Metrics for Measuring Security in Peer-to-Peer Software

Ståle Botnen

Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2005

The MSc programme in Information Security is run in cooperation with the Royal Institute of Technology (KTH) in Stockholm.

Institutt for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Preface

This thesis is my final project of the Masters course in Information Security at Gjøvik University College (Høgskolen i Gjøvik).

My earlier education, a bachelor degree in Network Technology, has been of a technical character. It was therefore important for me to be able to use some of that knowledge in my MSc project, but at the same time make use of the theories and methods learned during my time at Gjøvik University College.

One of the fields that have interested me the most has been the concept of identifying and developing security metrics that can be used to measure how well applications, methods and businesses perform on security related topics. During my education I have also been fascinated with the rapid evolution of Peer-to-Peer (P2P) file-sharing technology, which has grown from being a relatively obscure technology used to swap MP3 files, to becoming one of the most talked about and used technologies on the Internet today.

It was therefore natural for me to combine these interests and make an attempt to develop metrics that can be used by consumers, developers and reviewers to objectively measure how different P2P file-sharing applications perform on some specific security related topics. To my knowledge there has been no prior attempt to combine the field of security metrics with P2P file-sharing. P2P file-sharing is one of the fastest growing technologies on the Internet today and is increasingly being adopted by large corporations who see the potential in this technology. Security is one of the elements that have to be fulfilled before P2P can become commercially viable. This thesis explores how security metrics can be used on P2P software to evaluate how different implementations of such software perform on security related topics. Several metrics for security evaluation of P2P software have been defined, as well as a ranked list based on the security rating of the assessed P2P software.

Ståle Botnen

Abstract

In today's society we see an increased use of Peer-to-Peer (hereafter called P2P) technologies in both private and commercial settings. This new technology comes in many shapes and forms, from sharing multimedia files in large unstructured networks, to information dissemination in large internal corporate networks. Much has been written about the protocols, search algorithms, authentication methods and the legal considerations that affect these networks, but there has been little focus on the security implications that the P2P applications bring with them.

This thesis will try to answer some basic questions about security in P2P and explore the use of metrics to measure security in P2P software. It will attempt to do so by studying different P2P network topologies and common attacks. This knowledge will create a base on which we can develop metrics that will help us measure security in P2P applications; these metrics will then be used to evaluate some popular P2P applications.

Sammendrag (Abstract in Norwegian)

Vi ser i dag en økt bruk av Peer-to-Peer (heretter: P2P) teknologier, denne økningen gjelder både i private og kommersielle settinger. Denne nye teknologien kommer i mange ulike former og kan strekke seg fra deling av multimedia-filer i store ustrukturerte nettverk til å være en måte å spre informasjon og data raskere i et internt bedriftsnett. Mye har blitt skrevet om de ulike områdene innen P2P, da hovedsakelig om protokoller, søkemetoder, autentisering og de rettslige gråsonene som disse nettverkene lider av. Det har vært mindre fokus på hvilke sikkerhetsimplikasjoner P2P applikasjoner bringer med seg og hvordan disse kan måles.

Dette prosjektet vil forsøke å svare på noen av de grunnleggende spørsmålene rundt sikkerhet i P2P, samt å utforske bruken av metrikker for å evaluere sikkerhet i P2P software. Dette vil oppnås gjennom en analyse av de ulike P2P nettverkstopologier og potensielle angrep på disse, for deretter å utvikle metrikker som kan måle hvor sikre ulike implementasjoner av P2P applikasjoner er. Disse metrikkene vil så bli anvendt for å evaluere noen populære P2P applikasjoner.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Due to popular services such as Kazaa **[39]** and Imesh **[51]**, Peer-To-Peer (P2P) has become one of the most talked about Internet technologies. The basic idea of cooperative computing and resource sharing has been around for a long time. The Internet as originally conceived in the late 1960s was in fact a peer-to-peer system **[2]**. But it is these new and popular P2P services that have shown the potential of P2P computing. The program that started it all was Napster **[22]** which allowed sharing of MP3 files among an arbitrary set of users. Napster used a centralized server to keep records of metadata (i.e. names of users and the files they share), but the transfer of files where performed between users only. Later services made use of a fully decentralized topology where there was no central entity and both file-searching and file-transfers were carried out between the peers only.

P2P applications are traditionally classified into three different categories **[32]**:

- Instant Messaging (IM): technologies for sending nearly instantaneous messages between users. Examples of such software are Microsoft's MSN Messenger **[52]**, Trillian **[53]** and ICQ **[54]**.
- File Sharing: technologies for sharing data between equal peers in large networks; one identifying characteristic of such networks is the lack of any central entity. Examples of such software are Kazaa **[39]**, Shareaza **[42]** and Limewire **[40]**.
- Grid Computing: technologies for sharing computer resources, most commonly CPU cycles, among many different systems. This can be used to perform processing of large amounts of data distributed over a large number of computers. An example of such software is the SETI@home project **[55]**.

Before going into security related details concerning P2P applications, we should first define what "Peer-to-Peer computing" means. A widely accepted definition of the concept does not exist, except for the general notion that the processing is spread over a large number of participating nodes with minimal, or no, central control **[23]**. Another way to describe P2P systems is to use the Peer-to-Peer Working group's definition **[33]**: *"Peer-to-Peer computing is the sharing of computer resources and services* *by direct exchange between systems. These resources and services include the exchange of information, processing cycles, cache storage and disk storage for files".*

The popularity of P2P file-sharing has made servents[1] like Kazaa, Limewire and BitComet **[56]** among the most downloaded computer software on the Internet today **[38]**. However carelessness in installing and using these applications can result in accidental sharing of sensitive information. Key privacy and security concerns related to this technology are **[24]**:

- Inadvertent sharing of sensitive personal information
- Installation of spyware or adware that communicates with a third party without the user's knowledge or consent
- Legal risks for those who, knowingly or unknowingly, violate copyright law or share illegal material (copyrighted material, child pornography, racist propaganda)

One of the major factors that need to be addressed before P2P applications can become interesting for business use is security **[31]**. Today the sharing of resources frequently takes place between peers who do not know each other personally, and who do not necessarily trust each other. To be able to use these P2P networks users frequently have to install software from third parties, thus potentially giving them access to internal resources. This has the added side effect of usually bypassing conventional security measures such as firewall software. Techniques and methods for providing authentication, authorization, availability, integrity and confidentiality are therefore among the largest challenges in relation to P2P security **[31]**.

When these goals are achieved a P2P infrastructure can be established that can act as a "P2P Service Platform" with standardized API's and middleware that can be used by any business application. By having such a P2P infrastructure, resource sharing and processing can be taken to new heights and entities can make full use of the computational resources that they have at their disposal. This has been discovered by several large international companies, and they are currently in the process of developing their own P2P technologies. **[50]** mentions some of these implementations.

[1] In true P2P systems there are no centralized servers, each client should combine the functionality of both a server and a client. Thus, Server + Client = Servent.

To achieve the goals stated earlier, this thesis will attempt to create metrics that can be used to evaluate the security of P2P file-sharing applications. The metrics should take into consideration the previously mentioned factors and should make it possible to create a ranked list based on the estimated level of security in the evaluated P2P applications.

1.2 Topics Covered by the Thesis

The main focus of the thesis will be on developing security metrics for P2P file-sharing software. As such much of the literature will be metric related. Other topics that a reader should be familiar with before reading this thesis are software development and software security since these play an important role in analysing and implementing the results produced by this thesis. Furthermore, an understanding of network topologies and network security is advantageous since these topics are discussed throughout the thesis.

1.3 Research Problem

Today's Peer-to-Peer software is insecure and can lead to information leakage and degradation of security on the hosts that run such software. Furthermore, third party programs that are included in P2P software can introduce software conflicts and increase the aforementioned risk of information leakage and security degradation. Knowing this, is it possible to create metrics that can accurately compare the security level of different P2P software implementations? And can these be made in such a fashion that consumers, software testers and developers alike can use them?

1.4 Motivation and Justification

By identifying possible weaknesses in P2P applications one should be able to gain a better understanding of how the use of these applications affects computer and network security. This would benefit both the users and the administrators of the network on which these applications run. Furthermore, it is important for entities that are about to develop their own P2P solutions for in-house operations (or modify existing solutions) to have a method or framework that they can use to evaluate the security of such software implementations. The proposed metrics can also be used as a common framework for those who evaluate P2P applications; this would make it possible for consumers to compare results from different sources and check if the results are consistent.

1.5 Research Questions

The following is a list of the research questions that this thesis discusses. The main research questions have sub questions that will need to be understood in order to answer each of the main research questions. These sub questions will be answered mostly by studying previously published literature. This knowledge will then be used as a basis for answering the main research questions.

- Is it possible to create metrics that can be used to accurately compare the security of a P2P file-sharing servent?
    - Does network topology affect servent security?
    - How to improve security in P2P software?
    - How can P2P software be exploited by malicious users?
    - What properties should these security metrics fulfil?
    - Does there exist a framework for security metrics that can be used in this thesis?

- Can these metrics be combined to create a ranked list comparing the security rating of the different P2P applications?
    - Is it possible to represent the results in such a way that users can compare the applications against each other based on their individual security needs?

- Can these metrics be defined in such a way that they require few resources in order to be evaluated, thus making it possible to quickly perform new measurements when new versions of P2P software are released?
    - What is the cost, in time and resources, to evaluate each metric?

1.6 Method

P2P technology spans a wide area (ex: instant messaging, GRID computing and file-sharing), it is therefore important to define the area that our metrics will focus on. We have chosen to focus on P2P file-sharing technology; there are several reasons for this:

- P2P file-sharing has in recent times received much media coverage
- The technology is responsible for a large portion of Internet traffic today
- Security has not been a priority when designing P2P file-sharing applications
- P2P file-sharing technology is increasingly being adopted by large commercial entities that see the potential of this technology **[67, 68, 69]**

It is our belief that P2P file-sharing applications pose a serious risk to the security of any host running such applications. For use in this thesis the concept of security will be defined as the ability to keep the confidentiality and integrity of sensitive information stored on the host computer intact. The metrics developed in this thesis will be used to assess if our belief is true or false, they will also make it possible to differentiate between the security levels of different P2P applications.

In order to achieve this goal the metrics will be designed in such a way that they are quantitative in nature. This means that all measurements that are used in the metrics shall be based on performance data from the different P2P applications. The performance that is to be measured, and the approach used to measure this performance, will be defined in the metrics themselves. The metrics will be based on NIST sp800-55 **[12]**. The guidelines and requirements put forth in this standard will be followed, this should make it possible for other researches to reproduce our findings and validate our metric design.

When the measurements have been gathered the data will have to be converted into a shared numerical value, this should make it possible to combine the performance results from the different metrics into one overall score for any given P2P application tested. The process that defines how the gathered measurements shall be converted into a shared numerical value will need to be based on a qualitative assessment based on published literature. This means that a literature study needs to be performed in order to establish a solid foundation on which we can base our assessments on. The drawback to this approach is that we can not guarantee that the conversion process will not be affected by our subjective understanding of the literature; this will need to be taken into consideration.

The work can be divided into several phases:

- Identify, gather and read published papers concerning:
    - Security in P2P applications.
    - P2P network architecture
    - Established standards for metrics design and assessment
- Develop and test metrics:
    - Formulate metrics
    - Develop experiments for use in metrics
- Perform security evaluation on P2P applications by using developed metrics
- Evaluate results and their implications
    - Theoretical value
    - Real-world value
- Present results and conclusion

These proposed phases will serve as a general structure and guide in our work.

2 Previous Work

2.1 Topology of P2P networks and their influence on security

The topology of a P2P network will have a direct impact on the overall security of the whole system. It is therefore important to understand what effect the different topologies will have on the security of the P2P software.

A P2P networks stability, dependability and security will be highly affected by the choice of network topology **[1]**. The problem of measuring these characteristics is that they are usually context sensitive. While some users may feel secure in one implementation of a system, other users might regard the system as completely insecure. So when designing a P2P system careful consideration should be given to the choice of topology, since this will affect many critical characteristics of the system as a whole.

Traditionally, security has been defined by the tree elements (Confidentiality, Integrity and Availability). The choice of topology will have a direct impact on this. A fully decentralised P2P network is likely to handle “Denial of Service” (DoS) attacks quite well, but would encounter problems when authenticating peers since there is no central server that can handle this task. A semi-centralized P2P system would help solve the authentication problem, but would be more vulnerable to DoS attacks since it introduces a potential single point of failure **[1]**. A P2P network that is fully-centralized does not exist since this would be a client-server system.

By reading **[1]** it is apparent that there exist two basic types of P2P topologies.

- Decentralized: Each node in the topology is regarded equally, and there are no control nodes. File-transfers are performed between peers in the network.
- Semi-Centralized: There exists at least one control node that performs an authoritative role in the network. File-transfers are performed between peers in the network; control nodes only perform indexing and other services.

These two basic topologies can have many variations and are therefore very adaptive to the users needs (Figure 1).

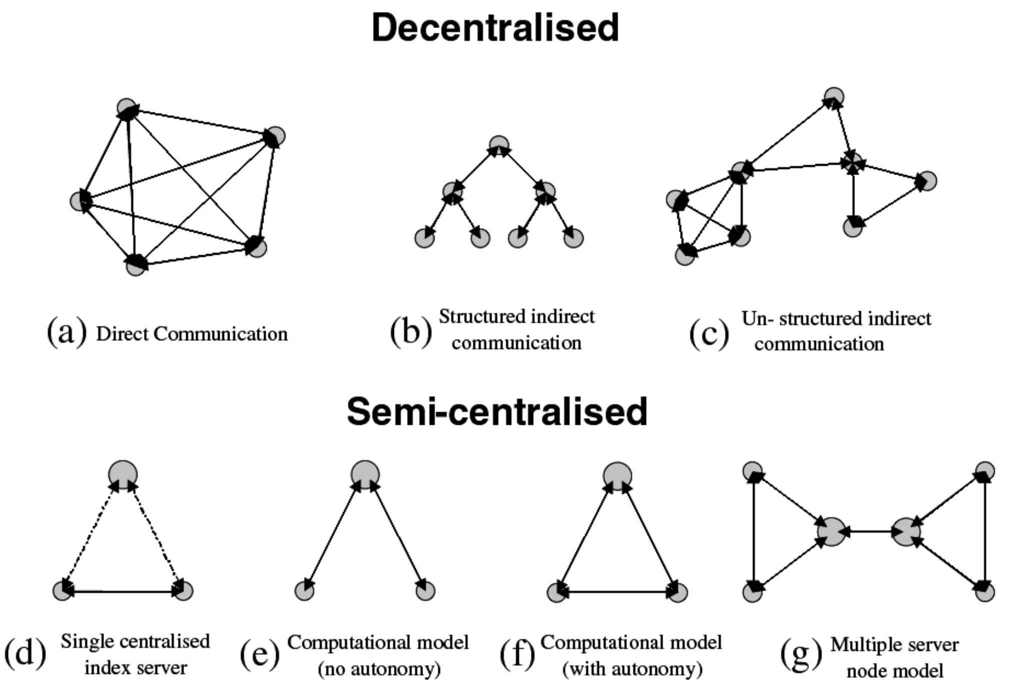

Decentralised
(a) Direct Communication
(b) Structured indirect communication
(c) Un- structured indirect communication
Semi-centralised
(d) Single centralised index server
(e) Computational model (no autonomy)
(f) Computational model (with autonomy)
(g) Multiple server node model

Figure 1 - Variations of P2P topologies

Different architectures will also have different values for the properties that affect security in any given P2P network. In **[5]** several properties that affect the dependability of a P2P network are identified. Some of these also directly affect the security of the P2P network:

- Manageability: How easy it is to manage a system. This will affect how large the system can grow before the rules and regulations no longer can be enforced.
- Information coherence: How authoritative information is. This will affect auditing and non-repudiation.
- Survivability: How well a system can perform its tasks in a timely manner in the presence of attacks, failures or accidents.
- Safety: The systems ability to operate without catastrophic failures.
- Responsibility, Accountability and Reputation: How the system enforces the rules of behavior. A strict enforcement will reduce the

risks of socially unacceptable behavior in the network and thereby increase the perceived safety of the system.

- Fault tolerance: The ability of the system to continue giving correct service following the occurrence of faults through errors in the system design, implementation, or as the consequence of an attack.
- Data integrity: How well the system maintains the integrity of the data that is stored and manipulated within the system.
- Peer discovery: How the mechanism for discovering other peers in the network works. This mechanism can often be vulnerable to flooding attacks; this should be taken into consideration at design time.
- Peer addressing: How the system assigns addresses to nodes within the system. Malicious users will try to gain addresses surrounding critical resources and use these to block other peers from these resources.
- Load balancing: How the load is distributed on peers within a system. This must be balanced to ensure that a node is not overworked or underused. A good load balancing algorithm will also make the system more robust against DoS attacks.

From the basic two topologies several variations are used today **[2]**, the following is a representation of their strengths and weaknesses as described by Nelson Minar and descriptions from **[3]**.

Centralized Topology

The concept of a centralised topology is based on the traditional concept of the client/server model. There exist a centralized server that manages a database of peers and their files. The client contacts the server and gives its current IP address and file-list; this is done every time the client is launched. The information that the server receives from the peers will then be used to create a centralized database that maps filenames to sets of IP addresses. When a client performs a search the query will be sent to the central server, which will perform a search in its database. If there is a match the server will return the direct link to the resource to the querying client. The client will then connect to the node that contains the required data; the transfer will only involve the two peers, the server is only used for searching.

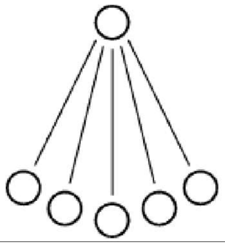
Type	Property	Value	How
Centralized	Manageable	Yes	System is all in one place
	Coherent	Yes	All information is in one place
	Extensible	No	Only one can add on to the
	Fault tolerance	No	Single point of failure
	Secure	Yes	Simply secure one host
	Scalable	?	Overload on server will be a problem in large systems

Table 1 - Centralized Topology

Ring Topology

As shown earlier the central server of a centralized topology can become a bottleneck and single point of failure. The ring topology solves this problem. It is made up of machines arranged in the form of a ring and acts as a distributed server. These machines will act together to provide a better load balancing and high availability. This topology is usually only used when all machines are relatively close to one another.

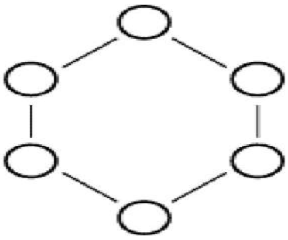
Type	Property	Value	How
Ring	Manageable	Yes	Simple rules for relationships
	Coherent	Yes	Easy logic for state
	Extensible	No	Only ring owner can add
	Fault tolerance	Yes	Fail-over to next host
	Secure	Yes	As long as ring has one owner
	Scalable	Yes	Add more hosts into the ring

Table 2 - Ring Topology

Hierarchical Topology

In a hierarchical topology, authority flows from the root server to the servers below it. Many Internet applications operate in this fashion (ex:DNS). This kind of topology is very suitable for the systems that require a form of governance and delegation of rights or authority.

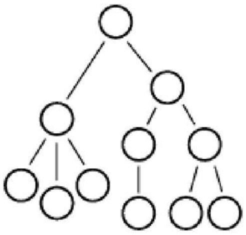
Type	Property	Value	How
Hierarchical	Manageable	Yes	Chain of authority
	Coherent	Yes	Cache consistency
	Extensible	Yes	Add more leaves, but must rebalance
	Fault tolerance	No	Root is vulnerable
	Secure	No	To easy to spoof links
	Scalable	Yes	Hugely scalable. Ex: Domain Name Servers (DNS)

Table 3 - Hierarchical Topology

Decentralized Topology

This topology is used in pure P2P networks where all peers are equal; this creates a flat, unstructured network topology. Since there is no central server in the network a node that wishes to connect to the network must first contact a bootstrapping node (a node that is always online) and receive the IP addresses of one or more peers. Each peer will only have information about its neighbors. Since there are no servers to manage searches, queries for files are flooded through the network, query flooding has proven to be a problem since it entails a large overhead of traffic in the network.

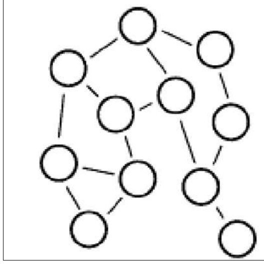
Type	Property	Value	How
Decentralized	Manageable	No	Very difficult, many owners
	Coherent	No	Difficult, unreliable peers
	Extensible	Yes	Anyone can join
	Fault tolerance	Yes	Redundancy
	Secure	No	Difficult, open research
	Scalable	?	Theory: yes, practice: no. As the size grows performance deteriorates

Table 4 - Decentralized Topology

Centralized + Ring Topology

This topology is very common in web hosting where heavy loaded web servers usually have a ring of servers that specializes in load balancing and failover **[3]**. This means that the servers work in a ring topology, while the clients connect to the servers in a client/server relationship. This creates a very robust topology, while still remaining manageable.

