv2)', [Online]. http://www.caida.org/research/security/code-red/coderedv2_analysis.xml

[8] Concept Virus(CV) V.5. (2001), 'Nimda worm virus report–final', [Online]. http://www.di.unisa.it/~ads/corso-security/www/CORSO-0102/NIMDA/link_locali/nimda-update-sept27.pdf

[9] Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., and Weaver, N. (2003), 'Inside the slammer worm', *IEEE Security & Privacy*, 33–39.

[10] Bailey, M., Cooke, E., Jahanian, F., Watson, D., and Nazario, J. (2005), 'The blaster worm: Then and now', *IEEE Security & Privacy* **3**(4), 26–31.

[11] Shannon, C. and Moore, D. (2004), 'The spread of the witty worm', *IEEE Security & Privacy* **2**(4), 46–50.

[12] Gutmann, P. (2007), 'World's most powerful supercomputer goes online', [Online]. http://seclists.org/fulldisclosure/2007/Aug/0520.html

[13] Microsoft. (2004), 'Help protect yourself from the Conficker computer worm' [Online]. http://www.microsoft.com/security/pc-security/conficker.aspx

[14] Vlachos, V., Androutsellis-Theotokis, S., and Spinellis, D. (2004), 'Security Applications of Peer-to-Peer Networks', *Computer Networks* **45**(2), 195–205.

[15] Ludwig, M. (2002), *The Little Black Book of Email Viruses*, Panama City, Republic of Panama: Lexington & Concord Partners.

[16] Holz, T. (2005), 'A short visit to the bot zoo', *IEEE Security & Privacy* **3**(3), 76–79.

[17] Holz, T. (2005), 'Spying with bots', *login* **30**(6), 18–23.

[18] Reiher, P., Li, J., and Kuenning, G. (2004), 'Midgard worms: Sudden nasty surprises from a large resilient zombie army', *UCLA Computer Science Department Report CLACSD040019*.

[19] Geer, D. (2006), 'Hackers get to the root of the problem', *IEEE Computers* **39**(5), 17–19.

[20] Young, S. (2006), 'Cryptoviral Extortion Using Microsoft's Crypto API: Can Crypto APIs Help the Enemy?', *International Journal of Information Security* **5**(2), 67–76.

[21] Abu Rajab, M. and Ballard, L. (2010), 'The Nocebo Effect on the Web: An Analysis of Fake Anti-Virus Distribution', [Online]. http://static.usenix.org/event/leet10/tech/full_papers/Rajab.pdf

[22] Spafford, E. (1989), 'The internet worm program: an analysis', *SIGCOMM Computer Communications* **19**(1), 17–57.

[23] CAIDA. 'Analysis of Code Red', [Online]. http://www.caida.org/research/security/code-red/

[24] CERT. 'Nimda Worm', [Online]. https://www.cert.org/historical/advisories/CA-2001-26.cfm

[25] CAIDA. 'The Spread of Sapphire/Slammer Worm', [Online]. http://www.caida.org/publications/papers/2003/sapphire/sapphire.html

[26] Arce, I. and Levy, E. (2003), 'An analysis of the slapper worm', *IEEE Security & Privacy* **1**(3), 82–87.

[27] Arora, A. and Telang, R. (2005), 'Economics of software vulnerability disclosure', *IEEE Security & Privacy* **3**(1), 20–25.

[28] Arora, R. (2004), 'Detecting worms through decentralized monitoring', [Online]. http://homepages.cae.wisc.edu/ raman/Projects/WormDet.pdf

[29] Staniford, S., Moore, D., Paxson, V., and Weaver, N. (2004), 'The top speed of flash Worms', in *Proceedings of 2004 ACM workshop Rapid malcode*, pp. 33–42.

[30] Weaver, N., Paxson, V., and Staniford, S. (2004), 'A worstcase worm', in *Proceedigns of the 3rd Annual Workshop on Economics and Information Security*.

