# Security Applications of Peer-to-Peer Networks [*] [†]

Vasileios Vlachos, Stephanos Androutsellis-Theotokis, Diomidis Spinellis
Department of Management Science and Technology
Athens University of Economics and Business
Patission 76, GR-104 34, Athens, Greece

**Abstract**

Open networks are often insecure and provide an opportunity for viruses and DDOS activities to spread. To make such networks more resilient against these kind of threats, we propose the use of a peer-to-peer architecture whereby each peer is responsible for: (a) detecting whether a virus or worm is uncontrollably propagating through the network resulting in an epidemic; (b) automatically dispatching warnings and information to other peers of a security-focused group; and (c) taking specific precautions for protecting their host by automatically hardening their security measures during the epidemic. This can lead to auto-adaptive secure operating systems that automatically change the trust level of the services they provide. We demonstrate our approach through a prototype application based on the JXTA peer-to-peer infrastructure.

**Keywords** Peer-to-peer, Antivirus, Intrusion Detection, JXTA

## 1 Introduction

The rapid evolution of the Internet, coupled with the reduction in the cost of hardware, have brought forth very significant changes in the way personal computers are used. Nowadays, the penetration of the Internet is wide, at least in the developed world, and high percentage of connectivity is handled through broadband technologies such as DSL, cable modems, satellite links and even 3G mobile networks. Many companies have permanent connections to the Internet through leased lines and optical fibers, and many home users through the aforementioned broadband connections. If one also takes into account the significant development of wireless networking technologies (such as Wireless LAN, HyperLAN), the immediate result is an almost universal connection of

[*] *Computer Networks*, 45:195–205, 2004.

most users on a 24-hour basis. Although the potential benefits arising from these developments are various and important, so are the dangers that follow from the possibility of malicious abuse of this technology.

The proliferation of viruses and worms, as well as the installation of Trojan horses on a large number of computers aiming at Denial of Service (DoS) attacks against large servers, constitute one of the major current security problems. This is due to the extent to which critical infrastructures and operations such as hospitals, airports, power plants, aqueducts etc. are based on networked software-intensive systems. The measures taken for protection against such threats include [45] the use of firewalls, anti-virus software and intrusion detection systems(IDS). Considerable importance is also placed on the topology of the network being protected [43], as well as to its fault tolerance to ensure that its operation will continue even if a part of it is damaged.

A significant increase in the spread of viruses, worms and Trojan horses over the Internet has been observed in the recent years. Recent evidence shows that older boot sector viruses, as well as viruses transmitted over floppy disks no longer constitute a considerable threat [12]. At the same time, though, modern viruses have become more dangerous, employing complex mutation, stealth and polymorphism techniques [37] to avoid detection by anti-virus software and intrusion detection systems. These techniques are particularly advanced and, combined with the fact that antivirus software is often not properly updated with the latest virus definitions, can lead to uncontrollable situations.

In the last two years it has been proven both theoretically [38, 23] but mainly practically that the infection of hundreds of thousands of computers within a matter of hours — or even minutes — is feasible. At the theoretical level Staniford [38] presented scanning techniques (random scans, localized scans, hit-list scans, permutation scans) which, used by a worm, can perform attacks of this order. Indeed such worms are often referred to as *Warhol worms* or *Flash worms* due to their potential velocity of transmission.

A similar confirmation was obtained practically in the cases of the worms *Code Red* [31], *Code Red (CRv2)* [5], *Code Red II* [13], *Nimda* [26, 27, 20], and *Slammer* [22], which were characterized as epidemics by the scientific community [44] (although a more appropriate epidemiological term would be *pandemics*). Recently the Blaster-worm [24, 21] caused significant disruption in the Internet, although the infection rate of the specific worm was relatively slow in comparison with the previously mentioned worms. The reason for the effectiveness of the Blaster-worm was the exploitation of the Windows DCOM RPC interface buffer overrun vulnerability. This vulnerability affects all unpatched Windows NT /2000/ XP systems, as opposed to Code Red worms variations or the Slammer worm which were focused on machines acting as Web Servers or SQL Servers respectively.

All of the above is evidence that *rapid malcode* is extremely hard to confront using the "traditional" way of isolating and studying the code to extract the appropriate signature and update the IDS in real time.