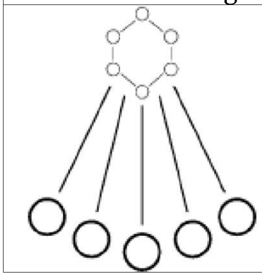
Type	Property	Value	How
Centralized + Ring	Manageable	Yes	Just manage the ring
	Coherent	Yes	As coherent as ring
	Extensible	No	No more than ring
	Fault tolerance	Yes	Works as a large ring
	Secure	Yes	As secure as ring
	Scalable	Yes	As long as the main ring can handle the increased traffic

Table 5 - Centralized + Ring Topology

Centralized + Decentralized

In this topology there exist peers that function as super nodes. These super nodes perform the tasks that would have been performed by a centralized server in the centralized topology, but only for a subset of peers. The super nodes themselves are connected to each other in a decentralized network. The topology introduces two tiers of control: a centralized client/server relationship between the peers and a super node, and a decentralized network among the different super nodes.

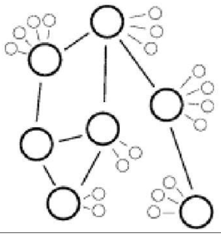
Type	Property	Value	How
Centralized + Decentralized	Manageable	No	Same as decentralized
	Coherent	Yes	An improvement on
	Extensible	Yes	Anyone can still join
	Fault tolerance	Yes	Plenty of redundancy
	Secure	No	Same as decentralized
	Scalable	Yes	Shows great potential in scalability

Table 6 - Centralized + Decentralized Topology

As we can see from the above section there are a large number of different topologies used in P2P networks. All of these topologies have different weaknesses and strengths. The choice of network topology is something that will have to be strongly influenced by what context the P2P system is going to be used in. Some P2P systems will require ways to authenticate users and keep track of the data in the network, while other systems will require a high degree of redundancy and resistance to attacks. As of today there is no single topology that can fulfill all needs. Developers will therefore need to consider carefully their choice of network topology since it will be a difficult task to change this later during the development of the system. All these topologies are vulnerable to different forms of attack ranging from simple DoS attacks on nodes to more subtle integrity and forgery attacks against the resources in the network.

P2P network topologies used today

As shown by the sections above there exists many different P2P topologies. However, the P2P file-sharing software used today mostly relies on a subset of these network topologies. The most common implementations are the centralized + decentralized topology and variations of the centralized topology.

Centralized + Decentralized Topology:

- Kazaa
- Bearshare **[57]**
- Grokster **[41]**
- Shareaza
- Limewire
- Edonkey**[58]**

Centralized Topology:

- Napster **[22]**
- BitTorrent **[60]**
- Direct Connect **[59]**

The centralized + decentralized topology has many advantages over the other topologies, and is today the most common implementation in P2P networks. One of the reasons for this is that this topology has proven that it is scalable as well as searchable even when there are large numbers of peers connected. Earlier, decentralized implementations produced an enormous overhead when searching through the network. The peers connected to such a decentralized network did not necessarily have the required amount of bandwidth; this often resulted in a fragmented network. In the centralized + decentralized network, query request are handled by the supernodes. These supernodes are more likely to have the required bandwidth and a breakdown of the network is therefore less likely **[3]**. The networks that protocols such as BitTorrent and Direct-Connect operate on are not strictly centralized; they are a "hybrid" version. In this case that means that the index[2] is accessed in client-server mode, whereas the files are transferred directly between peers.

The P2P software that will be evaluated in this thesis will be Kazaa, Limewire, Grokster and Shareaza. Kazaa and Limewire have been studied in **[35]** with focus on bundled third party software. We have therefore chosen Kazaa and Limewire in order to compare our findings with those in **[35]**. Grokster and Shareaza have been chosen based on their frequent occurrence in discussions on several P2P forums and their high number of downloaded copies.

The chosen P2P applications all have approximately the same network topology. Based on our findings earlier in the thesis it is our opinion that it is not necessary to develop metrics for measuring how the network topology affects the security of the individual P2P applications. This is based on the fact that all the chosen P2P applications operate on a centralized + decentralized network topology. There are differences in the architecture of these networks but these are mostly variations on the same theme. One example of this is the fact that the various networks operate with some differences in the hierarchy of supernodes **[3]**. To go into details on these variations and their effects on P2P software security is outside the scope of this thesis.

[2] A list over online peers and the files they are sharing

2.2 Factors that affect security in P2P networks

There are many factors affecting the security of any given P2P system. This section will focus mainly on the P2P software. Several articles have been published that discuss the dangers of installing P2P software on computers, but in many cases the advantages of P2P functionality far outweighs the associated risks. The following is a study of published literature that discusses some of the inherent risks related to the use of P2P software and proposes solutions to some of these risks.

Open P2P networks are often insecure since users can join without any authentication of their identity or proof that the data they are sharing is not malicious software. It is a known fact that P2P networks are used by malicious users to spread viruses, trojans and other malicious programs.

[4] proposes a way to drastically improve the resilience of P2P networks by introducing a system called "NetBiotic". In this system several computers in the network will disseminate information about probable security attacks to each other; this will ensure a rapid spread of information regarding new attacks between the cooperating nodes. Each node will be responsible for:

1. Detecting whether a virus or worm is propagating through the network and possibly causing an epidemic.
2. To automatically send out warnings and information to other peers connected to the network.
3. Take precautions for protecting its host. This can be done by a stricter security policy during the time span of the suspected epidemic.

The hope is that by gathering this information the nodes will be able to estimate when a new wave of attacks are about to happen, and take appropriate countermeasures without the intervention of the user.

This method can provide protection against the spread of viruses and trojans, but will not be able to protect an application against attacks that rely on the actions of the user. It would therefore be important to find ways to protect the user from performing actions that would result in an increased chance of exposure to attack.

One such method is to make a trust based system available in P2P networks. This goes for both P2P applications by themselves and the data shared on P2P networks. Today there are few ways to confirm the integrity and authenticity of P2P programs; these are programs that usually require full access and privileges on the host computer to operate in a satisfactory way. Since it is nearly impossible to control that the P2P software itself is secure, it is necessary to have architecture to safely run un-trusted code on. Several such architectures have been proposed in **[24]**.

When it comes to protecting the host computer from malicious nodes, there are some methods that can be implemented. When users share their data with others, there is a chance that they accidentally share more data than they know. Windows XP users can reduce the chance of malicious users gaining access to sensitive data by using the built in file-sharing features. They can then designate data as either shared or private. Private data can only be accessed by the machine's owner. User should not depend on the built in protection of the P2P software as it can easily be bypassed by an experienced hacker **[64]**.

Backdoor attacks are also a common form of attack, not only on P2P networks, but throughout the Internet. As much as 45% of files downloaded from P2P networks have been shown to contain some form of malicious code **[65]**. Malicious users can disguise viruses and trojans in well known file formats; this is done with software commonly known as "Wrappers". The most efficient way to defend against such attacks is by having up to date antivirus software. This software will analyse any suspicious files and alert users when it detects malicious code **[64]**. However, this does not provide 100% protection since antivirus software only detect viruses and trojans that they recognize. This means that unknown variations of such malicious code will go undetected.

2.3 Possible attacks when using P2P

As with most software implementations today P2P software is insecure. It is widely known that the installation of such software will create new ways for malicious users to cause damage **[4**]. While some of these weaknesses are relatively unknown by the users and developers, others are known and could have been easily avoided had the developers considered the problem during development **[6, 7]**.

An example of P2P software that has been criticized for its many weaknesses, but which was one of the most popular P2P applications of its day, is Gnutella. Gnutella has since been surpassed in popularity by programs such as Kazaa, Bitcomet and Direct Connect, but its architecture has been adopted by many new P2P implementations. Most of the basic weaknesses that plagued the Gnutella network can therefore be expected to exist in these new implementations.

Several papers have been published discussing the weaknesses of Gnutella and other P2P implementations **[8, 9]** list several of the most serious problems with the Gnutella implementation.

Gnutella has no login, no authentication and no central authority of any type; it is therefore a completely decentralized architecture. This brings with it the problem that no user can truly know who he is sharing or receiving data with, this anonymity makes the Gnutella community the perfect breeding ground for malicious software.

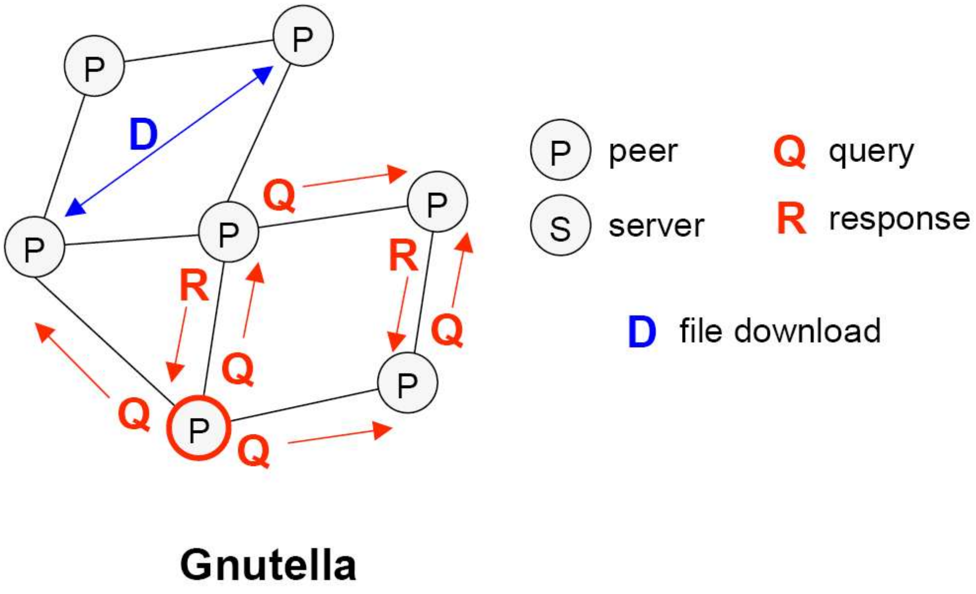


Figure 2 - Gnutella architecture

As Figure 2 shows, queries spread throughout the network and every peer with the requested file will send a response to the originator telling him about the location of this file. If the requested file is not available on a node, it will send the query to its neighbors. This will be repeated until the TTL (Time To Live) property in the query runs out. This method of searching can be exploited by malicious users to generate flooding attack by using unchecked IP addresses and port numbers.

It has also been found that the Gnutella servent has been used by malicious users to probe systems to discover operating system version and other information that can help an attacker in the information gathering phase. On Windows 95, 98 and NT, Gnutella's GUID[3] has been shown to contain the hardware MAC address (which should remain constant over time), making it possible for an attacker to track request over time and thereby gathering information that can be used in an attack against the targeted user.

In addition to the above problems Gnutella also contains the PUSH message in its protocol. This message was implemented to allow downloads from firewalled hosts. The querying node sends this message to the firewalled node, the receiving node then starts a TCP connection back to the querying node with a string indicating the file in question. When this standard outgoing TCP connection is established (allowed by most

[3] Globally Unique ID

firewalls) the querying node can send a HTTP request and receive the file. This effectively bypasses any security that the firewall can provide **[34]**.

Information leakage is a serious concern when it comes to the use of P2P software and Gnutella is no exception. It provides malicious users with an easy way to gather information about many users. Several problems exist with Gnutella that could have been solved in the development phase:

- It announces IP addresses. This represents a serious problem, especially for those networks which do not safeguard their users with hiding processes such as Network Address Translation (NAT) or various other types of proxies. This exposure can have two consequences. The first is the possibility of users being monitored by third parties. The second is that attackers could, once they recognize the IP address used for the connection, use it to perform security probing or more severe attacks.
- It announces full path names, making it possible for attackers to get a complete picture of the system on which the software is running.
- It announces Gnutella topology, which may reflect real-world patterns of association. The worst case scenario would be that attackers get a complete picture of the number and placement of clients on an internal corporate network.
- It can use any port number which makes it very hard to detect and to control outbound connections via the firewall. Gnutella even has a special “Push” command that asks the receiver to establish an outbound connection to the sender of the “PUSH” command, thereby possibly bypassing the firewall.
- An eavesdropper can easily record queries and responses, making it possible to create content that will attract special groups of users (e.g. those who search for a specific type of content) and target these users for attacks.
- The combination of “Query/Push” makes it possible for an attacker to forge the return address, and thereby induce other nodes to try to send a large file to some arbitrary destination. This method has been used to create DoS attacks similar to “FTP Bounce” attacks.

- There is no guarantee that what a user receives is what he wanted. A node can return false content (virus or trojans) or users can receive obscene and possibly illegal content in response to innocent queries.
- Nodes can falsely advertise a high-speed connection to attract more clients, and thereby spreading malicious software quicker.

Another paper **[8]** that discusses Gnutella states that there are many side effects of using P2P software that are not easily apparent to users of these programs. One of these "side effects" is spyware programs[4]. Virus files are also a severe problem but have lately been estimated to be less severe than virus that spread over e-mail **[11]**. Poorly written P2P clients pose another problem in these networks since they expose the users to all the exploits and weaknesses of these implementations. The P2P software that has been developed after Gnutella is also frequently shown to contain serious flaws and weaknesses. One example of this is a weakness that was discovered in May 2003, in Kazaa version 2.02, that made it possible for malicious users to crash or run code on any supernode on the network[5].

[24] introduces the concept of structured Peer-to-Peer overlays such as CAN **[25]**, CHORD **[26]**, PASTRY **[27]** and TAPESTRY **[28]**. These overlays provide a solid base for large scale P2P applications by providing a powerful construction ground for a multitude of decentralized services. These services include network storage, content distribution, web caching, searching and indexing, and application-level multicast. Furthermore, structured overlays allow applications to locate any object or resource in a probabilistically bounded, small number of network hops. These systems are scalable, fault tolerant and provide effective load balancing. Wallach claims that making these systems secure will be a significant challenge since any system that is not designed to withstand an adversary can easily be broken by one. P2P overlay systems are no exception.

[24] Discusses several of the attack possibilities, both those aimed at unstructured P2P systems and those aimed at structured P2P systems.

[4] A *spyware program is a program that is often distributed together with the P2P program. Such programs can send out personal information to third parties* **[10]**

[5] Http://www.securitytracker.com/alerts/2003/May/1006846.html

Attacks on unstructured P2P systems:

Attacks by self replication

Most P2P systems today assign user IDs independently from their IP address. This makes it possible for malicious users to operate without concern since they can easily acquire a new identity whenever they need to. A malicious user can answer positively to all queries, thus indicating that he possesses the requested resource, but return content that he has manipulated. If he gets discovered he can easily change to another ID and continue disrupting the network. Furthermore, honest peers, who are unaware of the altered content, will continue to share it and thus contributing to the diffusion. An example of this is a Gnutella worm called "Mandragore" which registers itself as an active peer within the network and responds positively to all requests. As an answer to any request, it sends a renamed copy of itself, thus replicating itself throughout the network.

Man in the middle attack

This type of attack takes advantage of the application level routing in the P2P network. By placing itself between two peers a malicious user can intercept traffic between them. By altering the IP address and port number in a "QueryHit" message (contains confirmation on the requested resources) a malicious node can deceive the querying peer and make it connect and download altered content from the malicious node.

Denial of service attack

This is the most common form of attack on an unstructured P2P network. These attacks take advantage of the querying structure implemented in most P2P networks. Querying is performed by sending out queries to all connected peers; these queries will then propagate throughout the network. By using this mechanism, a malicious user can create flooding attacks by continuously generating new queries with a high TTL (Time To Live) on the network. These queries will generate a large amount of network traffic, possibly rendering the network unusable by honest peers.

Attacks on structured P2P systems:

Routing attacks:

Routing attacks are aimed at exploiting weaknesses in the routing protocol used by the different P2P overlays. There are several variants of routing attacks:

- Incorrect lookup routing: A malicious node can route lookup requests to non-existent nodes. If this can be achieved in a large enough scale then the network performance will degrade.
- Incorrect routing update: Each node in a lookup system builds its routing table by requesting routing information from other peers. This makes it possible for a malicious peer to corrupt the routing table of other (innocent) peers by supplying them with incorrect updates. A subtler approach would be to supply peers with routing information leading to unreliable, high latency peers, or to other malicious peers.
- Partition attacks: These attacks attempt to form a parallel network running the same protocol as the legitimate network. By using the bootstrap method[6] malicious users can deceive innocent peers into connecting to this illegitimate network.

Storage and retrieval attacks

A malicious node can join the network and participate in the lookup protocol correctly, but when other peers wish to download from this malicious node it would deny them access to the data or deny the existence of such data.

DoS attacks

Denial of service attacks work just as well in structured P2P networks as in unstructured P2P networks. A malicious node can generate garbage packets and thus overloading a targeted node. This can cause the targeted node to fail, and remove this node's resources from the network.

Node joins and leaves

A malicious node could degrade the performance of the network by constantly joining and leaving the network. Events such as a join require that the network update its routing tables and rebalance the distribution of shared data by moving data to the

[6] E*very connecting peer needs to have the address of at least one node from which it can receive routing data from; this node is called the bootstrap node.*

newly joined node(s). If nodes join and leave at a high rate this will create a large overhead of traffic and processing, thus degrading the performance of the network.

It can be shown that structured P2P overlays can be effective when it comes to information retrieval, load balancing and distribution of resources. Overlays can remedy some of the weaknesses that exist in unstructured P2P networks, but they are far from being secure systems **[24]**.

3 Proposed Metrics

The Internet has made it possible for attacks to spread at a much faster rate than before and cause more damage than would be possible if they would have limited to the physical realm. The defence against this is not to try to create perfectly secure systems and software, since that is impossible **[19]**. Instead, one should concentrate on creating speed bumps to slow down the attacks and keep the negative consequences to a minimum **[19]**. By having metrics to measure the security of applications running on the system, we can remove software that achieve a low security rating and use software that achieved a high rating instead, thus potentially increasing the overall security of the system.

According to **[20]** metrics can be separated into 3 different categories:

- Technical: Metrics that measure/compare technical objects, e.g., vulnerabilities detectable by scanner, known bugs. These are used to differentiate between different technical alternatives. They can also be used to measure and document other factors, like software interoperability.
- Organizational: Metrics that are best applied on processes within the organization and program implementations.
- Operational: Metrics used to measure properties of systems that are in operation, operating practices and measures relating to specific environments.

The metrics proposed in this thesis are of technical character; they attempt to measure degrees of security in different P2P applications. The hope is that, based on the results, we can create a list over the different P2P servents and their level of security in relation to each other.

Furthermore, **[20]** proposes several properties that should be fulfilled by a "good" IA (Information Assurance) metric. These include:

- Scope: The portion of the information security problem domain that the metric describes should be clearly characterized.
- Sound foundation: The metric should be based on well defined methods and documentations on the subject of which it is concerned.
- Process: The metric assessment process should be well defined. This means that the process description should contain information regarding:

  - Identification of required information
  - Instruction on how specific factors are to be measured or assessed
  - Algorithms for combining factor values into final values
  - Explanation of sources of uncertainty
- Repeatable: A second evaluation by the same evaluators should produce the same results.
- Reproducible: A second assessment by a different set of evaluators produces the same result.
- Relevance: Metrics should be useful for decision-makers.
- Effectiveness: It should be possible to evaluate the metric quickly enough, and with low enough costs, for it to be useful for the decision-makers who will use it.

These properties reflect the importance of reliability and validity. Without reliability we cannot fulfill the requirements for repeatable and reproducible measurements. And without validity we cannot be sure that the measurements are of relevance since we may in fact be measuring something different from what we intended. The properties presented above will function as guidelines in our work to develop metrics that can measure the security level in different P2P applications; we have therefore chosen to make use of the template that is specified in **[12]** when developing the metrics, (see "Appendix E – Sample Metric").

As indicated by the metric properties above, metrics are tools used to facilitate decision making by providing a structured process for collecting, analyzing and reporting performance data. The basic premise is to develop a standardized method for evaluating performance; this can be used to compare products or designs from different vendors. The metrics should be repeatable and reproducible **[12]**.

Since the introduction of P2P networks have created yet another way for attacks to spread, it is important to have the necessary tools to measure how this affects the security of the system. By creating metrics to measure the security of different P2P servents we can help improve the overall security of systems running these P2P applications. By creating metrics that measure the security of the individual P2P servents, we shall make it possible for consumers and administrators to evaluate different solutions against one another and choose the applications that will fulfill their security needs. Some may choose to use P2P applications that do not contain any spyware/adware programs; others may focus on program size and complexity.

NIST sp800-55 **[12]** defines metric design as the following;

> *"IT security metrics must yield quantifiable information for comparison purposes, apply formulas for analysis, and track changes using the same points of reference. Percentages or averages are most common, and absolute numbers are sometimes useful, depending on the activity that is being measured."*

Of note here is the focus on a numerical value; this value will be used to compare the measurements in a structured manner. Furthermore, the measurements must provide relevant and correct data that can be used to evaluate performance trends over time.

[12] also specifies that the number of metrics developed should be between 5-10 per stakeholder and that weighting scales can be used to differentiate between the importance of selected metrics, thus ensuring that the results accurately reflect security priorities.

The following form is based on the template defined in **[12]**, see also "Appendix E – Sample Metric" for an example of a NIST developed metric. In this paper we have done some modifications to the template.

- "Critical element" and "Subordinate question" have been replaced by "Metric ID" and "Name"; since these metric will be used in a very limited area of operation we only need to differentiate between them (the defined area of operations is security in P2P software implementations).
- Rows for "Validity" and "Reliability" have been added. As defined in **[13]** validity and reliability are important properties when it comes to the use of metrics. Validity and reliability estimates how well the results corresponds to the true value. We have therefore chosen to have an assessment of the metrics values for these two properties.

[13] defines reliability as being how reproducible measurements are. This means that good reliability equals a low number of random errors. Validity is defined as the absence of systematic errors in the measurement.