[31] Gordon, S. (1997), 'What is wild?', in *Proceedings of the 20th National Information Systems Security Conference.*

[32] Weaver, N., Paxson, V., Staniford, S., and Cunningham, R. (2005), 'Large scale malicious code: A research agenda', [Online]. http://www.icir.org/vern/papers/large_scale_malicious_code.pdf

[33] Kotadia, M. (2003), 'Update Windows today - before it gets Blasted', [Online]. http://www.zdnet.com/update-windows-today-before-it-gets-blasted-3039115645/

[34] de Dr´ezigu´e, D., Fizaine, J., and Hansma, N. (2006), 'Indepth analysis of the viral threats with openoffice.org documents', *Journal Computer Virology* **2**(3), 187–210.

[35] Lekkas, D. and Spinellis, D. (2005), 'Handling and reporting security advisories: A scorecard approach', *IEEE Security & Privacy* **3**(4), 32–41.

[36] Weaver, N., Paxson, V., Staniford, S., and Cunningham, R. (2003), 'A taxonomy of computer worms', in *1st Workshop on Rapid Malcode (WORM).*

[37] Forrest, S., Somayaji, A., and Ackley, D. (1997), 'Building diverse computer systems', in *IEEE 6th Workshop on Hot Topics in Operating Systems*.

[38] Pincus J. and Baker, B. (2004), 'Beyond stack smashing: Recent advances in exploiting buffer overruns', *IEEE Security & Privacy* 2(4), 20–27.

[39] Staniford, S. (2003), 'Containment of scanning worms in enterprise networks', *Journal Computer Security*.

[40] Furnell, S. and Ward, J. (2005), 'The true computer parasite', [Online]. http://www.symantec.com/connect/articles/true-computer-parasite

[41] Zou, C., Gong, W. and Towsley, D. (2002), 'Code red worm propagation modeling and analysis', in *Proceedings of 9th ACM Conference on. Computer. and Communication Security*, Washington DC, USA.

[42] Kermack, W. O. and McKendrick, A. G. (1927), 'A contribution to the mathematical theory of epidemics', in *Proc. Royal Society of London, Series A*, vol. 115, pp. 700–721.

[43] Leveille, J. (2002), 'Epidemic spreading in technological networks', *School of Cognitive and Computer Science*, University of Sussex at Brighton, Bristol.

[44] PastorSatorras, R. and Vespignani, A. (2001), 'Epidemic spreading in scale free Networks', *Physical Review Letter* **86**(14), 3200–3203.

[45] Whitman, M. and Mattord, H. (2007), *Principles of Information Security*, Boston, Cengage Learning, pp. 290-301.

[46] Porras, P. and Neumann, P. (1997), 'EMERALD: Event monitoring enabling responses to anomalous live disturbances', in *Proceedings of National Information Systems Security Conference*.

[47] Neumann, P. and Porras, P. (1999), 'Experience with EMERALD to date', in *1st USENIX Workshop Intrusion Detection and Network Monitoring*, Santa Clara, California, pp 73-80.

[48] Vigna, G. and Kemmerer, R. A. (1999), 'Netstat: A network-based intrusion detection system', *Journal Computer Security* **7**(1).

[49] Chun, B., Lee, J. and Weatherspoon, J. (2003), 'Netbait: a distributed worm detection service'.

[50] Papadopoulos, C., Lindell, R., Mehringer, J., Hussain, A., and Govindan, R. (2003), 'Cossack: Coordinated suppression of simultaneous attacks', in *DARPA Information Survivability Conference and Exposition*, vol. 1, pp. 2-13, Washington DC.

[51] Janakiraman, R. and Zhang, M. (2003), 'Indra: a peer-to-peer approach to network intrusion detection and prevention', in *Proceedings of the 12th IEEE International Workshops on Enabling Technologies*.