We now propose to the reader to consider human behavior during a flu epidemic. Obviously a visit to a doctor and the use of vaccines is essential, however there is also need for an increased awareness and use of hygiene rules: avoiding crowded spaces, increasing the ventilation of our working area etc. Once the epidemic subsides, these

measures can be suspended; a person showing symptoms of the disease, of course, should still visit a doctor to receive medical care, regardless of whether the epidemic is still taking place.

The classic computer protection methods can be likened to the above medical situation: The vaccination of the population can be compared to updating the virus signature files; the lookout for symptoms may be compared to detection by an IDS; while the hygiene rules followed, which are essential for the protection of the larger, still unaffected population, may be compared to the operation of our proposed system, described in the following sections.

# 2 Architecture

Peer-to-peer networks, which we will hereafter reference as p2p networks, are often considered to be security threats for organizations, companies or plain users, mainly due to the use of p2p-based applications for illegal file sharing, and to the ability of worms to be spread through such applications (e.g. VBS.GWV.A [41, 40] and W32.Gnuman [10]). Our work indicates, however, that p2p networks can also be positively utilized to significantly reinforce network security, by offering substantial help in the protection against malicious applications. We propose an effective way to achieve this by collecting and exchanging information that will allow us to obtain a global overview of the network status, with reference to ongoing security attacks. The goal of our methodology is to select the most appropriate security policy, based on the level of danger posed by rapid malcode circulating in the network.

P2p networks leverage the principle that a much better utilization of resources (processing power, bandwidth, storage etc.) is achieved if the client/server model is replaced by a network of equivalent peers. Every node in such a p2p network is able to both request and offer services to other peer nodes, thus acting as a server and a client at the same time (hence the term "servent" = *SERV*er + cli*ENT* which is sometimes used).

The motivation behind basing applications on p2p architectures or infrastructures derives to a large extent from their adaptability to variable operating environments, i.e. their ability to function, scale and self-organize in the presence of a highly transient population of nodes (or computers/users), hardware failures and network outages, without the need for a central administrative server.

Our proposed application, which we call "NetBiotic", requires the cooperation of several computers within a common *peer group*, in which messages are exchanged describing the attacks received by each computer. It consists of two independent entities: a Notifier and a Handler. These entities act as independent daemons for UNIX systems, services for Windows NT/2000/XP or processes for Windows 9x/Me. From now on we will be referring to these entities as daemons for simplicity. Figure 1 illustrates the architecture of the proposed system within a group of cooperating peer computers.

The Notifier is a daemon responsible for monitoring the computer on which it runs and collecting any information relevant to probable security attacks. There is a plethora of different approaches to incorporate in the Notifier; for simplicity in our preliminary implementation we only monitor the log files of several security related applications,

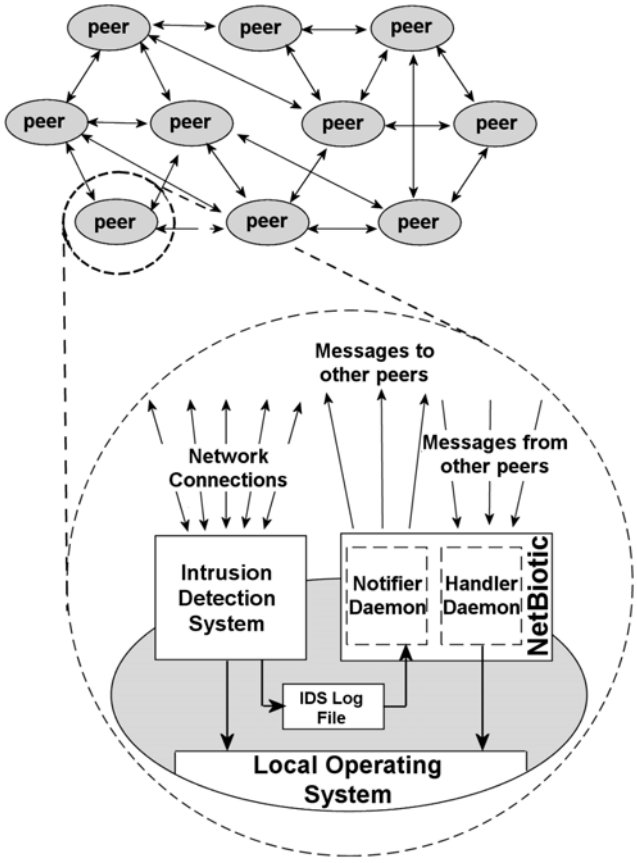


Figure 1: The architecture of the NetBiotic system within a group of cooperating peer computers.