Metric ID	A unique number that identifies the metric
Name	Name of the metric
Description	Description of the metric
Metric	Description of what we are trying to measure with this metric
Formula	Describes the formula that will convert the metric into a numerical form
Purpose	What is our goal with measuring this metric
Implementation	How the measurements are gathered
Frequency	How often does one need to retest this metric
Cost	The time and resources needed to complete one measurement of this metric
Indicators	Information about what the metric values indicate in terms of security
Validity	An evaluation of the possibility that we are not actually measuring what we defined as the purpose of the metric
Reliability	An evaluation of the chance of random errors with this metric

Table 7 - Metric Template

The gathered metrics will be of both quantitative and qualitative values; these will be run through formulas or fixed scales to produce a numerical value that can be used to compare the different software implementations. The ideal metrics would be where we achieved a purely quantifiable value from the measurement alone; this may be achievable in some of the metrics. The metrics will be designed in such a way that all results will be in the same format, in this case a number between 0 and 100, where 0 is the worst score and 100 is the best score

3.1 Number of Lines of Code

It is a well known fact that programmers make errors when coding software. According to **[14]** the rate at which errors are introduced is estimated to be 1.2 errors for every 200 lines of code. However, during a 4 year long test of Linux, researchers at Coverity concluded that Linux has approximately 985 bugs in 5.7 million lines of code (2.6 Linux production kernel). According to Carnegie Mellon University's CyLab Sustainable Computing Consortium commercial software contains 20-30 bugs for every thousand lines of code **[31]**. As these two examples show, there are no sure numbers when it comes to error rate in programming.

The research from Coverity **[31]** has been gathered during 2000-2004 and is one of the more comprehensive studies of error injection. Unfortunately no papers based on these measurements have been published yet. Since Coverity's study is one of the few studies that have been conducted over such a long time span, this is also a fairly recent study. Because of these factors we choose to use Coverity's numbers for error injection. Their numbers will be used in developing our metric for measuring the theoretical number of errors in a P2P servent. Since there has been little focus on security during the development of P2P application, we also choose to use the high-end error injection number (30 errors per thousand lines) to reflect the lack of focus on security in P2P applications. **[14]** also claim that the increasing complexity and size of software today increases the chance that these errors will have a serious impact on the security of the computers running the software. **[45]** states that when an interface tries to accomplish to much its implementation will probably be large, slow and complicated. This is what we see examples of in today's P2P applications. More and more functionality is incorporated into applications that have little or no need of such functionality. This results in applications that are larger and more complex than they need to be.

As stated earlier, in recent years the size and complexity of the P2P clients have increased **[21]**. This is mainly due to the higher demand by users for easier to use applications and added functionality. The developers of these applications seem to have had little focus on security, as evident by the lack of security related topics in their online documentation **[39, 40, 41, 42]**. Since P2P applications in the past have had little or no commercial value, few steps were taken to ensure that the clients would be secure. Today we see P2P applications being used in many commercial settings (ex:

distribution of software and MP3 **[61]**), but security still seems to be a low priority for users and developers.

Applications that have a relatively small number of coded lines will be easier to maintain, error check and audit. This will reduce the risk that serious programming errors will remain undetected. It will also be easier for the developers to make changes in the software when an error is discovered, since they will not be restrained by the complexity of the programming structure **[15]**.

Our metric will use the error rate specified in **[31]** to deduct the theoretical number of possible errors in the code. This will give us a way to measure the probability that an exploit can be crafted and used to penetrate the software. This will also reflect the fact that larger and more complex programs contain more errors than small and simple programs.

Metric ID	M1
Name	Lines of code.
Description	Measures the lines of code written in the specified P2P software.
Metric	There is a correlation between lines of code and number of errors in the software
Formula	Potential errors = $\frac{number\_of\_coded\_lines}{1000} \times 30$
Purpose	To get a numerical representation of the theoretical number of errors in the software.

Implementation

Open-Source software

If the software is open-source then a simple count of the number of code lines can be performed.

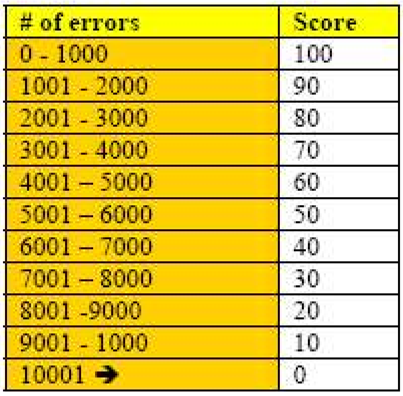
Proprietary software

If the software is proprietary then we will need to get the necessary information directly from the developers.

If this is not possible then two other possibilities exist:

- Reverse engineering the software to get approximate number of code lines
- Compare with open-source software of the same complexity and size to get approximate number of code lines

From the formula defined earlier we get the number of potential errors in the given P2P software. This number is then used in the table below to identify the security score.



# of errors	Score
0 - 1000	100
1001 - 2000	90
2001 - 3000	80
3001 - 4000	70
4001 – 5000	60
5001 – 6000	50
6001 – 7000	40
7001 – 8000	30
8001 -9000	20
9001 - 1000	10
10001 ➔	0

Frequency

This metric will need to be updated after any major software upgrade of the P2P servent.

Cost

Very low. No special equipment or resources are required.

Indicators

A high number of theoretical errors will indicate that there is a high possibility that there exist buffer overflows or other weaknesses that can be used in an exploit.

Validity	It is unrealistic to believe that all programming errors in software can be used to craft exploits; this may affect the validity of this measurement since we want to measure the errors that can be used in exploits.
Reliability	This metric should be reliable as long as the same formula is used to calculate number of theoretical errors. It is important that the number of code lines is accurate or at least that the discrepancy between actual number and perceived number does not change. It will also be important that the length of the code lines is similar; if there are large variations in length this will affect the reliability negatively.

Table 8 - Number of errors

3.2 Number of Spyware and Adware Included

It is well known that the introduction of spyware and adware increases the risk of information leakage and software conflicts. It can also detract from the usability and stability of the users' computers **[16, 44]**. In 2001 is was revealed that several P2P file-sharing programs came with a hidden program called "ClickTillUWin" (this program was included in Kazaa, BearShare and Limewire). This program logged every site that a user visited and transmitted this information to a third party. And in 2002 it was revealed that Kazaa had bundled a "sleeper" spyware program (called Altnet SecureInstall) with its servent software. This software was activated after a set amount of time and connected its host computer into a hidden P2P network. This network was then used to host and distribute content from third parties. Only by thoroughly reading the EULA could users identify that they were installing this software **[64]**.

It will be of interest to measure the amount of these programs with the intention of using them in a measurement on the security of P2P servents. The problem of unwanted software bundled with P2P file-sharing applications has been noted earlier in an FTC press release **[36]**. The problem has even been discussed internally in Sharman Networks, developers of Kazaa, as shown by a well known internal document **[37]** from Sharman Networks.

The subject of spyware/adware in P2P applications has been discussed by Ben Edelman in **[35].** His findings show that spyware and adware are bundled with popular P2P servents. However, he finds that such third party programs are usually documented in the P2P applications end user license agreement (EULA). We have deliberately selected two of the same P2P file-sharing applications tested in **[35]** (Limewire and Kazaa) to see if our results will match those found in that paper. If our results match his, we shall have further proof that our metric is usable. In addition to Kazaa and Limewire two other applications will also be tested, namely Grokster and Shareaza.

There are many different types of spyware. We can define the following types:

- Cookies and web bugs: Passive form of spyware that rely on web browser functionality

- Browser hijackers: Hijackers attempt to change web browser functionality to modify start page, search functionality and other browser functionality.
    - Brower Helper Object (BHO): In its natural state a BHO is not malicious. A modified BHO can infiltrate browsers (usually Microsoft Internet Explorer), and perform any action on the available windows and modules. This includes detection of events, creation of windows, monitoring of messages and actions. These are not stopped by firewalls because they are seen by the firewall as normal browser traffic.
- Keyloggers: Log the input received from the user, can also log web page access, IM messages, windows opened and programs executed. This is then sent to a third party.
- Tracks: A track is a term used for information recorded by the operating system about actions that the user has performed. These tracks can be mined by malicious users and programs.
    - Data Miner: Mines the system for personal information and sends this to a third party. Common information that is mined is surfing habits, visited web-pages and personal information.
- Malware: Refers to a variety of malicious software, including viruses, worms, trojan horses and automatic phone dialers.
- Adware: Displays advertisements tuned to the user's current activity, potentially sending information to a third party. Can cause conflicts with other software.
    - P2P Adware: subcategory of adware; uses P2P technology to distribute and download advertisements and other information. No central download server needed.
- Downloader: Program designed to retrieve and install additional files when run. Most will be configured to retrieve files from a designated web or FTP site. Often used in conjunction with adware or spyware to repair or update installations of said programs.

All the discovered spyware and adware programs will not be considered equal when it comes to the amount of security risks they introduce. We are aware that some types of spyware/adware represent a larger security risk than other types. This is a factor that will be considered by assigning the different types a numerical value that corresponds to the perceived threat that they represent **[10].** This is also in line with the recommendations specified in **[12]** concerning the weighting of metrics or elements within metrics.

Tests have shown **[46]** that the available spyware and adware detection software does not have a 100% detection rate. This will need to be taken into consideration in any metric that makes use of spyware detection software. One way to compensate for this is to make use of two or more spyware and adware detection software solutions. This has also been incorporated in our metric.

Metric ID	M2
Name	Number of spyware and adware
Description	Measures the number of spyware and adware components that are introduced when installing a P2P software servent
Metric	The introduction of spyware and adware represents a security threat by introducing untested code and third party software.
Formula	Score= $100-\sum_{8}^{1}(numbers\_of\_i \times score\_of\_i)$
Purpose	To get a numerical representation of the number of third party software that are hidden in the P2P servent software

Implementation

Spyware and Adware

Detection software:

“Spybot – Search and destroy” v1.3 or higher

“XoftSpy” v4.1 or higher

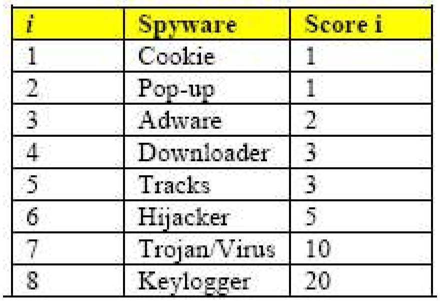
(version must be consistent in all the measurements)

Before the installation of the P2P software it is necessary to run the detection program to discover if there are any spyware or adware programs already present on the computer. If there are any spyware or adware then these must be removed.

The installation of a P2P servent can commence when all spyware and adware programs have been removed

When the installation is complete the detection program must be executed and the number of spyware and adware documented.

The following table contains the ID and the score of each of the mentioned types of spyware and adware. This information is used to calculate the total score by inserting the appropriate values into the formula.



i	Spyware	Score i
1	Cookie	1
2	Pop-up	1
3	Adware	2
4	Downloader	3
5	Tracks	3
6	Hijacker	5
7	Trojan/Virus	10
8	Keylogger	20

Frequency

This metric will need to be updated after any major software upgrade of the P2P servent

Cost	Low. Does require some expenditure of time when installing the application, detecting spyware/adware and returning the system to its default state. No expenditure of money.
Indicators	A high number of spyware and adware will increase the risk of security weaknesses in the P2P servent. This is because the third party software can create unpredictable conflicts with existing software, lead to information leakage and introduce unnecessary lines of code
Validity	We cannot be sure that the detection software discovers all spyware/adware programs; this can represent a problem with the validity of the measurement. The issue may be reduced by using several spyware removal tools. There is a risk that the installation of the detection software by itself will introduce spyware or adware programs.
Reliability	A problem with reliability arises if we do not always use the same kind of detection program. We cannot then be sure that we get comparable results because the different detection software uses different detection algorithms and signatures

Table 9 - Spyware and Adware

3.3 Reveal and Download Rate

Information leakage is one of the major threats facing P2P users today **[17]**, and this is a threat that will continue to grow as more and more businesses incorporate P2P file-sharing as a business aide. Information leakage through P2P file-sharing can occur in several different ways:

- An insider knowingly shares confidential data: Insiders are one of the most serious threats when it comes to information leakage; the introduction of P2P file-sharing has given insiders a new tool in their arsenal. As always, a dedicated insider is notoriously hard to stop.
- An attacker gains access to the system through a software error in the P2P application: P2P software comes with a whole host of security weaknesses; these can be exploited by an attacker to gain access to an otherwise well secured system.
- A user unknowingly shares confidential information: Information that a user unknowingly shares represents a serious danger when it comes to information leakage. It is suspected that malicious users regularly search P2P networks for files that users have shared by mistake.

The powerful search capabilities built into the P2P software makes it easy for malicious peers to search through the millions of users that are online and find data that can be used in several malicious ways:

- Commit fraud
- Invade privacy
- Commit identity theft
- Blackmail
- Damage a persons social standing

These consequences show that the accidental sharing of sensitive data can be a serious security risk. However, reports from the General Accounting Office and the Federal Trade Commission indicate that Internet sources of information constitute a very small percentage of the amount of identity theft cases, but underreporting is also a contributing factor and should not be underestimated. People either don't know that they have been the victims of identity theft, or they are afraid of reporting it since they may have engaged in illegal downloading **[23, 48].** But it is reasonable to assume that the rapid growth of P2P will result in an increase of identity thefts through the use of P2P software.

This metric will measure how fast files of a sensitive nature are revealed and downloaded in the different P2P networks. As such, this metric will mostly concern information leakage through files that are shared by mistake. When conducting this measurement it is important to share files that will appear authentic to a malicious user, otherwise we shall not get truly valid measurements.

Another concern is the amount of time these files should be made available for download on the network. We need to have a fixed upper time limit for the experiment to work. This time limit must closely resemble the amount of time that sensitive information would be available online in a real-world situation. **[9]** has measured the average session time on Napster and Gnutella. For both applications the average session duration was approximately 60 minutes. According to the paper this was not surprising; as it corresponded to the time it typically took for users to download a small number of music files from the service. We shall be using this average time as the upper time limit in our measurements. One concern is the fact that the size of the files that are shared on P2P networks today is bigger than at the time of the study performed in **[9]**. We shall compensate for this by making the sensitive data available over several sessions.

Metric ID	M3
Name	Reveal and download rate
Description	Measures the time it takes for sensitive data to be downloaded by a potentially malicious peer
Metric	Data that is accidentally shared can represent a serious threat. This metric’s goal is to measure the average time it takes before such data is revealed and downloaded by a potentially malicious peer
Formula	Score = $\frac{100}{60} \times Average\_time\_before\_download$
Purpose	To get a numerical representation of the chance that accidentally shared data will be downloaded

Implementation

Reveal and download rate

Different types of sensitive data will be shared through the P2P servent (For a list and description see "Appendix C – False Sensitive Data"). This data will be made to appear real, but will not contain any sensitive or real information.

The time that the data is made available for download will need to be finite; a timescale has therefore been created. The data will be made available to other peers for the average peer life-time identified in **[9]** to simulate a real situation.

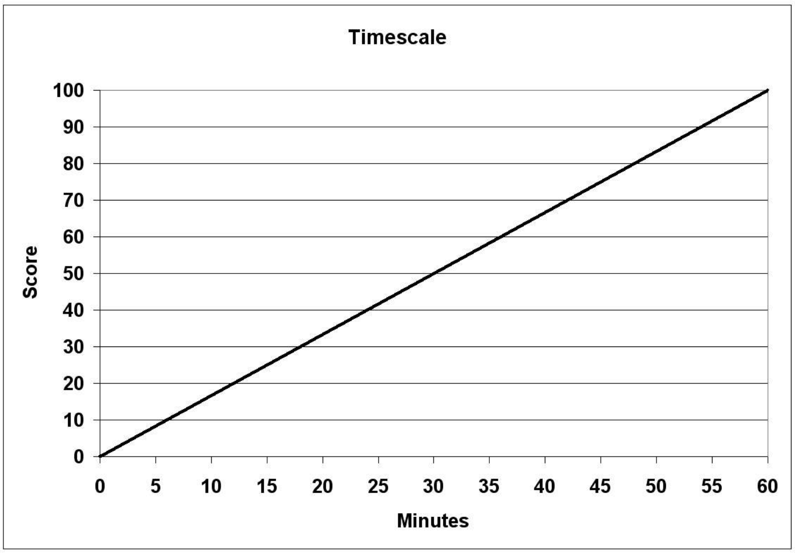


Figure 3 - Reveal and download timescale

Every category of data will be made available 10 times. If it has not been downloaded within the allotted time span it will score the highest possible on the scale. If it gets downloaded we will compute the average time before download and use this in the timescale. The timescale starts at 0 minutes and ends at 60 minutes.

Logging will be activated on the P2P servent to monitor downloads; if this is not possible from within the servent then external logging will be implemented.

Frequency	This metric will need to be updated after any major software upgrade on the user interface of the P2P servent
Cost	Very low. Requires some amount of time to set up the monitoring software, but the monitoring itself is usually performed automatically. No expenditure of money.
Indicators	If the shared information is quickly revealed and downloaded this will have a negative effect on security and will increase the risk of information leakage. This will also indicate if there are malicious users that actively scan the network for sensitive data.
Validity	Since benign peers have little use for sensitive data we can be quite sure that peers that download such data have malicious intentions. The introduction of an upper time limit may reduce the validity of this metric since this limit will vary in the different P2P implementations (average time was only computed for Gnutella and Napster in **[9]**). And today we see that users are downloading large media files, this may impact the average online time for peers. The fact that we make the sensitive data available 10 times will help to simulate the fact that users today download larger files and thus probably remain connected to the P2P network for longer stretches of time than before.
Reliability	The limit of sharing sensitive data only 10 times before calculating average time will increase the risk of random errors having an effect on the result. But the limit is necessary, if a limit did not exist the experiment would become to time consuming. This can be alleviated by sharing it more than the specified 10 times, but will also result in an increased need for more time and resources.

Table 10 - Reveal and Download Rate

3.4 Accidental Sharing Rate

When users share their data with other peers, there is a chance that they will accidentally share information that should not be made available for download by others. This data can be of sensitive or confidential nature (ex: credit card information, private photos, bank statements, certificates). Malicious peers can then search through the P2P network using common keywords for sensitive data; this will result in a list of peers who are sharing such data. This is a very efficient and low risk method for malicious users to gather data. Recent studies have found that a large number of P2P users have made available for download sensitive documents like their tax returns, e-mail inboxes or credit card information **[30, 48]**. Once available these files could be used to commit fraud, invade privacy, or even commit identity theft **[23]**. Systems that are designed to match users' capabilities, requirements and expectations will result in systems that minimize human error **[47]**. P2P developers have been criticized of making it difficult for users to identify what files they are sharing **[47, 48]**. If P2P application were designed with these flaws in mind, this could result in P2P applications that would make it harder for users to accidentally share sensitive data.

The main factor that affects accidental sharing is GUI (Graphical User Interface) design **[29]**. If the GUI is not designed to prevent users from accidentally sharing data, then users will eventually share data that is of sensitive or confidential nature. This was a major problem in the early implementations of both Kazaa and Gnutella. This has, according to the developers, been solved in later releases of the software. Other factors that affect accidental sharing is that these systems are used by millions of users, many of whom have a limited knowledge of computer systems. The large number of users, combined with the powerful search tools incorporated into the P2P software, makes it easy for malicious users to filter the network and find sensitive files. Furthermore; inexperienced users can have a hard time finding out what they have shared, thus sensitive material can be exposed over an extended time period without the users knowledge **[23, 29]**.

This is made worse by today's situation where many users share one computer in a single machine, multiple user environment. This is a common configuration in many homes today. In such a setting a parent could have a secure VPN connection from his home computer to his job, thus enabling him to work from home. Any other family member, most likely a teenage son or daughter since 41% of P2P users are aged between 12 and 18 years **[30]**, could then inadvertently share the parents confidential files to others on the P2P network.

Files that are accidentally shared on P2P network come in a wide variety **[30, 48]**:

- Tax return papers with social security numbers
- Medical files
- Confidential legal documents
- Personal correspondence
- Business files
- Political records
- Resumes with job histories, salary information and references
- Criminal records

The following metric attempts to measure what the chance is that a user shares more data than he intended to share. Users will be given a list containing the name and paths of several files and folders that they must share on the P2P network; they will not be told the purpose of this exercise since this may affect their behavior and make them more careful than normal. A list of the files that are to be shared is presented in "Appendix A – Shared Files". This list contains a limited number of files to ensure that the measurement can be gathered in a reasonable amount of time per person tested. It would have been preferable to have a more extensive list of files, but this would have made the metric too time consuming for the scope of this thesis. As it is now, the measurement should take approximately 5-10 minutes per person. Had the measurement been more time consuming than this, then it would have been harder to find subjects willing to participate in the experiment.

There are 2 characteristics we need to consider when we choose test persons:

- Age: Those aged between 17-24 years are the heaviest users of P2P file-sharing **[18]**, while the majority of P2P users are aged between 12-18 years **[30]**. But the introduction of P2P technology in business applications means that people that are older and with little prior knowledge of P2P file-sharing will come into contact with this technology. This group will also need to be represented in the population.
- Computer knowledge: How familiar are the participants with computer systems. This will be ranked by categories:

- Intermediate: User can perform simple tasks like writing letters, checking mail and browsing the Internet
- Knowledgeable: User can perform complicated task like installing new software, configure mail and network settings. Has prior knowledge of P2P file sharing.

We need to have an even distribution of these characteristics in our population for the experiment to reflect the real-world situation.