[52] Cai, M., Hwang, K., Kwok, Y., Song, S., and Chen, Y. (2005), 'Collaborative internet worm Containment', *IEEE Security and Privacy* **3**(3), 25–33.

[53] Dash, D., Kveton, B., Agosta, J., Schooler, E., Chandrashekar, J., Bachrach, A., and Newman, A. (2006), 'When gossip is good: Distributed probabilistic inference for detection of slow network intrusions', in *Proceedings of National Conference on. Artificial Inteligence*, vol. 21, pp. 1115, Cambridge, London.

[54] Fung, C., Baysal, O., Zhang, J., Aib, I., and Boutaba, R. (2008), 'Trust management for host-based collaborative intrusion detection', in *19th IFIP/IEEE Int. Workshop Distributed Syst*ems.

[55] Zhong, Z., Ramaswamy, L., and Li, K. (2008), 'Alpacas: A large-scale privacy-aware collaborative anti-spam system', in *27th Conference on Computer Communication*, pp. 556–564.

[56] Polla, D., McConnell, J., Johnson, T., Marconi, J., Tobin, D., and Frincke, D. (1998), 'A framework for cooperative intrusion detection', in *Proceedings of the 21st National Information. System Security Conference*, pp 361-373.

[57] Yegneswaran, V., Barford, P., and Jha, S., 'Global intrusion detection in the domino overlay system', in *Proceedings of the Network and Distributed Systems Security Symposium.*

[58] Locasto, M., Parekh, J., Keromytis, A., and Stolfo, S. (2005), 'Towards collaborative security and P2P intrusion detection', in *Proceedings of the 6th Annual IEEE SMC*, pp. 333–339.

[59] Ghosh and Sen, S. (2004), 'Agent-based distributed intrusion alert system', in *Proceedings of the 6th International Workshop on Distributed Computing*.

[60] Li, Z., Chen, Y., and Beach, A. (2006), 'Towards scalable and robust distributed intrusion alert fusion with good load balancing', in *Proceedings of the 2006 SIGCOMM Workshop Large-Scale Attack Defense*, New York, USA, pp. 115–122.

[61] Bakos, G., and.Berk, V. (2002), 'Early detection of internet worm activity by metering icmp destination unreachable messages', in *Proceedings of the SPIE Aerosense*.

[62] Ullrich, J., 'DShield', [Online]. http://www.dshield.org/indexd.html

[63]SANS, 'Internet Storm Center (ISC)', [Online]. http://isc.sans.org/

[64] Cuppens, F., and Miege, A. (2002), 'Alert correlation in a cooperative intrusion detection framework', in *IEEE Symposium on Security and Privacy*, pp. 202–215.

[65] Microsoft, 'What is SmartScreen Filter?', [Online]. http://www.microsoft.com/security/filters/smartscreen.aspx

[66] Zhou, C. V., Leckie, C., and Karunasekera, S. (2009), 'Collaborative detection of fastflux phishing domains', *Journal Networks* **4**(1), 75–84.

[67] NIST., 'National Vulnerability Database (NVD)', [Online]. http://nvd.nist.gov/

[68] Security Focus, 'Bugtraq Vulnerabilities', [Online]. http://www.securityfocus.com/

[69] Symantec Corporation, 'Symantec DeepSight Threat Management System', [Online]. https://tms.symantec.com/Default.aspx

[70] OSVD, 'Open Source Vulnerability Database', [Online]. http://osvdb.org/

[71] MITRE Corporation, 'CVE - Common Vulnerabilities and Exposures', [Online]. http://cve.mitre.org/

[72] Oliner, A., Kulkarni, A., and Aiken, A. (2010), 'Community Epidemic Detection Using Time-Correlated Anomalies', in *Proceedings of the 13th Int. Conference on Recent Advances in Intrusion Detection*.

[73] Roesch, M., 'The SNORT Network Intrusion Detection System', [Online]. http://www.snort.org

[74] Paxson, V. (1998), 'BRO: A System for Detecting Network Intruders in Real Time', in *Proceedings of the 7th USENIX Security Symposium*.