Metric ID	M4
Name	Accidental sharing rate
Description	Measures the chance of users accidentally sharing data.
Metric	Data that is accidentally shared can represent a serious threat for information leakage; this metrics goal is to measure the chance of a user accidentally sharing data.
Formula	$\frac{number\_of\_test\_persons\_that\_accidentally\_share}{number\_of\_test\_persons} \times 100$
Purpose	To get a numerical representation of the chance that users of a given P2P servent accidentally share data

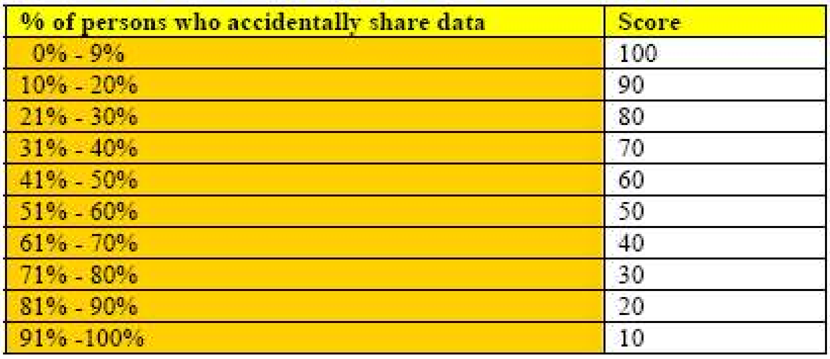
Implementation

Accidental sharing

Users will be given a list of files and folders that they are supposed to share (See “Appendix A – Shared Files”). They will then proceed to share the files and folders listed. The test persons will have access to any online documentation regarding the P2P servent they are using, but will not be allowed to cooperate with each other.

When the test person has completed the list, a review will be performed. If any data is shared that is not specified in the list, then the test person has accidentally shared data.

When all the test persons have performed this task, the number of persons that accidentally shared data will be tallied and this number will be used to measure the rate at which data is accidentally shared in the specified P2P servent. We shall get a percentage score from the formula above. This score is to be used in the table below to identify the security score.



% of persons who accidentally share data	Score
0% - 9%	100
10% - 20%	90
21% - 30%	80
31% - 40%	70
41% - 50%	60
51% - 60%	50
61% - 70%	40
71% - 80%	30
81% - 90%	20
91% -100%	10

Frequency

This metric will need to be updated after any major software upgrade on the user interface of the P2P servent

Cost

High. This measurement requires a large amount of time to complete, the amount of time can be decreased by lowering the number of participants. But this will affect the reliability of the measurement. It might be necessary to offer participants something in order to get them to participate.

Indicators

If a high percentage of the test persons accidentally share data this will indicate that the user interface is not well enough designed and represent a threat to the security level of the software.

Validity	The validity of this measurement will depend on the construction of the file and folder system in the test. This must resemble a real world file system, and files must not be intentionally made to confuse the test person. The applications will have to be distributed to the test persons in a random order. If all the test persons get the applications in the same order then this will affect the results since the users will be inexperienced with P2P application to begin with, but more experienced when they reach the last application. Furthermore, we cannot be sure that the scores defined in the metric will reflect the real world; this will need to be evaluated when more data has been gathered.
Reliability	If this measurement is to be reliable the file system must remain constant throughout the test, and preferably remain constant in later measurements as well. The number of test persons will also affect reliability; a high number of test persons will reduce the effect of random errors.

Table 11 - Accidental sharing

3.5 New Vulnerabilities Introduced

The installation of P2P software can introduce new vulnerabilities into the system. These vulnerabilities can be exploited by a malicious user, and are often quite easy to detect by using vulnerability detection software. This detection software is used by administrators to find and remove vulnerabilities in the system, but the software is also used by malicious users to probe potential victims during the information gathering phase.

P2P software can introduce several weaknesses into a system, but the most common are:

- Listening on new ports: When P2P software is installed this software needs to contact other peers and receive data from these. To do this, the P2P software opens new ports on the computer and listens on these. These unsupervised ports can represent a security risk.
- Known Bugs: P2P software is no different than other types of software; it has bugs. These bugs come in a variety of types; some are harmless, while others can represent a serious security risk. To exploit bugs in software requires a knowledgeable attacker since an exploit will have to be developed. This takes time and requires an expert knowledge.
- Known Exploits: Exploits that have been published on the Internet represent a serious security risk. While taking advantage of a software bug requires a skilled attacker, an exploit can be run by an inexperienced attacker (commonly referred to as a “Script-Kiddie”). This results in a much larger population of attackers for this type of vulnerability.

There exists a whole host of different vulnerability detection software (ISS, NeWT, Nessus, etc). We have decided to use NeWT, a free, Open-Source detection software package, which is compatible with Windows XP. NeWT will be able to scan the hosts and determine if there are changes introduced by the installation of P2P software. The most likely result will be that the P2P applications will open and listen on several new ports. This can represent a possible way into the system for an attacker or lead to information leakage. P2P software is no different from other types of software when it comes to bugs that can be exploited, but the fact that a computer with P2P software will be interconnected with a large number of other untrusted peers will lead to an increased exposure to malicious users and software **[30]**.

Metric ID	M5
Name	New vulnerabilities
Description	Measures the amount of system vulnerabilities.
Metric	The installation of P2P software can introduce new vulnerabilities into a system.
Formula	*(Total number of discovered vulnerabilities after installation)-(Total number of vulnerabilities before installation)= Vulnerabilities introduced by P2P servent*
Purpose	To get a numerical representation of the amount of vulnerabilities introduced by the P2P software

Implementation

New vulnerabilities:

Vulnerability detection software: NeWT Security Scanner

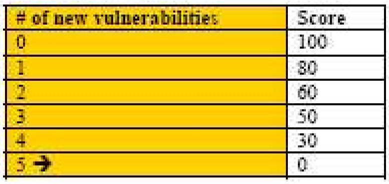
Software version: 2.1

A vulnerability scan is performed prior to the installation of the P2P servent. Each discovered vulnerability needs to be documented with the following properties:

- Type of vulnerability
- Degree of seriousness (as reported by the detection software)

Another vulnerability scan must be performed after the installation of the P2P servent: The P2P servent must be operational while the scan is in progress.

A comparison must be performed of the vulnerabilities detected before and after installation of P2P software. From the formula above we get the number of new vulnerabilities introduced. This number is then used in the table below to identify the security score.



# of new vulnerabilities	Score
0	100
1	80
2	60
3	50
4	30
5 →	0

Frequency

This metric will need to be updated after any major software versions of the P2P servent

Cost

Low. Vulnerability scan is performed automatically, but will have to be manually analyzed. No expenditure of money.

Indicators

A high number of new vulnerabilities will indicate that the software is vulnerable to probing by malicious users. A probe can uncover these vulnerabilities and thus give an attacker a way into the system.

Validity	This measurement will be valid since vulnerability detection software is used by attackers in the information gathering phase. P2P software that introduces several new vulnerabilities will weaken the overall security of the system. But we cannot be 100% sure that the software will detect all new vulnerabilities or that the scoring assigned will reflect the real- world implications on security. However, in this metric we do not take into consideration how dangerous each individual threat is, this might be necessary to accurately assess the danger posed to the system.
Reliability	To keep this measurement reliable it is important that the same detection software is used throughout the measurement.

Table 12 - New Vulnerabilities

4 Experimental Data

4.1 Introduction

The following sections contain information relating to the setup of the experimental network and the measurements gathered through the metrics proposed earlier in this paper.

4.2 Environment

The test environment consists of a network of four computers connected through a router with a connection to the Internet. These computers have identical installations of operating system and software components. The standard configuration is backed up and is used to restore the systems to the default state after each P2P application has been assessed. It is necessary to return the systems to the default state between experiments in order to avoid that fragments from earlier assessments interfere with later measurements.

Standard Configuration
OS:
Windows XP Professional with Service Pack 2
Components:
Spybot – Search and destroy” v1.3 or higher
XoftSpy v4.1 or higher
Filemon logging software
NeWT Security Scanner
AVG Virus Control (Free Edition)
Firefox v.1

Table 13 - Standard Configuration

This standard configuration will be scanned for spyware and adware before being stored in backup, any spyware or adware detected will be removed to ensure that the standard configuration will be as clean as possible. However, we cannot be 100% sure that all spyware or adware will be removed; these hidden programs can affect our later measurements (especially the vulnerability scan). A vulnerability scan will also be performed, and any vulnerability detected will be documented. This documentation will be used to compare the number and type of vulnerabilities detected before and after the installation of P2P software.

Computer Hardware	
Computer 1 (DeskTop)	Computer 2 (DeskTop)
CPU: Pentium 150MHz	CPU: Pentium 350 MHz
RAM: 128 MB	RAM: 128 MB
HD: 2 GB	HD 4 GB
Computer 3 (LapTop)	Computer 4 (LapTop)
CPU: Pentium 2,2 GHz (Celeron)	CPU: Pentium 1,7 MHz (Centrino)
RAM: 256 MB	RAM: 1 GB
HD: 20 GB	HD: 80 GB

Table 14 - Computer Hardware

The four computers will be interconnected trough a router; this router will be the only connection point to the Internet. The router will have basic firewall functionality enabled (ingress and egress filtering, limitations on ICMP messages, MAC-address filtering, etc), but the computers themselves will not have any host based firewall software installed. P2P software needs to have a legitimate way through the firewall in order to operate; a host firewall must therefore be configured to allow P2P traffic, thus rendering it ineffective against attacks aimed at P2P traffic and communication **[30]**.

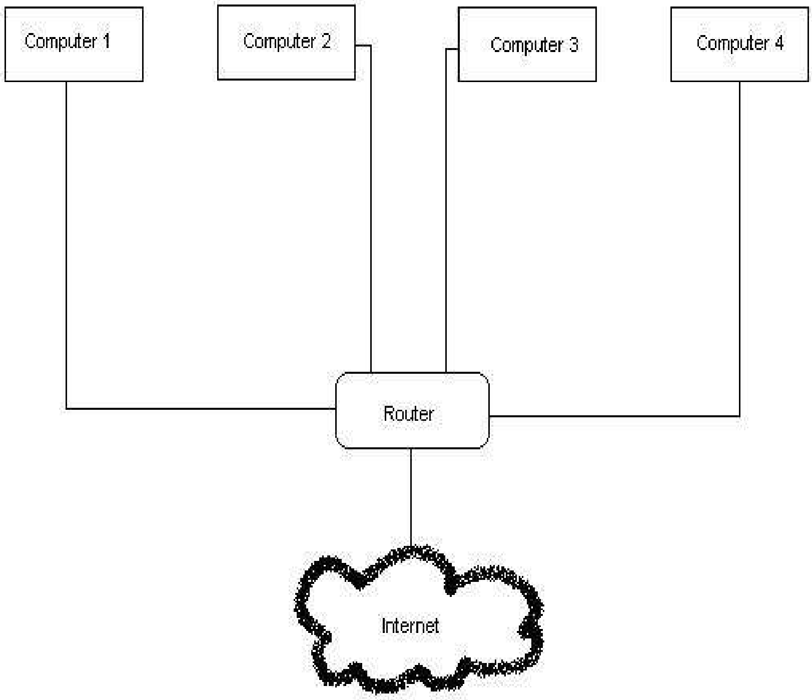


Figure 4 - Network architecture

4.3 Test Persons

The measurements performed in M4 require the use of several test persons. We have decided to make use of 20 test persons. The number of test persons has been set so low because of the time consuming tasks that need to be performed. A higher number of test persons would have been preferable but this would have made the measurements too time consuming for the scope of this thesis. Similar tests have been performed by other cited research papers. These have also had small groups of test persons **[29, 62, 63]**. All of the test persons have prior experience with computers and use computers for at least 2 hours each day. 12 of the test persons have used P2P file-sharing applications before and are knowledgeable in the use of computer systems. The others have only heard about P2P technology through recent media publicity, but have some experience with common computer tasks and systems. The age of the test persons is between 15 and 55 (Figure 5 shows age distribution in out test population).

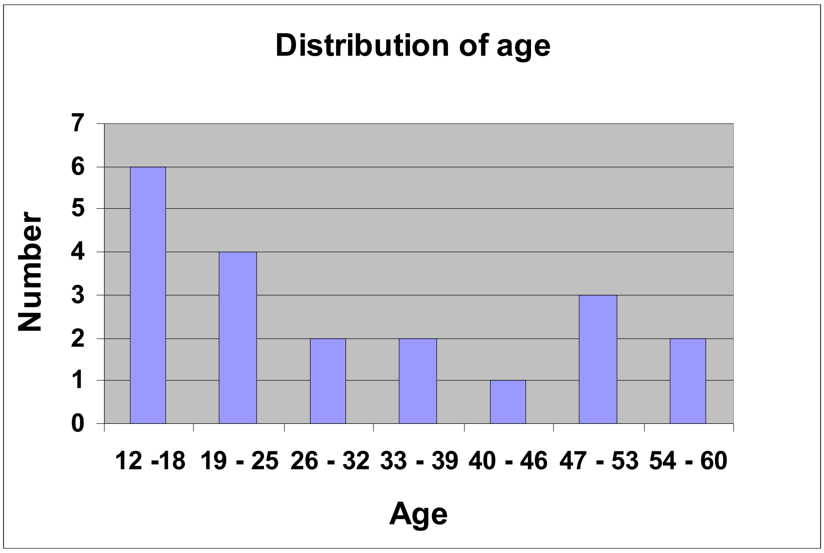


Figure 5 - Graph: Age distribution in test population

The following two figures show the distribution of gender and knowledge in our test population. The similarities of these 2 figures are coincidental and should not be interpreted to mean that all the men in our population were experienced, while the women were novices.

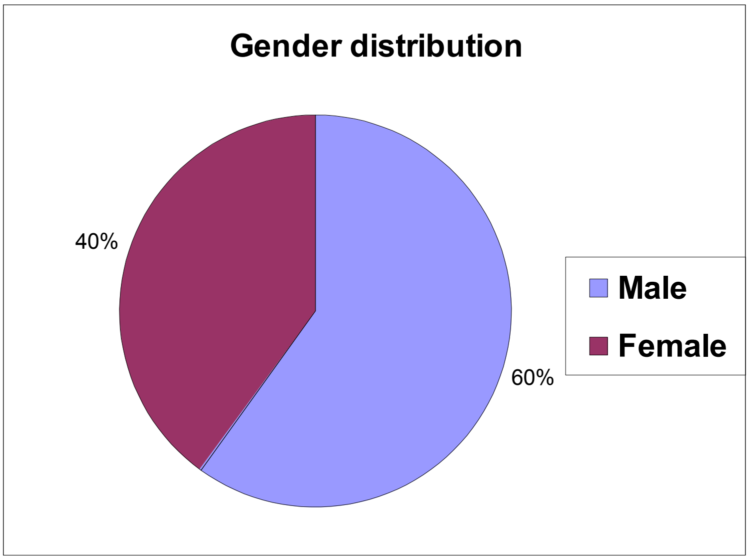


Figure 6 - Graph: Gender distribution

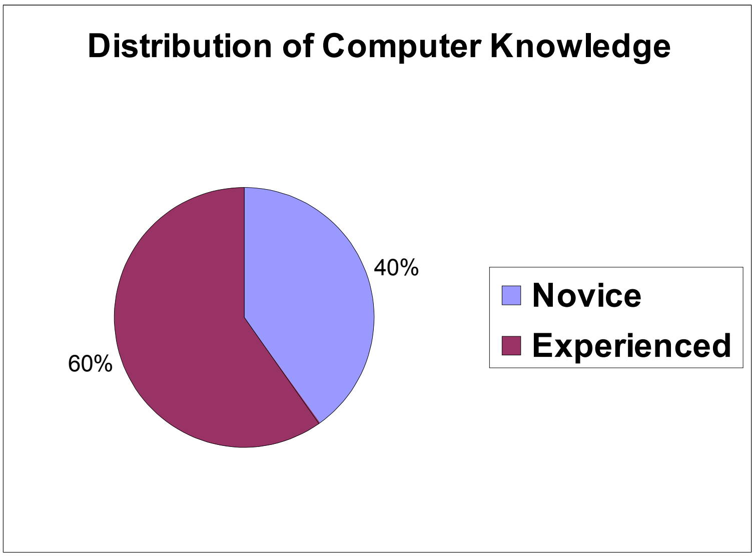


Figure 7 - Graph: Distribution of knowledge

5 Experimental Results

5.1 Security Assessment of Kazaa

The following will detail the security measurements performed on the Kazaa servent. Kazaa is commercial software developed by Sharman Networks, and has been one of the more popular P2P servents in recent time.

5.1.1 M1: Number of Lines of Code

Kazaa is proprietary software and information on code lines is not publicly available. But by contacting the developers we were able to get an estimated number of the actual code lines that Kazaa consists of. This number does not take into consideration any of the bundled third party software or Spyware/Adware.

$$\text{Number of theoretical errors} = \frac{350000}{1000} \times 30 = 10500$$

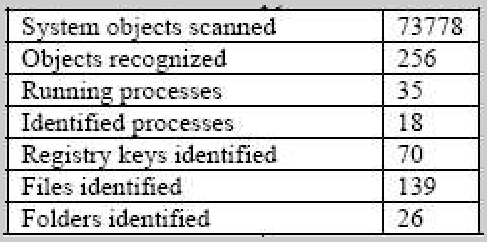
# of errors	Score
0 - 1000	100
1001 - 2000	90
2001 - 3000	80
3001 - 4000	70
4001 – 5000	60
5001 – 6000	50
6001 – 7000	40
7001 – 8000	30
8001 -9000	20
9001 - 1000	10
10001 =>	0

Table 15 - M1 Score for Kazaa

5.1.2 M2: Number of Spyware and Adware Included

Kazaa's user documentation states that the software contains adware but no spyware; this is also stated in the EULA[7]. Upon installation several unwanted and undocumented programs were installed and shortcuts added to the desktop (ex: "Play Poker Now!" and "Your Free Casino Chips").

XoftSpy



System objects scanned	73778
Objects recognized	256
Running processes	35
Identified processes	18
Registry keys identified	70
Files identified	139
Folders identified	26

Vendor/Program Name	Category
Adware P2P Networking	Adware
Claira/Gain/Gator/Dashbar	Data Miner
Kazoom	P2P Adware
TopPicks	Data Miner
TopSearch	BHO[8]
Web P2P Installer	Downloader
Bargain Buddy Bundle	Adware
DownloadWare	Adware
Twain-Tech	BHO
Cydoor	Data Miner

Table 16 - Types of Spyware/Adware in Kazaa

SpyBot S&D did not discover any spyware/adware that was missed by XoftSpy.

ID *i*	**Spyware**	**Score i**	**Number**	**Score**
1	Cookie	1	0	0
2	Pop-up	1	0	0
3	Adware	2	4	8
4	Downloader	3	1	3
5	Tracks	3	3	9
6	Hijacker	5	2	10
7	Malware	10	0	0
8	Keylogger	20	0	0
			Total Score:	70

Table 17 - M2 Score for Kazaa

[7] End User Licence Agreement

[8] BHO is a program module that is used to hijack browser functionality

5.1.3 M3: Reveal and Download Rate

Kazaa has a documented maximum user mass of 4.6 million **[39]**. On average, our Kazaa servent was simultaneously connected to approximately 2.3 million peers during the measurements. Even in such a large population Kazaa's search algorithm makes it easy to identify specific material. This can work in the malicious user's favor.

#	**Date and Time**	**Time of download**	**File downloaded**	**Elapsed time**
1	28.02.05 15:30	15:52	Admin Password.doc	22 minutes
2	01.03.05 13:00	13:26	Budget.xls	26 minutes
3	02.03.05 15:00	15:48	Admin Password.doc	48 minutes
4	02.03.05 16:00	16:24	Classified.jpg	24 minutes
5	02.03.05 17:00	No Download		60 minutes
6	03.03.05 12:00	12:14	Visa creditcard + pincode.txt	14 minutes
7	03.03.05 13:00	13:47	Password.jpg	47 minutes
8	03.03.05 14:00	No Download		60 minutes
9	04.03.05 18:00	18:07	Private photo.jpg	7 minutes
10	04.03.05 19:00	19:48	Inbox.pst	48 minutes
Average Time:				35.6 minutes
Total Score:				59

Table 18 - M3 Average Time for Kazaa

A more detailed list of downloaded files is given in "Appendix B – Downloaded Files".

5.1.4 M4: Accidental Sharing:

It is well know that many users of the Kazaa P2P network are accidentally sharing sensitive data **[29, 30]**. The following metric will measure how likely it is that such data is accidentally made available to other users of the Kazaa P2P network.

- 19 out of 20 managed to share all the specified files
- 5 out of 20 shared files that were not on the list
- 1 of 20 did not manage to share any data

% of persons who accidentally share data	Score
0% - 9%	100
10% - 20%	90
21% - 30%	80
31% - 40%	70
41% - 50%	60
51% - 60%	50
61% - 70%	40
71% - 80%	30
81% - 90%	20
91% -100%	10

Table 19 - M4 Accidental sharing for Kazaa

5.1.5 M5: Number of New Vulnerabilities:

By using the NeWT security scanner before and after the installation of the Kazaa servent we can discover potential security weaknesses introduced by Kazaa. Examples of such weaknesses will include software exploits, open ports and information leakage.

Before installation of Kazaa			
ID	***Protocol / Port / Application***	***Description***	***Threat***[9]
1	TCP / 445 / Microsoft-ds	It was possible to log into remote host using a NULL session	-
After installation of Kazaa			
ID	***Protocol / Port / Application***	***Description***	***Threat***
1	TCP / 80 / HTTP	Kazaa HTTP server running	High
2	TCP / 445 / Microsoft-ds	It was possible to log into remote host using a NULL session	-
3	TCP / 3531 / Joltid	Web server running on this port	-
4	TCP / 3574 / dmaf-server	Kazaa HTTP server running	High
5	TCP / 1214 / Kazaa	Kazaa server running	High

Table 20 - M5 Vulnerabilities Kazaa

# of new vulnerabilities	**Score**
0	100
1	80
2	60
3	50
4	30
5 =>	0

Table 21 - M5 Score for Kazaa

5.1.6 Security Score for Kazaa:

Metric ID	**Score**
M1	0
M2	70
M3	53
M4	80
M5	30
Security Score:	**46.6**

Table 22 - Security Score for Kazaa

[9] As reported by detection software

5.2 Security Assessment of Limewire

The following will detail the security measurements performed on the Limewire servent. Limewire is software developed by Limewire-Downloading.

5.1.1 M1: Number of Lines of Code

Limewire is Open-Source software and information on code lines is publicly available. By downloading the source and counting the number of coded lines we were able to get a close estimate of the number. This number was later confirmed by the developers of Limewire. This number does not take into consideration any potentially bundled third party software.

$$\text{Number of theoretical errors}= \frac{190000}{1000} \times 30 = 5700$$

# of errors	Score
0 - 1000	100
1001 - 2000	90
2001 - 3000	80
3001 - 4000	70
4001 – 5000	60
5001 – 6000	50
6001 – 7000	40
7001 – 8000	30
8001 -9000	20
9001 - 1000	10
10001 =>	0

Table 23 - M1 Score for Limewire

5.1.2 M2: Number of Spyware and Adware Included

Limewire's user documentation and EULA states that there is no spyware, adware or virus bundled with its software. This is also the company's main selling point.

Table 24 - Types of Spyware/Adware in Limewire

SpyBot S&D did not discover any additional spyware/adware.

ID *i*	Spyware	Score i	Number	Score
1	Cookie	1	0	0
2	Pop-up	1	0	0
3	Adware	2	0	0
4	Downloader	3	0	0
5	Tracks	3	0	0
6	Hijacker	5	0	0
7	Malware	10	0	0
8	Keylogger	20	0	0
			Total Score:	100

Table 25 - M2 Score for Limewire

5.1.3 M3: Reveal and Download Rate

Limewire has a documented user mass of 1 million **[40]**. On average our servent had access to 119656 Mb of data distributed over 47230 files. Limewire uses a centralized + decentralized network topology where there are supernodes that help manage incoming queries from peers connected to the specified supernode.

#	Date and Time	Time of download	File downloaded	Elapsed time
1	03.03.05 18:30	No Download		60 minutes
2	03.03.05 19:30	No Download		60 minutes
3	04.03.05 15:00	No Download		60 minutes
4	04.03.05 16:00	16:55	Kid.jpg	55 minutes
5	04.03.05 17:00	No Download		60 minutes
6	05.03.05 12:00	No Download		60 minutes

7	05.03.05 13:00	No Download		60 minutes
8	05.03.05 14:00	No Download		60 minutes
9	06.03.05 18:00	18:39	Shower.mpg	39 minutes
10	06.03.05 19:00	19:12	Private.jpg	12 minutes
			Average Time:	52.6
			Total Score:	87.7

Table 26 - M3 Average Time for Limewire

A more detailed list of downloaded files is given in "Appendix B – Downloaded Files".

5.1.4 M4: Accidental Sharing Rate:

The following metric measures how likely it is that sensitive data is accidentally made available to other users of the Limewire P2P network.

- 20 out of 20 managed to share all the specified files
- 7 out of 20 shared files that were not on the list
- 0 of 20 did not manage to share any data

% of persons who accidentally share data	Score
0% - 9%	100
10% - 20%	90
21% - 30%	80
31% - 40%	70
41% - 50%	60
51% - 60%	50
61% - 70%	40
71% - 80%	30
81% - 90%	20
91% -100%	10

Table 27 - M4 Accidental Sharing for Limewire

5.1.5 M5: Number of New Vulnerabilities:

By using the NeWT security scanner before and after the installation of the Limewire servent we can discover what potential security weaknesses are introduced by Limewire. Examples of such weaknesses include software exploits, open ports and information leakage.

Before installation of Limewire			
ID	***Protocol / Port / Application***	***Description***	***Threat***
1	TCP / 445 / Microsoft-ds	It was possible to log into remote host using a NULL session	-
After installation of Limewire			
ID	***Protocol / Port / Application***	***Description***	***Threat***
1	TCP / 6346 / gnutella-svc	Port is open	-
2	TCP / 445 / Microsoft-ds	It was possible to log into remote host using a NULL session	-
3	TCP / 45100 / unknown	Port is open	Unknown
5	TCP / 135 / epmap	CIFS server running	-

Table 28 - M5 Vulnerabilities for Limewire

# of new vulnerabilities	**Score**
0	100
1	80
2	60
3	50
4	30
5 =>	0

Table 29 - M5 Score for Limewire

5.1.6 Security Score for Limewire:

Metric ID	**Score**
M1	50
M2	100
M3	100
M4	70
M5	50
Security Score:	74

Table 30 - Security Score for Limewire

5.3 Security Assessment of Grokster

The following will detail the security measurements performed on the Grokster servent. Grokster is commercial software developed by Grokster LTD.

5.1.1 M1: Lines of Code

Grokster is proprietary software and information on code lines is not publicly available. Grokster is approximately the same size in MB as Kazaa, operates on the same network structure as Kazaa and has substantial similarities in GUI. It is likely that Grokster and Kazaa are also similar when it comes to the number of coded lines. This number does not take into consideration any of the included third party software or Spyware/Adware.

$$\text{Number of theoretical errors}=\frac{300000}{1000}\times 30=9000$$

# of errors	Score
0 - 1000	100
1001 - 2000	90
2001 - 3000	80
3001 - 4000	70
4001 – 5000	60
5001 – 6000	50
6001 – 7000	40
7001 – 8000	30
8001 -9000	20
9001 - 1000	10
10001 =>	0

Table 31 - M1 Score for Grokster

5.1.2 M2: Number of Spyware and Adware

Groksters user documentation states that the software contains adware but no spyware; this is also stated in the EULA. Upon installation, a large number of unwanted and undocumented program-shortcuts were added to the desktop (ex: “Guardster”, “Get 10$ Free at Zodiac Casino”, “Blackjack Ballroom Casino”, “Casino”, “High Rollers Club Casino”, “Free Website” and “Sportsbook”). These shortcuts were described in the user agreement, but it required a great deal of detailed reading to identify the relevant chapters.

XoftSpy	
Vendor/Program Name	Category
Adware P2P Networking	Adware
TopRebates	Adware
Claira/Gain/Gator/Dashbar	Data Miner
AdRoar	Malware
BroadCastPC	Data Miner
TopPicks	Data Miner
TopSearch	BHO
Web P2P Installer	Downloader
Cydoor	Data Miner
BTV	Dialer
Bargain Buddy Bundle	Adware
TopMoxie	Adware
Wast	Adware
Tracking Cookie	Cookie
DownloadWare	Adware
Twain-Tech	BHO

Table 32 - Types of Spyware/Adware in Grokster

In addition SpyBot S&D discovered:

Vendor/Program Name	Category
Altnet	Adware
MyWay.MyBar	Adware

Table 33 - Additional Types of Spyware/Adware in Grokster

ID *i*	**Spyware**	**Score i**	**Number**	**Score**
1	Cookie	1	1	1
2	Pop-up	1	0	0
3	Adware	2	8	16
4	Downloader	3	1	3
5	Tracks	3	4	12
6	Hijacker	5	1	5
7	Malware	10	2	30
8	Keylogger	20	0	0
			Total Score:	35

Table 34 - M2 Score for Grokster

5.1.3 M3: Reveal and Download Rate

Grokster has an average user mass of 2.5 million **[41].** On average our Grokster servent was simultaneously connected to approximately 2.2 million peers during the measurements.

#	**Date and Time**	**Time of download**	**File downloaded**	**Elapsed time**
1	05.03.05 12:30	13:23	Credit Card.jpg	53 minutes
2	05.03.05 13:00	No Download		60 minutes
3	06.03.05 18:00	18:47	Windows 2K server password.txt	47 minutes
4	06.03.05 19:00	No Download		60 minutes
5	06.03.05 20:00	20:31	Credit card data.xls	31 minutes
6	07.03.05 12:00	No Download		60 minutes
7	07.03.05 13:00	13:19	Inbox.pst	19 minutes
8	07.03.05 14:00	14:23	Private.jpg	23 minutes
9	07.03.05 15:00	15:12	Classified.jpg	12 minutes
10	07.03.05 16:00	16:44	Home budget.xls	44
			Average Time:	40.9 minutes
			Total Score:	68.2

Table 35 - M3 Average Time for Grokster

A more detailed list of downloaded files is given in "Appendix B – Downloaded Files".

5.1.4 M4: Accidental Sharing Rate:

The following metric will measure how likely it is that sensitive data is accidentally made available to other users of the Grokster P2P network.

- 20 out of 20 managed to share all the specified files
- 9 out of 20 shared files that were not on the list
- 0 of 10 did not manage to share any data

% of persons who accidentally share data	**Score**
0% - 9%	100
10% - 20%	90
21% - 30%	80
31% - 40%	70
41% - 50%	60
51% - 60%	50
61% - 70%	40
71% - 80%	30
81% - 90%	20
91% -100%	10

Table 36 - Accidental sharing for Grokster

5.1.5 M5: Number of New Vulnerabilities:

By using the NeWT security scanner before and after the installation of the Grokster servent we can discover what potential security weaknesses are introduced by Grokster. Examples of such weaknesses will include software exploits, open ports and information leakage.

Before installation of Grokster			
ID	***Protocol / Port / Application***	***Description***	***Threat***
1	TCP / 445 / Microsoft-ds	It was possible to log into remote host using a NULL session	-
After installation of Grokster			
ID	***Protocol / Port / Application***	***Description***	***Threat***
1	TCP / 80 / HTTP	Kazaa HTTP server running	High
2	TCP / 135 / epmap	CIFS server running	-
3	TCP / 445 / Microsoft-ds	It was possible to log into remote host using a NULL session	-
4	TCP / 3531 / Joltid	Web server running on this port	-
5	TCP / 1731 / msiccp	Kazaa HTTP server running on this port	High
6	TCP / 3574 / dmaf-server	Kazaa HTTP server running on this port	High
7	TCP / 3531 / Unknown	Port is open	Unknown
8	TCP / 1214 / Kazaa	Kazaa server running on this port	High

Table 37 - M5 Vulnerabilities for Grokster

# of new vulnerabilities	Score
0	100
1	80
2	60
3	50
4	30
5 =>	0

Table 38 - M5 Score for Grokster

5.1.6 Security Score for Grokster:

Metric ID	Score
M1	20
M2	35
M3	88
M4	60
M5	0
Security Score:	**40.6**

Table 39 - Security Score for Grokster

5.4 Security Assessment of Shareaza

The following will detail the security measurements performed on the Shareaza servent. Shareaza is Open-Source software, and can be downloaded from www.shareaza.com.

5.1.1 M1: Number of Lines of Code

Shareaza is Open-Source software and information on code lines is publicly available. By downloading the source and counting the number of coded lines we were able to get a close estimate of the number.

$$\text{Number of theoretical errors}=\frac{250000}{1000}\times 30=7500$$

# of errors	Score
0 - 1000	100
1001 - 2000	90
2001 - 3000	80
3001 - 4000	70
4001 – 5000	60
5001 – 6000	50
6001 – 7000	40
7001 – 8000	30
8001 -9000	20
9001 - 1000	10
10001 =>	0

Table 40 - M1 Score for Shareaza

5.1.2 M2: Number of Spyware and Adware Included:

Shareaza's user documentation and EULA states that there is no spyware, adware or virus bundled with its software.

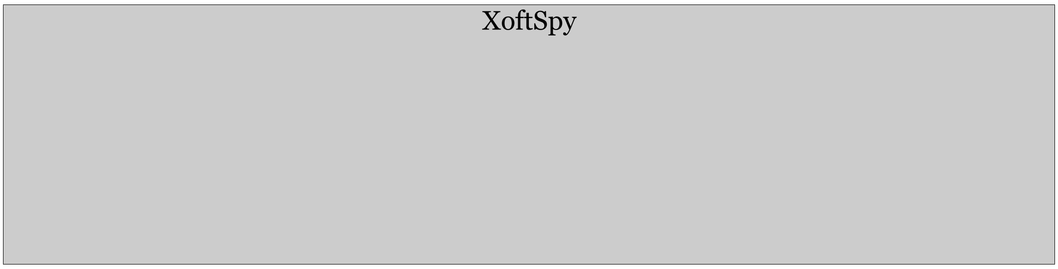


Table 41 - Types of Spyware/Adware in Shareaza

SpyBot S&D did not discover any additional spyware/adware.

ID *i*	**Spyware**	**Score i**	**Number**	**Score**
1	Cookie	1	0	0
2	Pop-up	1	0	0
3	Adware	2	0	0
4	Downloader	3	0	0
5	Tracks	3	0	0
6	Hijacker	5	0	0
7	Malware	10	0	0
8	Keylogger	20	0	0
			Total Score:	100

Table 42 - M2 Score for Shareaza

5.1.3 M3: Reveal and Download Rate:

Shareaza connects several different P2P networks together (Gnutella1, Gnutella2 and eDonkey2000). This makes it hard to get reliable data on how many connected users there are. Shareaza uses a centralized + decentralized network topology where there are supernodes that help manage incoming searches from peers connected to the specified supernode.

#	Date and Time	Time of download	File downloaded	Elapsed time
1	10.03.05 12:20	13:10	Visa creditcard + pincode.txt	50 minutes
2	10.03.05 13:20	13:34	Inbox.pst	14 minutes
3	10.03.05 14:00	No Download		60 minutes
4	10.03.05 15:00	No Download		60 minutes
5	10.03.05 16:00	16:51	Mastercard + pincode.txt	51 minutes
6	10.03.05 17:00	No Download		60 minutes
7	11.03.05 13:00	No Download		60 minutes
8	11.03.05 14:00	No Download		60 minutes
9	11.03.05 15:00	15:47	Credit card data.xls	47 minutes
10	11.03.05 16:00	No Download		60 minutes
			Average Time:	52.2
			Total Score:	87

Table 43 - M3 Average Time for Shareaza

A more detailed list of downloaded files is given in "Appendix B – Downloaded Files".

5.1.4 M4: Accidental Sharing:

The following metric measures how likely it is that sensitive data is accidentally made available to other users of the Shareaza P2P network.

- 20 out of 20 managed to share all the specified files
- 3 out of 20 shared files that were not on the list
- 0 of 20 did not manage to share any data

% of persons who accidentally share data	Score
0% - 9%	100
10% - 20%	90
21% - 30%	80
31% - 40%	70
41% - 50%	60
51% - 60%	50
61% - 70%	40
71% - 80%	30
81% - 90%	20
91% -100%	10

Table 44 - M4 Accidental Sharing for Shareaza

5.1.5 M5: Number of New Vulnerabilities Introduced:

By using the NeWT security scanner before and after the installation of the Shareaza servent, we can discover what potential security weaknesses are introduced by Shareaza. Examples of such weaknesses will include software exploits, open ports and information leakage.

Before installation of Shareaza			
ID	***Protocol / Port / Application***	***Description***	***Threat***
1	TCP / 445 / Microsoft-ds	It was possible to log into remote host using a NULL session	-
After installation of Shareaza			
ID	***Protocol / Port / Application***	***Description***	***Threat***
1	TCP / 6346 / gnutella-svc	Port is open	-
2	TCP / 445 / Microsoft-ds	It was possible to log into remote host using a NULL session	-
3	TCP / 135 / epmap	CIFS server running	-

Table 45 - M5 Vulnerabilities for Shareaza

# of new vulnerabilities	**Score**
0	100
1	80
2	60
3	50
4	30
5 =>	0

Table 46 - M5 Score for Shareaza

5.1.6 Security Score for Shareaza:

Metric ID	**Score**
M1	30
M2	100
M3	87
M4	90
M5	60
Security Score:	**73.4**

Table 47 - Security Score for Shareaza

6 Discussion

6.1 Analysis of the obtained Data:

In the following sections we shall analyse the experimental data. The analysis goes into details on the specific data obtained on each metric.

6.1.1: Number of Lines of Code (M1)

The results from M1 indicate that the size and complexity of P2P applications can vary substantially. All the applications that were tested in this paper had approximately the same functionality (search functionality, GUI and built-in media player). Even so, the size in coded lines varies by as much as 45% between the largest and the smallest application (Figure 9). It is reasonable to assume that as the complexity and size of an application grows, so does the chance that programming errors will go undetected **[66]**.

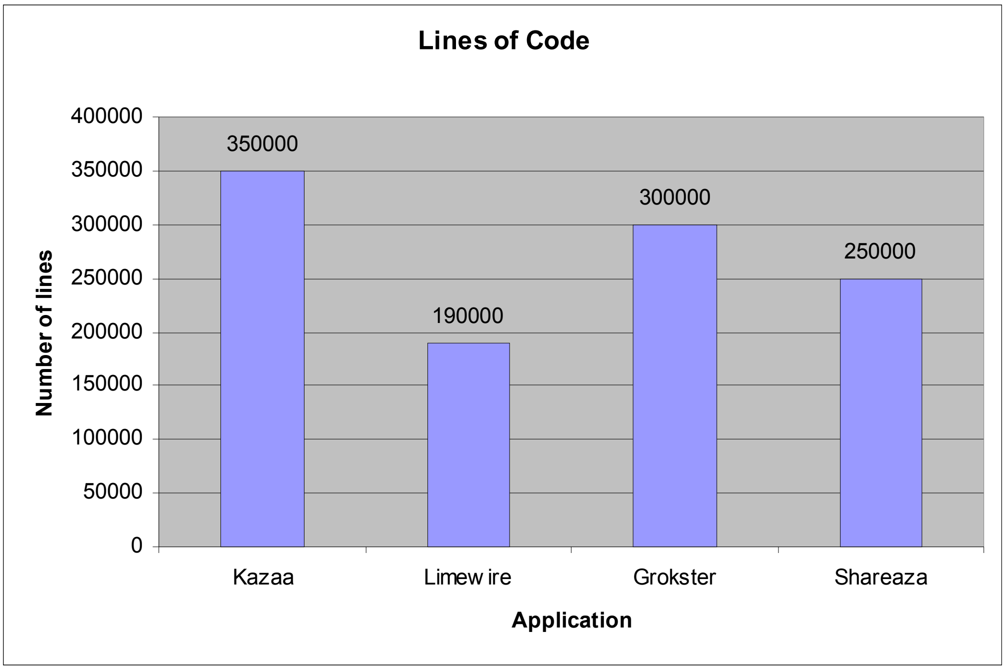


Figure 8 - Graph: Number of coded lines

We can see from Figure 8 that Limewire by far has the smallest number of coded lines in our measurement, while Kazaa tops the list with approximately 350000 lines of code. It is also of interest to note that the applications based on Open-Source

(LimeWire and Shareaza) are at the lower end of the scale, while the applications based on proprietary source occupy the higher end of the scale.

The scoring (Figure 9) reflects that smaller P2P programs will be easier to inspect and maintain. These applications have therefore scored higher in our measurements than applications that have many code lines. A side effect of having a small number of coded lines is that the applications usually will require less processing and storage on the host computer, and they will usually be quicker to load into memory. However, smaller programs might not have all the functionality that many users demand today.

We are aware that large programs do not necessarily contain more programming errors than small programs. But the history of P2P file-sharing application development has shown that security and programming quality has not been the developer's main focus. We have therefore based our interpretations of the result on that this lack of security and programming quality will cause an increase in programming errors as the size of the P2P file-sharing application grows..

It might be that we have been too harsh when assigning a score of zero to Kazaa, but we have decided to put the upper limit at 330000 lines of code (approximately 10.000 potential programming errors). Most of the open-source applications manage to stay well below this limit. This indicates that a P2P file-sharing application does not need to exceed this number in order to offer the necessary functionality and at the same time operate efficiently.

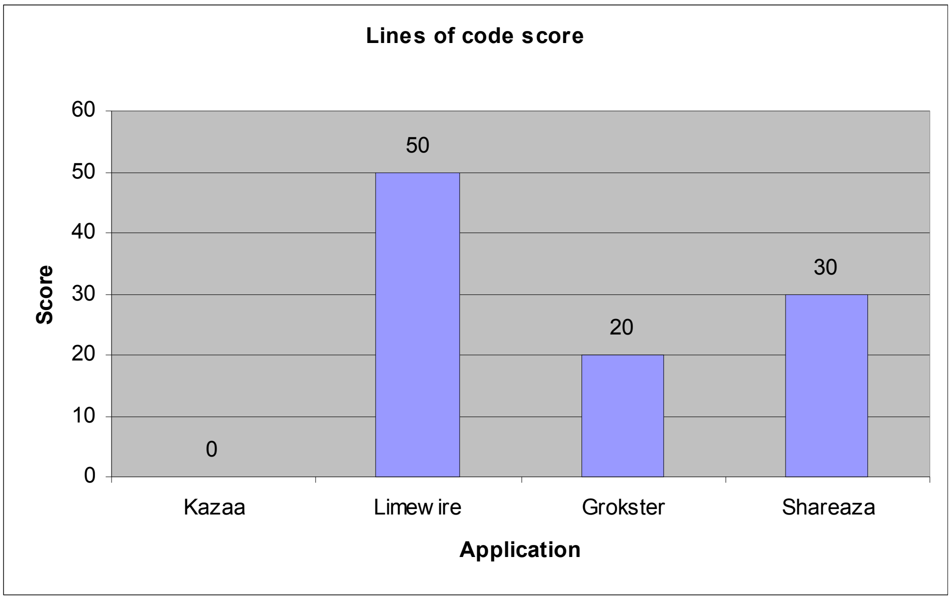


Figure 9 - Graph: Security score M1

6.1.2: Number of Spyware and Adware Included (M2)

As we can see from Figure 10 there seems to be a distinct difference between the Open-Source and proprietary P2P applications in our experiment. Only the proprietary software (Kazaa and Grokster) come bundled with third party software. This would seem to make the choice easy for users who do not want spyware/adware installed on their computers.