[75] Wang, H., Guo, C., Simon, D., and Zugenmaier, A. (2004), 'Shield: Vulnerability-Driven Network Filter for Preventing Known Vulnerability Exploits', in *Proceedings of the ACM SIGCOMM Conference.*

[76] Kreibich, C. and Crowcroft, J. (2003), 'Honeycomb - Creating Intrusion Detection Signatures Using Honeypots', in *ACM Workshop Hot Topics in Networks*.

[77] Wang, K. and Stolfo, S. (2004), 'Anomalous payload-based network intrusion detection', in *Proceedings of the RAID*.

[78] Kim, K. and Karp, B. (2004), 'Autograph: Toward Automated Distributed Worm Distribution', in *Proceedings of the USENIX Security Symposium*.

[79] Newsome, J., Karp, B., and Song, D. (2005), 'Polygraph: Automatically Generating Signatures for Polymorphic Worms', *IEEE Security and Privacy*, Oakland, CA.

[80] Li, Z., Sanghi, M., Chen, Y., Kao, M., and Chavez, B. (2006), 'Hamsa: Fast Signature Generation for Zero-day Polymorphic Worms with Provable Attack Resistance', *IEEE Security and Privacy*, Oakland, CA.

[81] Singh, S., Estan, C., Varghese, G., and Savage, S. (2004), 'Automated Worm Fingerprinting', in *6th Symposium on Operating System Design and Implementation (OSDI).*

[82] Cai, M., Hwang, K. (2005), 'Fast Internet Worm Containment', *IEEE Security and Privacy*.

[83] Sandhu, R., Xinwen, Z. (2005), 'Peer-to-Peer Access Control Architecture Using Trusted Computing Technology', in *Proceedings of the SACMAT05*, Stockholm, Sweden.

[84] Whyte, D., Kranakis, E., van Oorschot, P. (2005), 'DNS based detection of scanning worms in an enterprise network', i*n Proceedings of the 12th Annual Network and Distributed Systems Security Symposium.*

[85] Feng, Y., Haixin, D., Xing, L. (2004), 'Modeling and analyzing interaction between worm and antiworm in network worm spread', *Science in China Series E* **34**(8), pp. 841–856.

[86] Zhou, L., Zhang, L., McSherry, F., Immorlica, N., Costa, M., and Chien, S. (2005), 'A first look at Peer-to-Peer Worms: Threats and Defense', in *Proceedigns of the Peer-to-Peer Systems on 4th International Workshop*, NY, USA, pp. 24–35.

[87] Jianming, F., Zhiyi, H., Binglan, C., and Jingsong, C. (2006), 'Containing Worm Based on Immune-group in Scale-free P2P', in *Proceedings of the 1st International Conference on Complex Systems and Applications*, Huhhot, China, pp. 945–949.

[88] Anderson, D. (2004), 'Boinc: A system for public resource computing and storage', in *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, USA.

[89] Anderson, D., Korpela, E., and Walton, R. (2005), 'High performance task distribution for volunteer computing', in *Proceedings of the 1st IEEE International Conference on eScience and Grid Technologies*, Melbourne.

[90] Saroiu, S., Gummadi, K., and Gribble, S. (2003), 'Measuring and analyzing the characteristics of napster and gnutella hosts', *Journal Multimedia Systems* 9(2), 170–184.

[91] Blundell, N. and Mathy, L. (2002), 'An overview of gnutella optimisation techniques', in *Proceedings of the PGNET2002 Conference*, Liverpool, UK.

[92] Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., and Shenker, S. (2003), 'Making Gnutell alike p2p systems scalable', in *Proceedings of the 2003 conference on Applications*, Karlsruhe, Germany, pp. 407–418.

[93] Wilson, B. J. (2002), *JXTA*, New Riders.