More surprising is the fact that one of the applications actually contained malware in the form of trojans. During installation of Grokster the installed antivirus program (AVG Free Edition) detected 2 different trojans. To remove the possibility that this was caused by other programs a complete format was performed and Grokster reinstalled, but the problem persisted. This was also the case when a different antivirus program was used (Norton Antivirus 2005). Also of note was the fact that Grokster did not come with a visible uninstall option, and did not appear in the "Add or remove programs" option within the control panel. This made it quite hard to remove all components without performing a format or system restore.

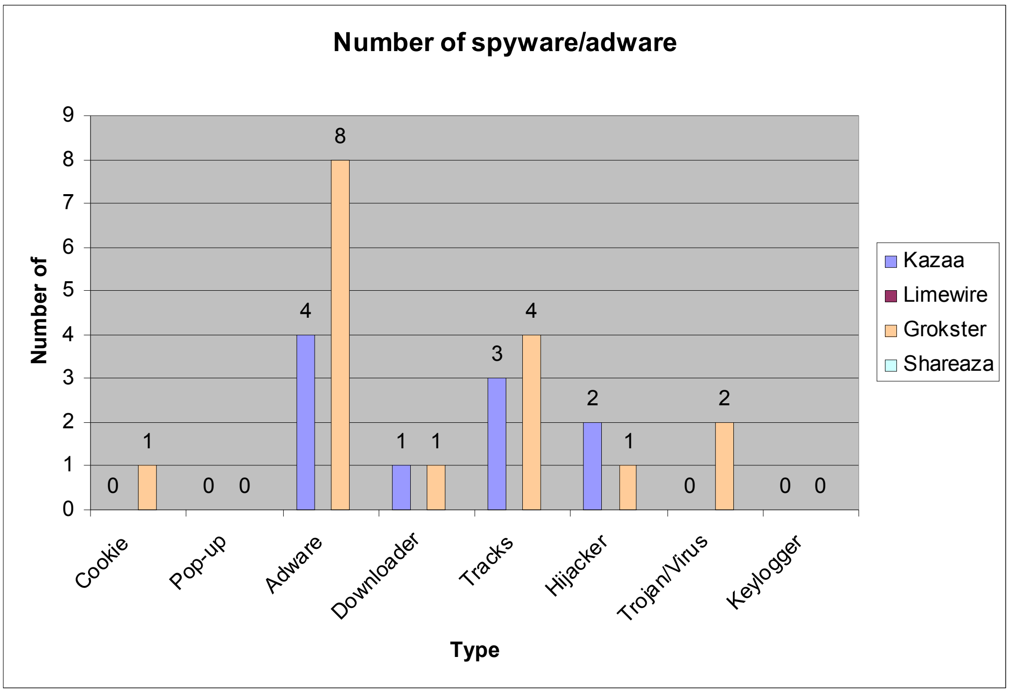


Figure 10 - Graph: Number of spyware/adware

Also of interest in Figure 10 is the fact that the distribution of spyware/adware is quite similar between Kazaa and Grokster. There could be several different reasons for this. This thesis proposes:

- This is the maximum distribution of spyware/adware that the developers can bundle with the software before they see a drop in the rate of software download as a consequence of unhappy users.
- There are a limited number of companies interested in advertising through the use of spyware/adware. Advertisements that are perceived as intrusive or potentially dangerous could have a negative impact on the advertising company's reputation.
- This is the amount of spyware/adware that is necessary for the developers to continue development of the software and have a reasonable income from their product.

At the time of writing these reasons are pure speculation, but should later measurements show a similar distribution this could possibly merit further studies.

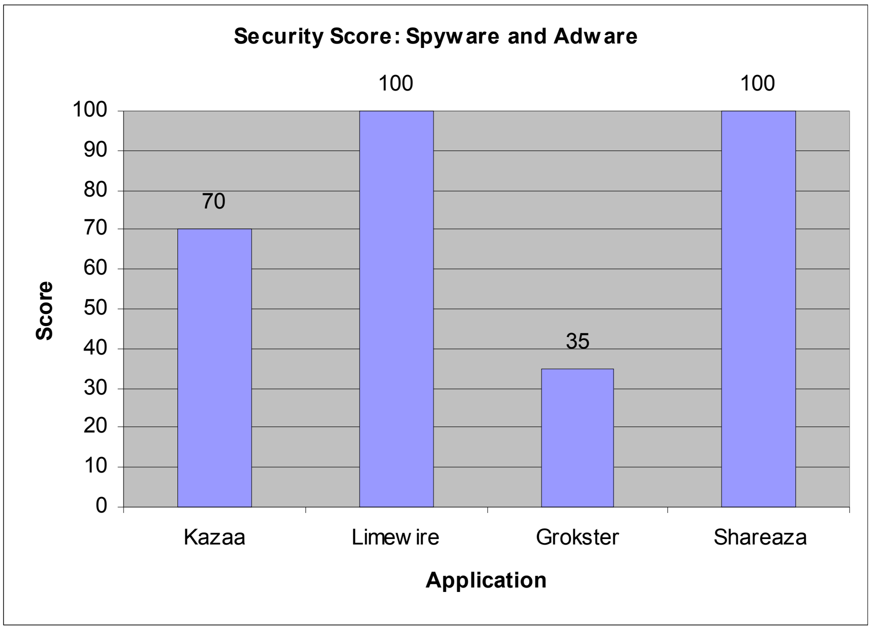


Figure 11 - Graph: Security score M2

As shown by the security score for M2 (Figure 11), only Limewire and Shareaza managed to achieve a full score. Kazaa managed to score a respectable 70 points; this is mainly due to the fact that the spyware/adware bundled with Kazaa is of a type that does not pose a high security risk (adware and tracks). Grokster's security score reflects the fact that Grokster came bundled with some highly suspicious software that was classified as trojan by the antivirus software. Grokster also contains a large amount of adware (twice the amount that Kazaa has). This greatly increases the risk of software conflicts and puts a strain on the host computer's processing. Comments from the test persons from M3 indicated that the adware in Grokster had a negative effect on their subjective experience with the application. One user was heard saying: "If I have to click one more popup I'll go insane!".

Our results also seem to match those achieved in **[35]**, where it was shown that Kazaa contained a large number of adware, while Limewire did not contain any kinds of spyware or adware.

6.1.3: Reveal and Download Rate (M3)

Our experiment on the reveal and download rate was conducted over a period of 11 days and involved 4 computers running identical P2P software during those sessions. We see from Figure 12 that Kazaa and Grokster had more early downloads of sensitive data than Limewire and Shareaza (note that 60 minutes means that there was no download during the session). This would seem to indicate that the P2P network that Kazaa and Grokster operate on have a larger or more effective population of malicious/curious users than Limewire or Shareaza's networks.

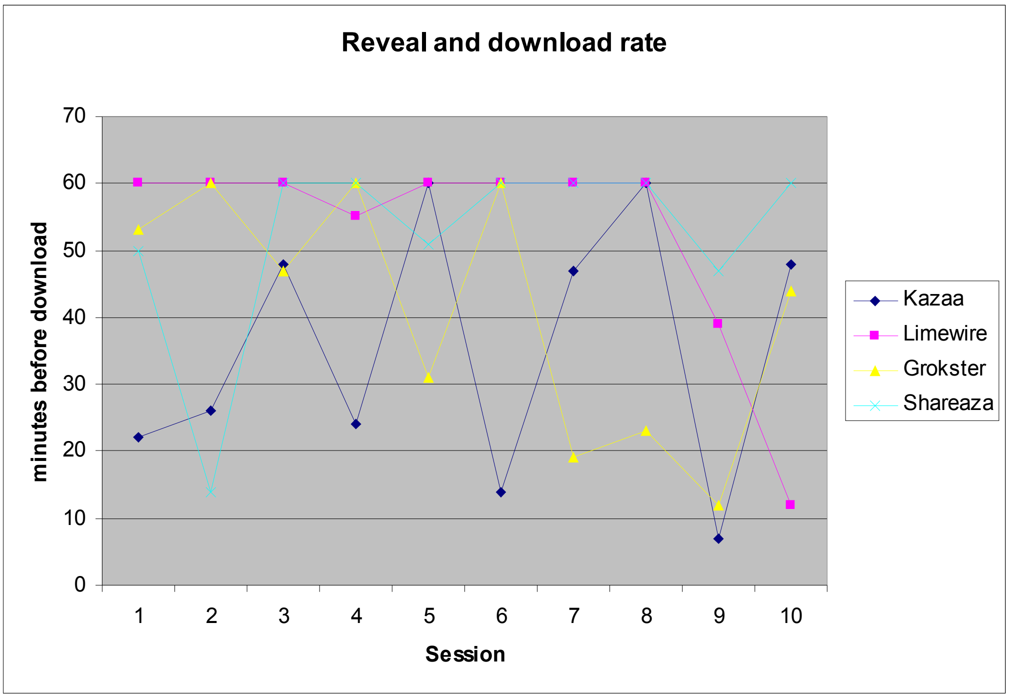


Figure 12 - Graph: Reveal and download rate

It is also interesting to note that Kazaa and Grokster had more sessions that ended in download of sensitive data than Limewire and Shareaza. The total amount of downloads during the test period was 40, and the distribution was as represented in Figure 13. The difference in downloads are actually so large that Kazaa or Grokster individually have more downloads than Limewire and Shareaza combined. It should be noted that the Limewire network that our servent was connected to during the test period was considerably smaller than the networks that Kazaa and Grokster operate on. During the test period the Limewire network had an average user mass of 965.000 with 119 GB shared in the network (as reported by the Limewire servent). In comparison, Kazaa had an average user mass of 2.3 million users with 52.948.992 GB shared in the network (as reported by the Kazaa servent).

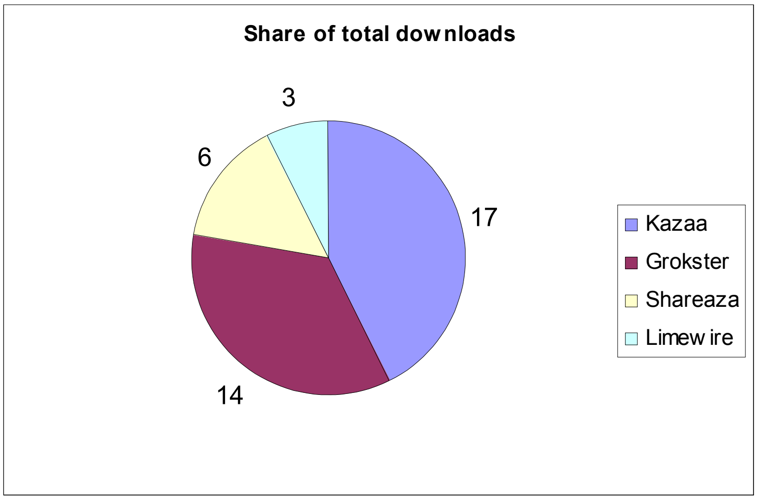


Figure 13 - Graph: Share of total downloads

We can see (Figure 14) that the average time before downloading any sensitive data was somewhat lower on Kazaa and Grokster. This indicates that users should be careful about what kinds of data they share in the network and should not believe themselves safely hidden in the large user population. All of the tested applications had well developed search functionality that can be used by malicious users to find data of a sensitive nature. It is a paradox that as search functionality becomes stronger, more efficient and more user friendly, exposed sensitive data becomes easier for malicious users to reveal and download. If unsure about how to correctly share data, users should therefore consult either the user documentation or someone who has experience with the software.

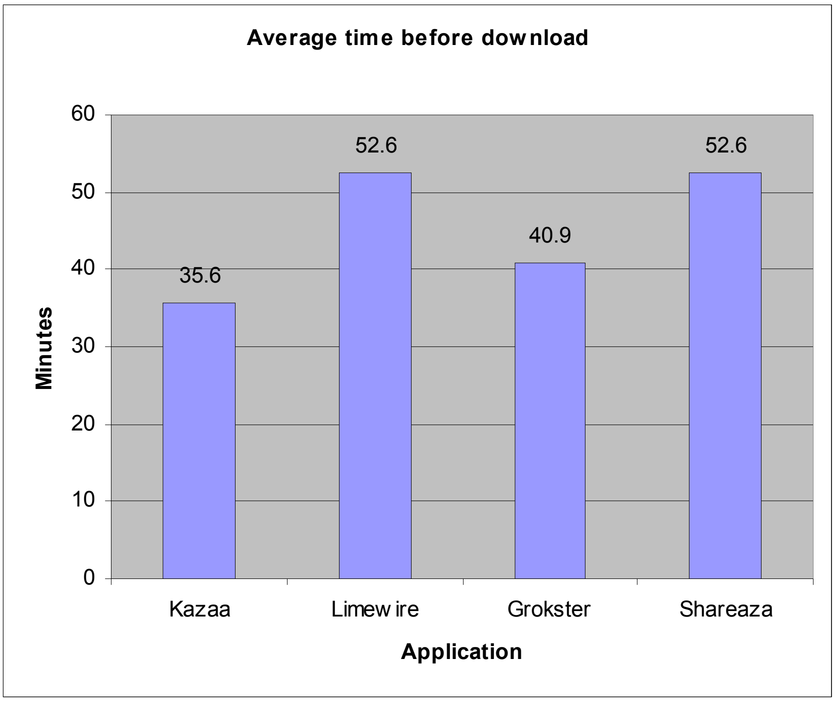


Figure 14 - Graph: Average time before download

When it comes to security scoring on M3 we can see again that Open-Source applications come out on top, while proprietary applications remain at the lower end. It is important to note that Kazaa and Grokster operate on closed networks (ex: FastTrack) **[3]**. To use such networks, developers of P2P applications often have to pay a substantial fee. These networks are not interconnected with the open networks on which Limewire and Shareaza operate. This means that we are dealing with at least two different populations of users. Based on the results gained from this limited test it would seem that the network on which Kazaa and Grokster operate contains a larger or more active population of malicious or curios users than that of Limewire's and Shareaza's networks. This conclusion is based on the measurements represented in Figures 12, 13 and 14. We can see from those three figures that sensitive files shared through Kazaa or Grokster will be downloaded more times and faster than if the same files had been shared in Limewire or Shareaza.

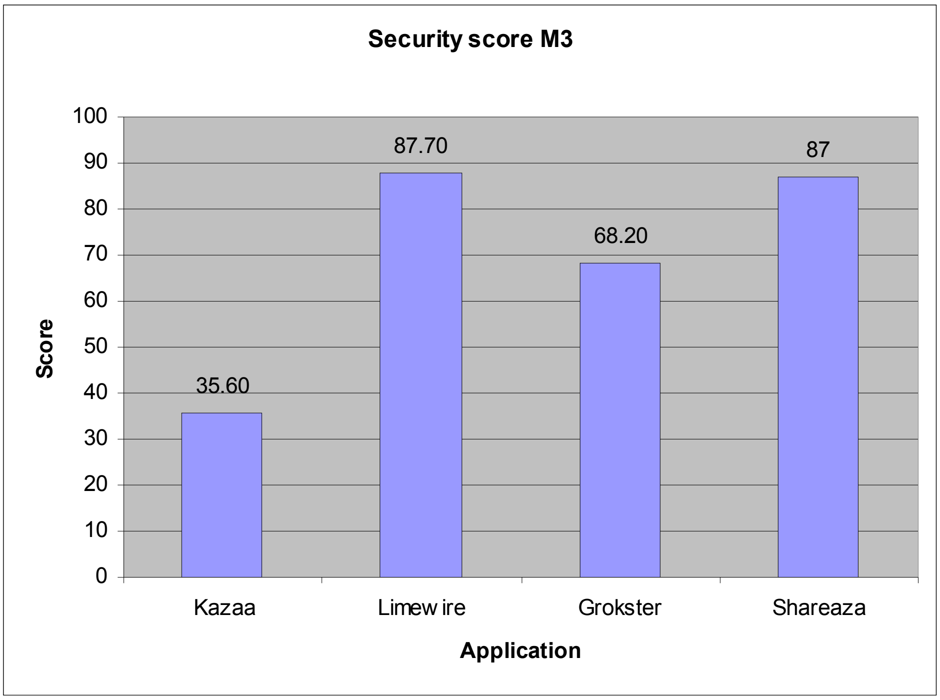


Figure 15 - Graph: Security score M3

6.1.4: Accidental Sharing Rate (M4)

The test population consisted of 20 people with varying degrees of computer knowledge, but what all of them had in common was the fact that they had some prior experience with computers and could be classified as either experienced or intermediate level within the field of computer literacy. The testing was conducted on computers that had the various P2P applications installed, and the participants had access to online user documentation. The participants were given a list (see "Appendix A – Shared Files") that contained all the files that should be shared. This list was provided in both English and Norwegian. Participants could use as much time as they wanted as there was no time limit imposed on them.

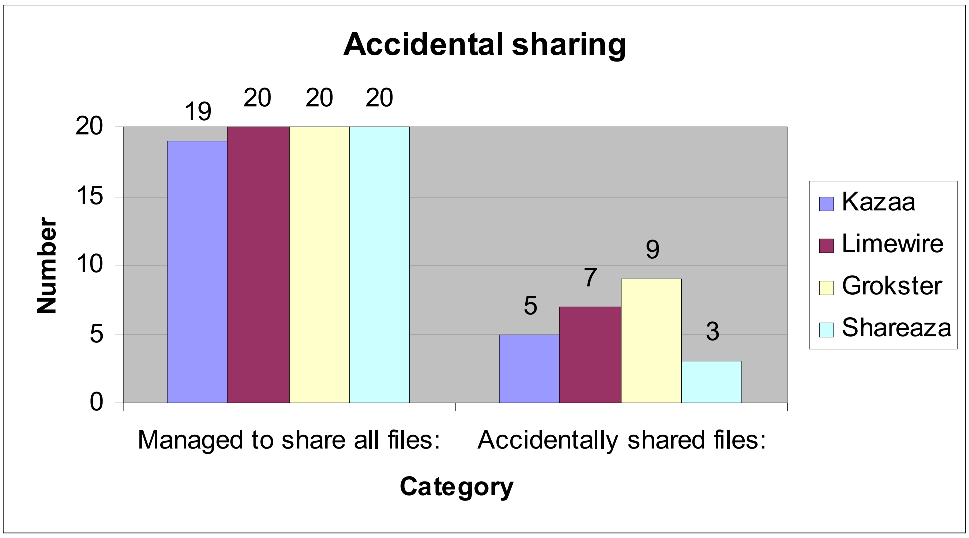


Figure 16 - Graph: Accidental sharing

As we can see (Figure 16), the test persons encountered few problems in sharing the required files. Only one person did not manage to share all of the files listed. There was however quite a large number of test persons who shared more files than they were supposed to. In the case of Grokster this amounted to 45% of the participants. Other applications also showed a high rate of accidental sharing. It is surprising that such a high percentage of our test population accidentally shared data on the P2P networks. Many of these incidents of accidental sharing can be traced back to the way the different applications show the users what files are shared. Several of the test persons found it difficult to identify the files they where sharing on the network. This was made worse by the fact that each application had its own way of representing shared files.

Our measurements show that accidental sharing occurs on a regular basis. It is therefore a real danger that users can accidentally share sensitive data with other users on the P2P networks. And it is important that developers take this into consideration when designing the applications. Our measurements show that it is too easy to accidentally share data in today's P2P applications. This is further evidenced by the amount of sensitive data shared on P2P networks today. "Appendix D –Examples of Shared Sensitive Data" contains some examples of sensitive data that was found during a 1 hour session using the various P2P applications with a limited set of queries. Names and other identifying information have been removed from the documents. There were also other files, but these where deemed too sensitive to show publicly (ex: medical records, lists of credit-card numbers and letters sent between lawyers and their clients).

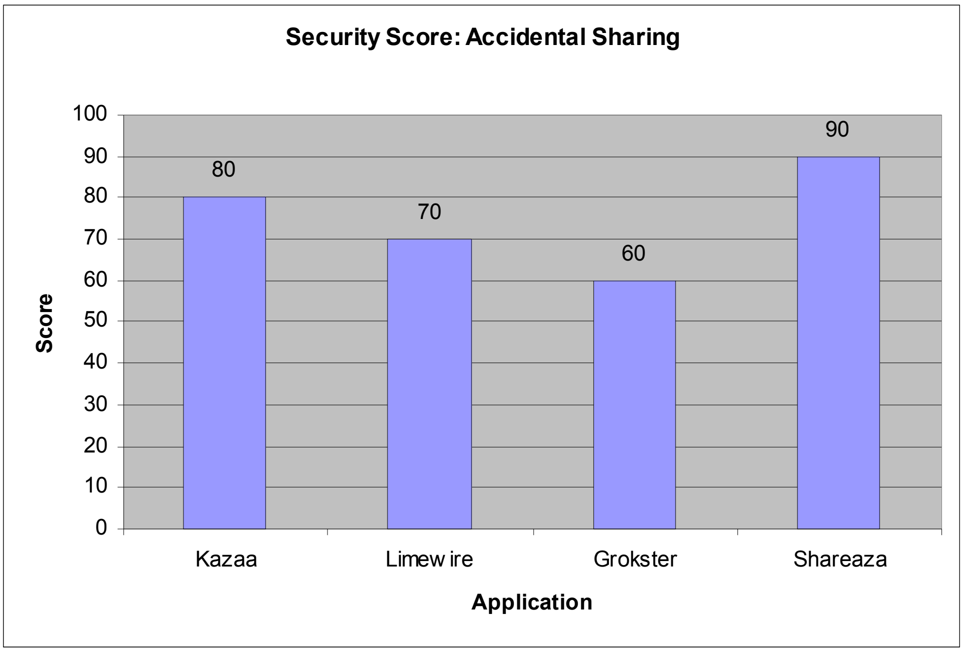


Figure 17 - Graph: Security score M4

The security score (Figure 17) reflects our findings in Figure 18 that show us that Grokster has the highest amount of accidental sharing, while Shareaza achieved the lowest amount of accidental sharing.

6.1.5: Number of New Vulnerabilities Introduced (M5)

The vulnerability measurement was performed simultaneously on four different computers; all computers had the same software and operating system installed. The experiment was divided into four sessions where each session concentrated on one P2P application. During the session, the P2P application was operational and it was sharing data with its P2P network.

It is important to note that the only vulnerabilities we were capable of detecting in our experiment (by using NeWT) was if the applications opened and listened on previously unused ports. We had hoped that NeWT would be able to detect other potential vulnerabilities in the P2P applications (ex: known exploits and bugs). This was not the case in our measurements.

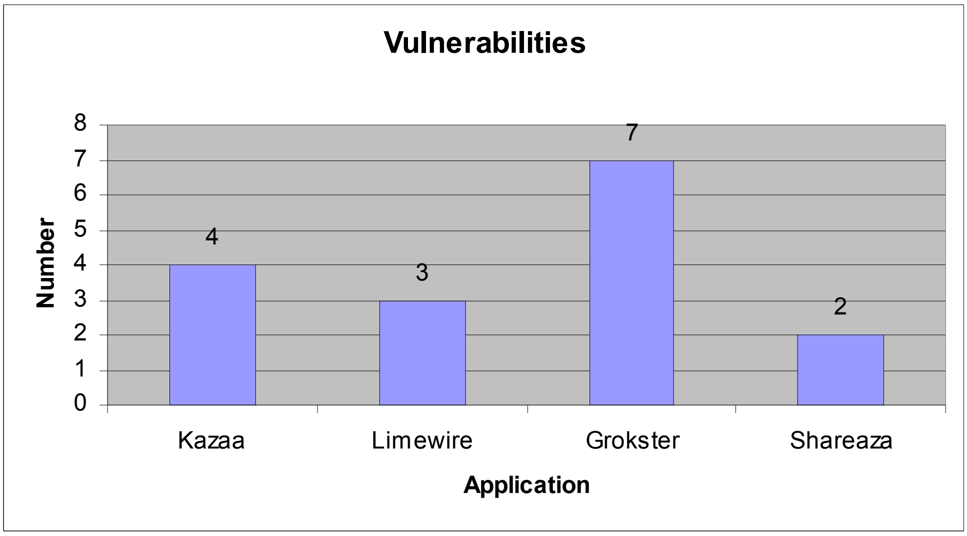


Figure 18 - Graph: Vulnerabilities introduced

As we can see from Figure 18 there was a substantial difference in the number of introduced vulnerabilities between Grokster and the other P2P applications. While Shareaza only introduced 2 new potential vulnerabilities, Grokster introduced 7. All of these vulnerabilities consisted of listening on new ports, but while Shareaza only needed 2 additional ports, Grokster used 7 ports, one of those ports could not be tied to any identifiable application and the service using this port was classified as unknown. We can not eliminate the possibility that one or more of the ports used by Grokster could be used by one of the trojans that were installed together with Grokster.

Our measurements show that a P2P application should be more than capable off operating efficiently with only 2-3 additional ports opened. Opening and listening on more than that number of ports did not seem to have any discernible advantages. This should be further examined by measuring other applications and documenting how these applications operate when it comes to port opening/listening. It is also interesting to note that several of the applications did not consistently use the same port numbers, but changed these at different intervals of time. We were not able to discern the external or internal factors that governed this.

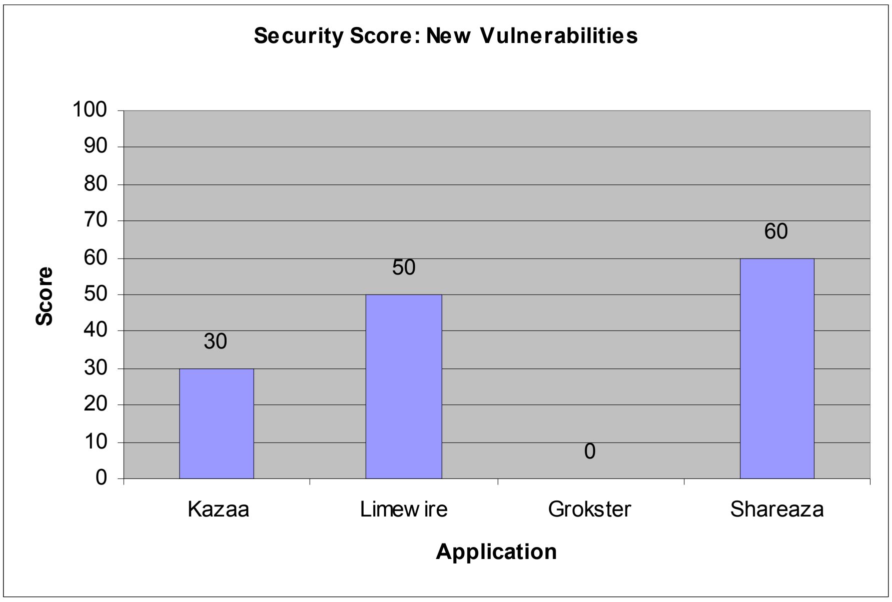


Figure 19 - Graph: Security score M5

Kazaa, Limewire and Shareaza scores reflect the fact that they introduced relatively few new vulnerabilities. Compared to theses three applications Grokster introduced a substantially larger number of new vulnerabilities. This is reflected in the fact that Grokster did not score any points in M5.

7 Conclusion and Further Work

As shown in the earlier sections, the assessed P2P applications have scored quite differently on different metrics. When designing a ranked list of the assessed applications we have taken into consideration the fact that the users will have different interests in P2P applications. Some will want an application that does not contain spyware/adware, while others may be more interested in the rate of accidental sharing. It is also likely that users will want a combination of different security metrics. The following table presents the results in a way we believe will make such comparisons possible.

Application	M1	M2	M3	M4	M5	Score
Limewire	50	100	100	70	50	**74**
Shareaza	30	100	87	90	60	**73.4**
Kazaa	0	70	53	80	30	**46.6**
Grokster	20	35	88	60	0	**40.6**

Table 48 - Security Matrix

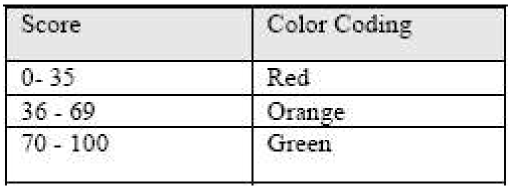
M1: Lines of Code

M2: Number of spyware and adware

M3: Reveal and download rate

M4: Accidental Sharing

M5: Number of new vulnerabilities



Score	Color Coding
0- 35	Red
36 - 69	Orange
70 - 100	Green

Table 49 - Explanation of Security Matrix

The security matrix shows that there is little difference in the total score when comparing Limewire with Shareaza, and comparing Kazaa with Grokster. It is therefore important for users to define their security focus when it comes to the choice of the P2P application. But the security matrix makes it obvious that there is a substantial difference in the security level between the proprietary and Open-Source software. There are very few instances where it will be advantageous to choose one of the proprietary P2P applications tested here instead of one of the Open-Source applications.

Based on the results gained from the metrics, the conclusion reached in this thesis is that of the applications tested it is the Open-Source applications that perform the best. The metrics developed in this paper have produced values that can be used to differentiate between the different P2P applications security performance. Most of the metrics do not require a large investment in time or money, and can therefore be performed quickly and often. This is an important factor since the measurements should be repeated when developers release new versions of their software. However, the calibration of the metrics needs to be evaluated by other researchers in order to assess if they accurately reflect the real-world security implications. As they stand now the tables that govern the conversion process of measured values into estimated security score, might be influenced by the author's subjective understanding of the source material on which the process is based. This should be evaluated by independent parties.

It would be advantageous to consider refining M4 and M5. In its current state M4 takes a somewhat large amount of time and resources to complete when compared to the rest of the metrics. One alternative approach would be to perform something called a Cognitive Walkthrough to analyze the interface of the different P2P application and identify areas in the interface that could cause accidental sharing. This approach has been used in **[29, 43]** and has achieved some interesting results. A cognitive walkthrough was considered used in this paper, but was at the time deemed to be too subjective to be used as a foundation for a metric. This might need further consideration in later research. M5 did not produce the intended results. It was hoped that we would be able to detect known exploits and weaknesses in the P2P software; unfortunately all that the chosen detection software was able to detect was if the P2P software listened on new ports. Later revision of M5 might consider combining data from sites that track exploits and software bugs with the data gathered from the detection software (in this case NeWT). This should give a more complete picture of the vulnerabilities introduced by the P2P software.

8 Contributors

I would like to thank all those who have contributed to this thesis. Without their aid this work would not have been possible. My guidance counselor Professor Slobodan Petrovic has been a great help during the development of this thesis, taking the time to read, comment and discuss both the research problems, results and layout.

I am also grateful for the contributions received from Greg Bildson (Limewire), the online support staff for Shareaza and members of the support staff responsible for Kazaa (who would prefer to remain anonymous). I am also grateful to all those people who were willing to use some of their spare time participating in the experiments performed in this thesis. This paper would not have been possible without their help.

References

[1] James Walkerdine, Lee Melville and Ian Sommerville; "Dependability Properties of P2P Architectures; Computing Department, Lancaster University, UK

[2] Nelson Minar;"Harnessing the Power of Disruptive Technologies"; 1st Edition March 2001 ISBN: 0-596-00110-X

[3] Choon Hoong Ding, Sarana Nutanong, and Rajkumar Buyya; "P2P Networks for Content Sharing"; Grid Computing and Distributed Systems Laboratory,Department of Computer Science and Software Engineering,The University of Melbourne, Australia

[4] Vasileios Vlachos, Stephanos Androutsellis-Theotokis, Diomidis Spinellis; "Security Applications of Peer-to-Peer Networks"; Department of Management Science and Technology, Athens University of Economics and Business

[5] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron and Dan S. Wallach; "Secure Routing for Structured Peer-to-Peer Overlay Networks"; Microsoft Research Ltd. and Rice University

[6] Constance M. Johnson, B.S.N.1,2, Todd Johnson, Ph.D.2, Jiajie Zhang, Ph.D.2; "Increasing Productivity and Reducing Errors through Usability Analysis: A Case Study and Recommendations"; The University of Texas

[7] Steve McConnell; "Software Quality at Top Speed"; Software Development, August 1996

[8] Demetrios Zeinalipour-Yazti; "Exploiting the Security Weaknesses of the Gnutella Protocol"; Department of Computer Science, University of California

[9] Stefan Saroiu, P. Krishna Gummadi, Steven D. Gribble; "A Measurement Study of Peer-to-Peer File Sharing Systems"; Dept. of Computer Science and Engineering, Univ. of Washington

[10] Terje Mjømen; "Spyware and identity theft"; College of Gjøvik 2005

[11] John Borland; "Gnutella viruses weaker than email bugs, experts say", news.com article, june 2000, http://www.news.com/2100-1023-241440.html?legacy=cnet] (Visited Feb 2005)

[12] Nist sp800-55; http://csrc.nist.gov/publications/nistpubs/index.html

[13] Edward G. Carmines and Richard A. Zeller; "Reliability And Validity Assessment"; Sage University Paper 1979; ISBN:0-8039-1371-0

[14] DeMarco and Lister; "Peopleware: Productive Projects and Teams"; Dorset House Publishing; ISBN: 0-932633-43-9

[15] Butler W. Lampson; ”Hints for Computer System Design”; Computer Science Laboratory,Xerox Palo Alto Research Center

[16] Stefan Saroiu, Steven D. Gribble, and Henry M. Levy; “Measurement and Analysis of Spyware in a University Environment”; Department of Computer Science & Engineering University of Washington

[17] AL BERG; “P2P, OR NOT P2P?”; Information Security, February 2001; http://infosecuritymag.techtarget.com/articles/february01/cover.shtml (Visited Feb 2005)

[18] OECD Information Technology Outlook 2004; “Peer to Peer Networks in OECD Countries”; Pre-release of Section from Chapter 5 of the Information Technology Outlook

[19] Andrew Odlyzko; “Economics, Psychology, and Sociology of Security”; Digital Security Center, University of Minesota

[20] Rayford B. Vaughn, Jr., Ronda Henning and Ambareen Siraj; ”Information Assurance Measures and Metrics – State of Practice and Proposed Taxonomy”;

[21] Edited by Andy Orham; “Peer-To-Peer: Harnessing the Power of Disruptive Technologies”; O’Reilly & Assosiates; ISBN: 0-596-00110-X; Published 2001

[22] Napster; http://ntrg.cs.tcd.ie/undergrad/4ba2.02-03/p4.html (Visited Jan 2005)

[23] Krishna Kant and Vijay Tewary; ”On the Potential of Peer-to-Peer Computing”; Intel Corporation.

[24] Dan S. Wallach; “A Survey of Peer-to-Peer Security Issues”; Rice University, Houston, TX 77005, USA

[25] S. Ratnasamy et al.; “A Scalable Content Addressable Network”; In: Proc. ACM SIGCOMM’01, San Diego, California; 2001

[26] I. Stoica et al.; “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications”; In: Proc. ACM SIGCOMM’01, San Diego, California; 2001

[27] A. Rowstron and P. Druschel; “Pastry: Scalable, Distributed Object Location and Routing for Large Scale Peer-to-Peer Systems”; In: Proc. IFIP/ACM Middleware; 2001, Heidelberg, Germany

[28] B. Y. Zhao et al.; ”Tapestry: An Infrastructure for Fault-Resilient Wide-Area Location and Routing”; Technical Report UCB//CSD-01-1141, U. C. Berkeley;2001

[29] Nathaniel S. Good and Aaron Krekelberg; “Usability and Privacy: A Study of Kazaa P2P File-Sharing”; Information Dynamics Lab HP Laboratories and Office of Information Technology, University of Minnesota

[30] United States House of Representatives Committee on Government Reform; Staff Report Prepared for Rep. Tom Davis and Rep. Henry A. Waxman; "File-Sharing Programs and Peer-to-Peer Networks: Privacy and Security Risks"; May 2003; Http://www.reform.house.gov

[31] ITsecurity.com, December 2004 http://www.itsecurity.com/tecsnews/dec2004/dec200.htm (Visited march 2005)

[31] H. Damker; "Sicherheitsaspekte von P2P-Anwendungen in Unternehmen"; In: "Peer-to-Peer – Ökonomische, Technologische und Juridische Perspektiven (pp.209-228)"; Berlin:Springer.

[32] C. Shirky et al.; "P2P Networking Overview" O'Reilly & Associates; 2001; ISBN: 0-596-00185-1

[33] P2P Working Group; http:www.peer-to-peerwg.org (Visited march 2005)

[34] Dimitri DeFigueiredo et al.; "Analysis of Peer-to-Peer Network Security Using Gnutella"; Department of Computer Science, University of California at Davis

[35] Ben Edelman; "Comparison of Unwanted Software Installed by P2P Programs"; 2005; http://www.benedelman.org/spyware/p2p/ (Visited Feb 2005)

[36] Prepared Statement of The Federal Trade Commission Before the Subcommittee on Competition, Infrastructure, and Foreign Commerce of the Committee on Commerce, Science, and Transportation; "United States Senate Hearing on P2P File-Sharing Technology"; Washington, D.C; June 23, 2004; www.ftc.gov/os/2004/06/040623p2ptest.pdf

[37] Internal Memo from Sharman networks; www.zdnet.com.au/news/business/0,39023166,39179933,00.htm (Visited January 2005)

[38] Download statistics from Download.com; www.download.com (Visited February 2005)

[39] Kazaa online documentation; www.kazaa.com (Visited February 2005)

[40] Limewire online documentation; www.limewire.com (Visited February 2005)

[41] Grokster online documentation; www.grokster.com (Visited February 2005)

[42] Shareaza online documentation; www.shareaza.com
(Visited February 2005)

[43] A. Whitten and J.D. Tygar; "Why Jonny Can't Encrypt"; Proceedings of the 8th USENIX Security Symposium; August 1999

[44] CDT Policy Post; Volume 9, Number 11; May 19, 2003;
www.cdt.org/publications/pp_9.11.shtml

[45] Butler W. Lampson;" Hints for Computer System Design"; Computer Science Laboratory, Xerox Palo Alto Research Center

[46] Eric Howe; "The spyware warrior guide to anti-spyware testing";
http://spywarewarrior.com/asw-test-guide.htm
(Visited March 2005)

[47] Constance M. Johnson, B.S.N.1,2, Todd Johnson, Ph.D.2, Jiajie Zhang, Ph.D.2; "Increasing Productivity and Reducing Errors through Usability Analysis: A Case Study and Recommendations"; The University of Texas

[48] Grant Gross; "Peer-to-Peer Peering Pondered: Do file-sharing services get to snoopy?"; www.pcworld.com/news/article/0,aid,110861,00.asp ; May 2003.
(Visited February 2005)

[50] K. Vanwasi; "Unleashing the Power of P2P; Network Magazine India;
www.networkmagazineindia.com/200204/200204focus3.shtml
(Visited February 2005)

[51] New version of Napster; www.napster.com
(Visited April 2005)

[52] MSN Messenger; http://messenger.msn.no/
(Visited February 2005)

[53] Trillian; www.ceruleanstudios.com/
(Visited April 2005)

[54] ICQ; www.icq.com/
(Visited April 2005)

[55] SETI@home; http://setiathome.ssl.berkeley.edu/
(Visited February 2005)

[56] BitComet; www.bitcomet.com/
(Visited April 2005)

[57] BearShare; www.bearshare.com/
(Visited February 2005)

[58] Edonkey; www.edonkey2000.com/

(Visited February 2005)

[59] Direct Connect; www.neo-modus.com/
(Visited February 2005)

[60] Bram Cohen; "Incentives Build Robustness in BitTorrent"; 2003

[61]Blizzard.com;"Blizzard Has a Problem";
http://www.worldofwar.net/articles/bittorrent-editorial.php
(Visited May 2005)

[62] Lingling Sun, Yamini Upadrashta, Julita Vassileva; "Ensuring Quality of Service in P2P File Sharing through User and Relationship Modelling"; MADMUC Lab, Computer Science Department, University of Saskatchewan

[63] Lauri Pohjanheimo, Heikki Ailisto, Johan Plomp; "User Experiment with Physical Pointing For Accessing Services with a Mobile Device"; VTT Electronics, Finland

[64] Michael Miller; "Absolute PC Securiy & Privacy"; Sybex Inc 2002;
ISBN: 0-7821-4127-7

[65] Bruce Hughes; "WildTrends 2003: A Look at Virus Trends in 2003 and a Few Predictions for 2004"; TrueSecure ICSA Labs; 2003;
www.trusecure.com/company/press/pr_20031229.shtml

[66] Jack Ganssle; "As Good As It Gets"; Embedded.com; 2001;
www.embedded.com/showArticle.jhtml?articleID=9900455

[67] Microsoft; "Avalanche: Improving P2P File Swarming Through Network Coding"; 2005; http://research.microsoft.com/~pablo/avalanche.htm

[68] Christos Gkantsidis, Pablo Rodriguez Rodriguez; "Network Coding for Large Scale Content Distribution"; IEEE Infocom, 2005;
http://www.research.microsoft.com/~pablo/papers/nc_contentdist.pdf

[69] Tom's Hardware; "Licensed P2P attracts support of major record labels";
http://www.tomshardware.com/hardnews/20050624_183338.html
(Visited May 2005)

Appendix A – Shared Files

List of files to be shared in Metric M4

Test person ID:	
Date and Time:	

The following files or folders are to be shared:

1. c:\documents and settings\all users\shared documents\shared video\videotest1.mpg
2. c:\documents and settings\all users\shared documents\shared pictures\
3. c:\documents and settings\administrator\my documents\nonsensitivedoc.doc
4. c:\program files\winamp\pxsdkpls.dll AND winamp.bm
5. c:\documents and settings\administrator\my documents\MYSQL.doc
6. c:\program files\MSN gaming zone\windows\zcorem.dll
7. Share the file "video12.mpg", located in the folder "my documents" on the desktop
8. Share the file myholdiday.jpg, located on the desktop
9. Share the subfolder "nonsensitive" located in the folder "my documents" on the desktop
10. Share all files except "illegal.mp3" in the folder "NonCopyrightedMP3", located on the desktop