[94] Androutsellis-Theotokis, S. and Spinellis, S. (2004), 'A survey of P2P content distribution technologies', *ACM Computing Surveys* **36**(4), 335 – 371.

[95] Riasol, J.E. and Xhafa, F. (2006), 'Juxta-cat: a jxta-based platform for distributed Computing', in *Proceedings of the 4th International Symposium on Principles and Practice of Programming in Java*, pp. 72–81.

[96] Keong Lua, E., Crowcroft, J., Pias, M., Sharma, R., and Lim, S. (2005), 'A survey and comparison of peer-to-peer overlay network schemes', *IEEE Communications Surveys & Tutorials, Second Quarter* **7**(2), 72-93.

[97] Stoicay, I., Morrisz, R., Liben-Nowellz, D., Kargerz, D. R., Frans Kaashoekz, M., Dabekz, F., and Balakrishnanz, H. (2003), 'Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications'.

[98] Project Kenai, 'JXTA Java™ Standard Edition v2.5: Programmers Guide' [Online]. https://jxse.kenai.com/Tutorials/JXSE_ProgGuide_v2.5.pdf

[99] Buford, J., Yu, H., Lua, E. K. (2008), *P2P Networking and Applications*, Morgan Kaufmann Publishers Inc.

[100] Traversat, B., Abdelaziz, M., Bernard, E. P. (2003), 'Project JXTA: A Loosely-Consistent DHT Rendezvous Walker', Project JXTA, Sun Microsystems, Inc.

[101] Verstrynge, J. (2010), *Practical JXTA II*, DawningStreams Inc.

[102] Dierks, T. and Allen, C., 'etf rfc 2246: The tls protocol version 1.0', [Online]. http://www.ietf.org/rfc/rfc2246.txt

[103] CCITT Recommendation X.509 (1988), 'The Directory – Authentication Framework'.

[104] Arnedo-Moreno, J., Herrera-Joancomartí, J. (2009), 'A Security Layer for JXTA Core Protocols', in *CISIS*, pp. 463-468

[105] Bailly, D. (2002), 'Cbjx: Crypto-based jxta (an internship report)', pp. 108–109.

[106] Kaliski B. (1998), 'Pkcs#7: Cryptographic message syntax version 1.5', [Online]. http://www.ietf.org/rfc/rfc2315.txt.

[107] Montenegro G. and Castelluccia C. (2004), 'Crypto-based identifiers (cbids): Concepts and applications', *ACM Transactions on Information Systems Security* **7**(1), 97–127.

[108] Arnedo, J., Herrera, J. (2008). 'JXTA security in basic peer operations', pp. 405-414

[109] Bouncy Castle Crypto APIs, [Online]. www.bouncycastle.org

[110] The Apache DB Project, [Online]. db.apache.org/derby/releases/release-10.5.1.1

[111] Apache Felix, [Online]. felix.apache.org

[112] H2 Database Engine, [Online]. www.h2database.com

[113] Project Kenai JXTA, [Online]. http://jxta.kenai.com/

[114] JUnit, [Online]. junit.org

[115] Netty, [Online]. netty.io

[116] The Jetty Web Server, [Online]. www.eclipse.org/jetty/about.php

[117] Sqlite Database, [Online]. www.sqlite.org

[118] Pschernig E., JUG Release 1.5 *alpha, [Online]. http://sourceforge.net/projects/jug/

[119] GraphPanel, ‘Graph java code’, [Online].
https://gist.github.com/roooodcastro/6325153

[120] HideToSystemTray, ‘Hide to system tray java code’, [Online].
http://stackoverflow.com/questions/7461477/how-to-hide-a-jframe-in-system-tray-of-taskbar

[121] importCertificate, ‘JXTA code’, [Online]. https://www.java.net/node/665776

[122] Microsoft, ‘Configure the Windows Firewall Log’, [Online].
http://technet.microsoft.com/en-us/library/cc947815(v=ws.10).aspx