If there were any files that you were unable to share, write their number and the reason you were unable to share them here:

Duration of test:	
Number of files shared:	
Number of files accidentally shared:	
Description of files accidentally shared:	

Appendix B – Downloaded Files

Files downloaded during the measurements of M3

<u>Kazaa 28.02.05 - 04.03.05</u>:

2.3 million users,

52948992 GB shared in the network, distributed over 793 million files

<u>Session 1:</u> Duration: 60 minutes Start time: 28.02.05 at 15:30 Downloaded files: 15:52 “Admin Password.doc” 15:56 “Wife.jpg” 16:09 “Password.doc” 16:27 “Password.doc”	<u>Session 6:</u> Duration: 60 minutes Start time: 03.02.05 at 12:00 Downloaded files: 12:14 “Visa creditcard + pincode.txt” 12:35 “Bedroom.mpg”
<u>Session 2:</u> Duration: 60 minutes Start time: 01.03.05 at 13:00 Downloaded files: 13:26 “Budget.xls” 13:03 “Mastercard + Pincode.doc”	<u>Session 7:</u> Duration: 60 minutes Start time: 03.03.05 at 13:00 Downloaded files: 13:47 “Password.jpg”
<u>Session 3:</u> Duration: 60 minutes Start time: 02.03.05 at 15:00 Downloaded files: 15:48 “Admin Password.doc” 15:56 “Homemovie.mpg” 15:56 “3 Sensitive.jpg”	<u>Session 8:</u> Duration: 60 minutes Start time: 03.03.05 at 14:00 Downloaded files: No Download

Session 4: Duration: 60 minutes Start time: 02.03.05 at 16:00 Downloaded files: 16:24 “Classified.jpg”	Session 9: Duration: 60 minutes Start time: 04.03.05 at 18:00 Downloaded files: 18:07 “private photo.jpg”
Session 5: Duration: 60 minutes Start time: 02.03.05 at 17:00 Downloaded files: No download	Session 10: Duration: 60 minutes Start time: 04.03.05 at 19:00 Downloaded files: 19:48 “Inbox.pst” 19:48 “Budget.xls” 19:49 “Admin Password.txt”

Limewire 3.03.05 – 06.03.05:

965000 users,

119 GB shared in the network, distributed over 47230 files

Session 1: Duration: 60 minutes Start time: 3.03.05 at 18:30 Downloaded files: None	Session 6: Duration: 60 minutes Start time: 05.02.05 at 12:00 Downloaded files: No Download

Session 2: Duration: 60 minutes Start time: 03.03.05 at 19:30 Downloaded files: None	Session 7: Duration: 60 minutes Start time: 05.03.05 at 13:00 Downloaded files: No Download
Session 3: Duration: 60 minutes Start time: 04.03.05 at 15:00 Downloaded files: None	Session 8: Duration: 60 minutes Start time: 05.03.05 at 14:00 Downloaded files: No Download
Session 4: Duration: 60 minutes Start time: 04.03.05 at 16:00 Downloaded files: 16:55 “Kid.jpg”	Session 9: Duration: 60 minutes Start time: 06.03.05 at 18:00 Downloaded files: 18:39 “Shower.mpg”
Session 5: Duration: 60 minutes Start time: 04.03.05 at 17:00 Downloaded files: No Download	Session 10: Duration: 60 minutes Start time: 06.03.05 at 19:00 Downloaded files: 19:12 “private.jpg”

Grokster 5.03.05 – 07.03.05:

2519151 users,

60853248 GB shared in the network, distributed over 787287542 files

Session 1: Duration: 60 minutes Start time: 05.03.05 at 12:30 Downloaded files: 13:23 Credit Card.jpg 13:23 Credit card data.xls 13:24 Visa creditcard + pincode.txt	Session 6: Duration: 60 minutes Start time: 07.03.05 at 12:00 Downloaded files: No Download
Session 2: Duration: 60 minutes Start time: 05.03.05 at 13:00 Downloaded files: No Download	Session 7: Duration: 60 minutes Start time: 07.03.05 at 13:00 Downloaded files: 13:19 Inbox.pst 13:50 Admin Password.txt
Session 3: Duration: 60 minutes Start time: 06.03.05 at 18:00 Downloaded files: 18:47 Windows 2K server password.txt 18:48 11 Password.jpg 18:58 Credit card data.xls	Session 8: Duration: 60 minutes Start time: 07.03.05 at 14:00 Downloaded files: 14:23 private.jpg

Session 4: Duration: 60 minutes Start time: 06.03.05 at 19:00 Downloaded files: No Download	Session 9: Duration: 60 minutes Start time: 07.03.05 at 15:00 Downloaded files: 15:12 classified.jpg 15:13 sensitive.jpg 15:13 password.jpg
Session 5: Duration: 60 minutes Start time: 06.03.05 at 20:00 Downloaded files: 20:31 Credit card data.xls	Session 10: Duration: 60 minutes Start time: 07.03.05 at 16:00 Downloaded files: 16:44 home budget.xls 16:49 budget.xls

Shareaza 10.03.05 – 11.03.05:

1428908 of users in the network

Session 1: Duration: 60 minutes Start time: 10.03.05 at 12:00 Downloaded files: 12:52 Visa creditcard + pincode.txt	Session 6: Duration: 60 minutes Start time: 10.03.05 at 17:00 Downloaded files: No Download
Session 2: Duration: 60 minutes Start time: 10.03.05 at 13:00 Downloaded files: 13:34 inbox.pst 13:58 Budget.xls	Session 7: Duration: 60 minutes Start time: 11.03.05 at 13:00 Downloaded files: No download
Session 3: Duration: 60 minutes Start time: 10.03.05 at 14:00 Downloaded files: No Download	Session 8: Duration: 60 minutes Start time: 11.03.05 at 14:00 Downloaded files: No Download
Session 4: Duration: 60 minutes Start time: 10.03.05 at 15:00 Downloaded files: No Download	Session 9: Duration: 60 minutes Start time: 11.03.05 at 15:00 Downloaded files: 15:47 Credit card data.xls

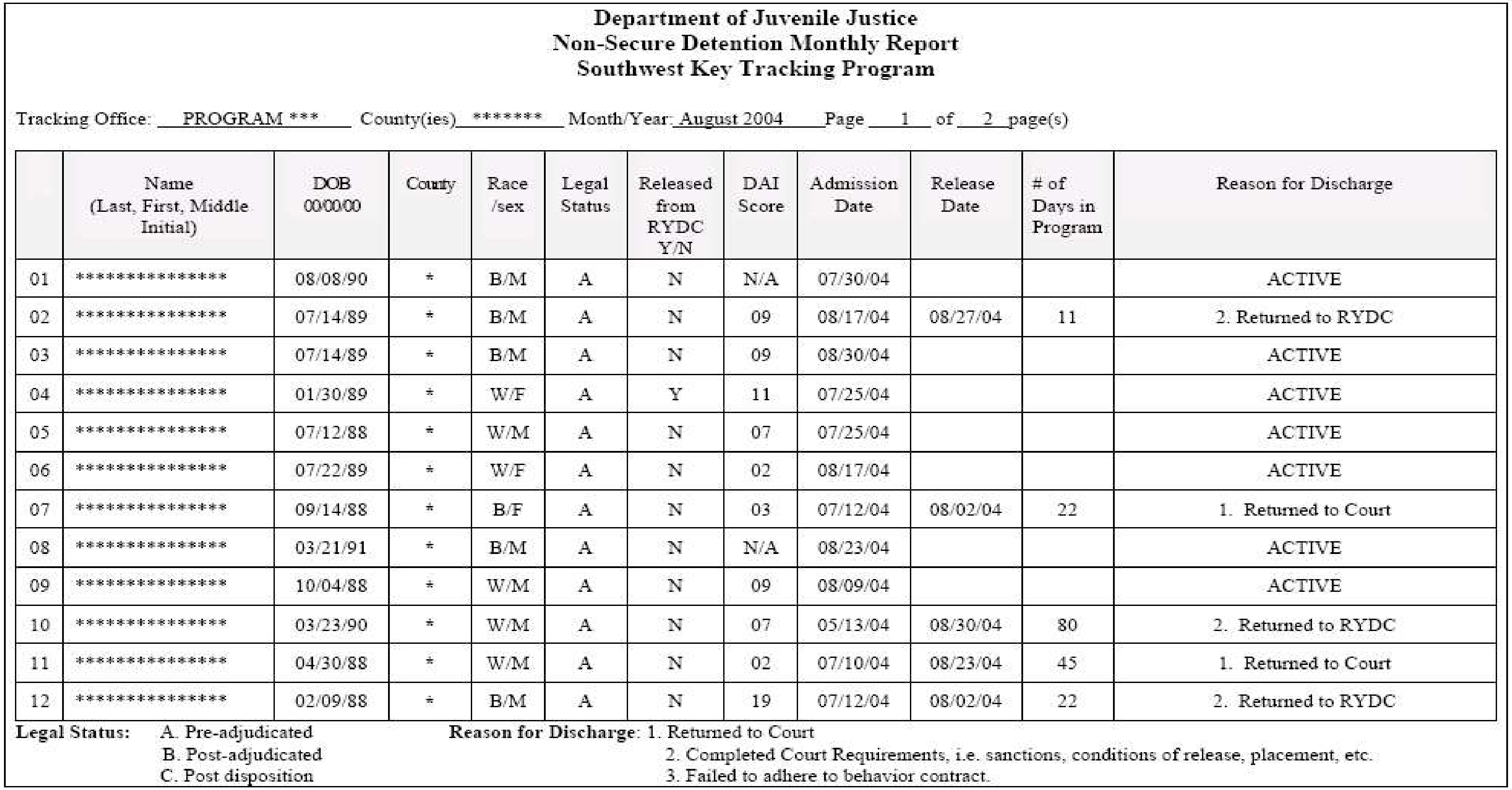
Session 5: Duration: 60 minutes Start time: 10.03.05 at 16:00 Downloaded files: 16:51 Mastercard + pincode.txt 16:52Visa creditcard + pincode.txt	Session 10: Duration: 60 minutes Start time: 11.03.05 at 16:00 Downloaded files: No Download

Appendix C – False Sensitive Data

List of false sensitive files shared in M3:

- Private (folder)
    - Inbox (folder)
        - “Budget.xls”
        - “Home Budget.xls”
        - “Credit card data.xls”
        - “Inbox.pst”
    - Private photos (folder)
        - “private.jpg”
        - “private photo.jpg”
        - “sensitive.jpg”
        - “credit card.jpg”
        - “wife.jpg”
        - “family.jpg”
        - “admin password.jpg”
        - “tax return.jpg”
        - “Microsoft money.jpg”
        - “Classified.jpg”
        - “Password.jpg”
        - “kid.jpg”
    - Private video (folder)
        - “Beach.mpg”
        - “Bedroom.mpg”
        - “Dancing.mpg”
        - “Hardlanding.mpg”
        - “Homemovie.mpg”
        - “Shower.mpg”
        - “Son.mpg”
    - “Admin Password.txt”
    - “Cisco Admin Pass.txt”
    - “Mastercard + pincode.txt”
    - “Visa creditcard + pincode.txt”
    - “Windows 2K server password.txt”

Appendix D - Examples of Shared Sensitive Data



Department of Juvenile Justice
Non-Secure Detention Monthly Report
Southwest Key Tracking Program

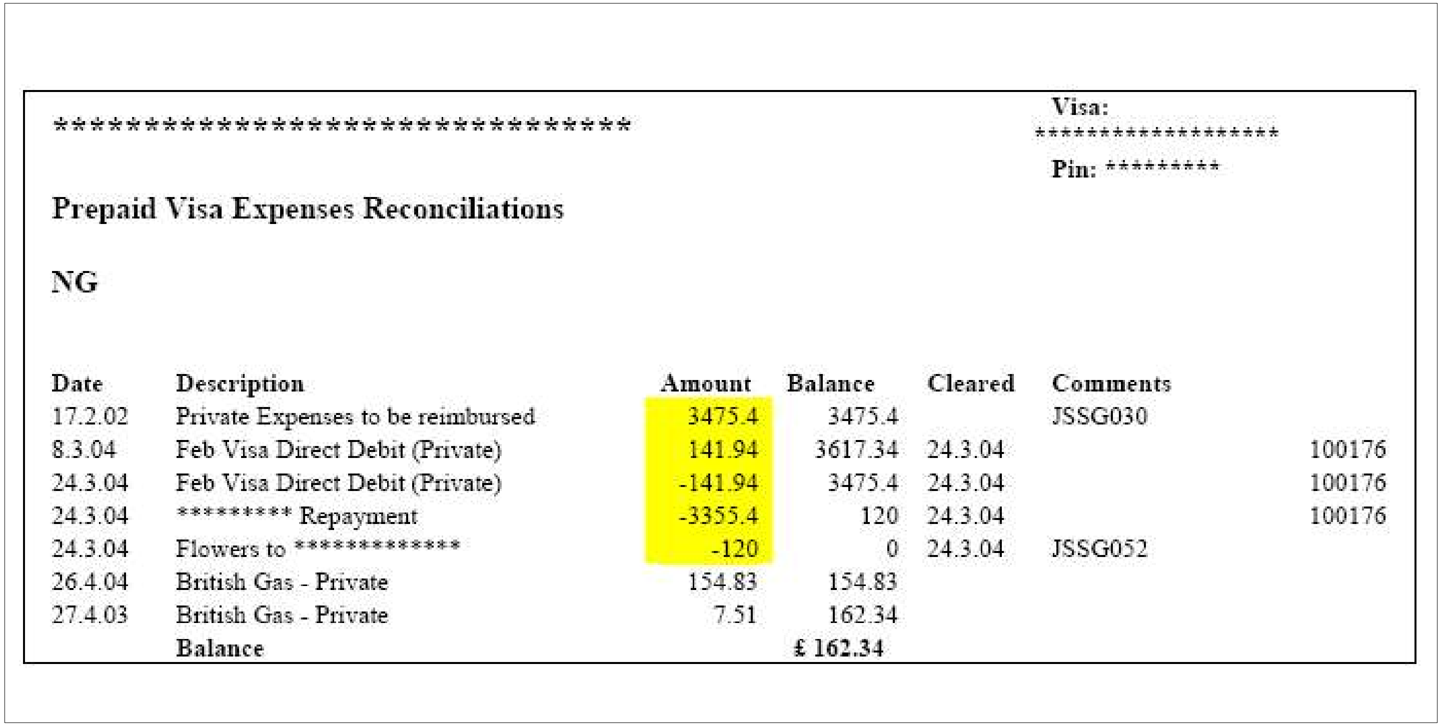
Tracking Office: PROGRAM *** County(ies) ******* Month/Year: August 2004 Page 1 of 2 page(s)

	Name (Last, First, Middle Initial)	DOB 00/00/00	County	Race /sex	Legal Status	Released from RYDC Y/N	DAI Score	Admission Date	Release Date	# of Days in Program	Reason for Discharge
01	***************	08/08/90	*	B/M	A	N	N/A	07/30/04			ACTIVE
02	***************	07/14/89	*	B/M	A	N	09	08/17/04	08/27/04	11	2. Returned to RYDC
03	***************	07/14/89	*	B/M	A	N	09	08/30/04			ACTIVE
04	***************	01/30/89	*	W/F	A	Y	11	07/25/04			ACTIVE
05	***************	07/12/88	*	W/M	A	N	07	07/25/04			ACTIVE
06	***************	07/22/89	*	W/F	A	N	02	08/17/04			ACTIVE
07	***************	09/14/88	*	B/F	A	N	03	07/12/04	08/02/04	22	1. Returned to Court
08	***************	03/21/91	*	B/M	A	N	N/A	08/23/04			ACTIVE
09	***************	10/04/88	*	W/M	A	N	09	08/09/04			ACTIVE
10	***************	03/23/90	*	W/M	A	N	07	05/13/04	08/30/04	80	2. Returned to RYDC
11	***************	04/30/88	*	W/M	A	N	02	07/10/04	08/23/04	45	1. Returned to Court
12	***************	02/09/88	*	B/M	A	N	19	07/12/04	08/02/04	22	2. Returned to RYDC

Legal Status: A. Pre-adjudicated
B. Post-adjudicated
C. Post disposition

Reason for Discharge: 1. Returned to Court
2. Completed Court Requirements, i.e. sanctions, conditions of release, placement, etc.
3. Failed to adhere to behavior contract.

The form above was found on Kazaa, it is one example (from many found) that shows what types of legal correspondence/documents users accidentally share on P2P networks.



****

Visa:
****

Pin: *********

Prepaid Visa Expenses Reconciliations

NG

Date	Description	Amount	Balance	Cleared	Comments	
17.2.02	Private Expenses to be reimbursed	3475.4	3475.4		JSSG030	
8.3.04	Feb Visa Direct Debit (Private)	141.94	3617.34	24.3.04		100176
24.3.04	Feb Visa Direct Debit (Private)	-141.94	3475.4	24.3.04		100176
24.3.04	********* Repayment	-3355.4	120	24.3.04		100176
24.3.04	Flowers to *************	-120	0	24.3.04	JSSG052	
26.4.04	British Gas - Private	154.83	154.83			
27.4.03	British Gas - Private	7.51	162.34			
	Balance		£ 162.34			

The above document was found on Kazaa and contained personal information, VISA card number and accompanying Pin-code. This was one of several such documents found.

August 30, 2002

Dear ******************:

It is with much regret that I submit my resignation as Associate Executive Director of the ******************. I have been employed at this wonderful branch for approximately 5 years and I feel that it is time to continue my professional journey of community development by gaining experience in new arenas and fields. I have prayed and sought counsel on this situation and I believe it is the best decision for the branch and myself. When I initially submitted my resignation in July, You gave me a good perspective and a vote of confidence for me to remain in my current position. I appreciated and valued the wisdom that you gave me, but my heart and spirit was still ready to move on to a new calling. I had to be true to the vision that was placed before me and prepare to transition down a new path.

I have been afforded many opportunities during my employment at the ******************. I matriculated from an $8.00 an hour Outreach Coordinator to a $40,000 Associate Executive Director. Your guidance, tutelage, and encouragement have been an inspiration to me to do great things. You believed in my talents and abilities when others my have had doubts. You have given me tasks that have challenged me to grow and to stretch from my comfort zone. Many times you have served as a counselor on issues on and off of the job. You have truly been a blessing to my family and me. It has been a pleasure with you; your honesty and integrity are to be commended. The staff team that we have had over the years has provided many powerful memories in my life. Working with them through many trials and tribulations as a team has strengthened me as a person. I realize that no matter the obstacle, a team that is together can overcome any situation.

The facility does work that would make God proud within the community. We assist families in all of our programs in a variety of ways. The impact that we make in the lives of our participants and members is tremendous. The facility is accessible to all who want to join any of our activities. The youth in our community have a positive place to learn new and exciting things in a safe and nurturing environment. I leave work every day knowing that I have contributed with a team to make a difference in the lives of members of our community. The job has given me more than I could have given to it. It has been with honor and great pride that I have served the ******************.

The above document was found on Grokster, it is an example of the many personal documents that people accidentally share.

NAME: ********************

ADDRESS: ********************

PHONE: ********************

D.O.B: 12/4/1968

QUALIFICATIONS

- •CERT 3 COMMUNITY SERVICES/ AGE CARE
- •CERT 2 ASSET MAINTENANCE (CLEANING OPERATIONS)
- •OCCUPATIONAL HEALTH & SAFETY
- •MANUAL HANDLING

EMPLOYMENT HISTORY

- •1997- 3 MONTHS ********************, Packing cucumbers seasonal work

- •1997-1998 SHOPPING CENTRE CLEANING, General cleaning

- •1998- 2001 ********************, Medical Orderly

- •2001- 2002 COMMUNITY AGE CARE WORK, Assisting the Age in their own home.

REFERENCES

********************, ******************** Care Program . Phone: *********

********************, ******************** Consultancies Manager. Phone:

The above document was found on Shareaza. Several similar documents were found containing phone numbers and personal information.

Appendix E – Sample Metric

Sample Template From NIST 800-55

Critical element	2.2 Does management ensure that corrective actions are effectively implemented?
Subordinate question	2.2.1 Is there an effective and timely process for reporting significant weakness and ensuring effective remedial action?
Metric	The average time elapsed between vulnerability or weakness discovered and implementation of corrective action.
Purpose	Measures the efficiency of closing significant system weaknesses to evaluate the existence, and the timeliness and effectiveness, of a process for implementing corrective actions.
Implementation Evidence	1. Do you have a tracking system for weakness discovery and remediation implementation? ? Yes ? No 2. How many system weaknesses were discovered within the reporting period (count all weaknesses that were opened and closed within the reporting period)? _____ 3. How many weaknesses discovered within the reporting period were closed in – 30 days ____ 60 days ____ 90 days ____ 180 days ____ 12 months ____ Remain open ____
Frequency	Quarterly, semiannually, annually

Formula	(Number of weaknesses x 30 + number of weaknesses x 60 + number of weaknesses x 90 + number of weaknesses x 180 x number of weaknesses x 365) (individual answers to Question 3)/Total number of weaknesses closed (Sum of all answers to question 3)
Data Source	Plan of Actions and Milestones (POA&M) tracking system
Indicators	A target time must be set for corrective action implementation. Results should be compared to this target. The trend for corrective action implementation/weakness closure should be towards shorter time frames, as management becomes more aware of applicable processes. Also, efficiencies are likely to be gained from the increasing experience of personnel and the institutionalization of a formal remedial action process. It should be noted that some corrective actions may require an extended period of time to implement.