such as firewalls, anti-virus software and IDS systems. These are applications that collect information about security threats and attacks to the computer system on which they are running and either notify the user of these attacks or take specific measures, while at the same time storing information relevant to the attacks into log files. By regularly reading the log files generated by these applications, the Notifier detects any recently identified security attacks to the computer it is running on. At regular time intervals $t$, the Notifier of node $n$ will record the number of hits ($h_t^n$) the node received over the past interval. It will then calculate and transmit the percentage $p_t^n$ by which this average differs from the average hits in an aggregate of the $k$ latest intervals, given by

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

where:

- $t$ is the ordinal number of a fixed time interval.
- $n$ is a node identifier.
- $h_t^n$ is the number of attacks node $n$ received in the interval $t$.
- $p_t^n$ is the percentage increase or decrease in attacks during the current interval $t$ on node $n$.
- $k(> 0)$ is the size of the "window" used, in number of $t$ time intervals, within which the average attack rate is calculated.

Selecting the appropriate length of the time interval $t$ is currently a subject of further research. In our current implementation we use a value of 15 minutes, which we feel provides a balance between increased network traffic and delay in notifying the network of attacks. This will be further discussed in the next Section.

A value of $p_t^n$ significantly greater than $1.0$ is an indication that node $n$ is under security attack during the interval $t$. The actual threshold used for $p_t^n$ is set by experience, and can vary according to the tolerance for false positives/negatives one has. With a small threshold it is possible to falsely interpret slightly increased rapid malcode activity as an epidemic (false positive), leading to an unnecessary activation of the available countermeasures, which in turn can disrupt some non critical useful services and cause inconvenience to the users. A very large threshold on the other hand, would probably fail to identify a rapid malcode epidemic (false negative) leaving the system protected only by its standard built-in security mechanisms. We tend to believe that is much better to tune the NetBiotic system towards a large threshold because rapid malcode epidemics cause a number of side-effects which are difficult to remain unnoticed. For us it is more important to ensure the timely recognition of these symptoms, in order to increase the security level of the protected system before a circulating worm may manage to launch an attack against it.

The Handler is also a daemon, responsible for receiving the messages sent from the Notifiers of other computers, and for taking the appropriate measures when it is deemed necessary. More specifically, it records the hit rates $h_t$ and percentage changes $p_t$ received from the different nodes in the peer group within a predefined period of time $t$, and calculates the overall change in attack rate, averaged for all $n$ nodes of the peer group that transmitted a message during the last interval:

$$p_{avg} = \frac{\left(\sum_{i=1}^{n} p_t^i\right)}{n}$$

The architecture supports countermeasures based upon predefined thresholds for $p_{avg}$, which are again set by experience. If $p_{avg}$ exceeds an upper threshold, the security level of the computer is raised. If, on the other hand, it drops below a lower threshold for a large period of time, the security level at which the computer functions is reduced.

Selecting the appropriate thresholds $\tau_{high}$ and $\tau_{low}$ for increasing or decreasing the security levels is crucial. In our approach, the thresholds are selected empirically and we have:

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

We base our decision for modifying the security policy on the rate of change of attacks, rather than on the actual number of attacks, to normalize the inputs from all peers with respect to their regular susceptibility to attacks; a peer whose actual number of attacks during a monitored time interval has increased from 1000 to 1100 has only experienced a 10% change in the attack rate, while a peer whose number of attacks increased from 50 to 150 within the same interval has experienced a 200% change in the attack rate; still, they have both received 100 attacks more than usual.

As far as the actual utilization of our architecture for protecting the computer system is concerned, the countermeasures taken will depend on many factors. A simple personal computer will be requiring different protection strategy than the central server of a large company. The type of operating system is also an important factor. The proposed system is not suggested as a replacement for traditional protection software (anti-viruses, IDS, firewalls etc.). The aim of NetBiotic is to assemble an additional, overall picture of the network status and suggest the basic security measures to be taken in the event of an epidemic. The NetBiotic architecture might not be capable to protect against a specific attack, however it will engage the standard measures that in many cases are crucial (such as disabling HTML previewing in several mail clients, not allowing Active X controls in various web browsers, disabling macros in some office application etc.).

In our prototype design, the recommended measures for a simple personal computer running Microsoft Windows would be to increase the security level of the default mail client and web browser. It would be additionally helpful to alert the user of the increased threat, in order to minimize threats of automated social engineering attacks. Servers can similarly disable non-critical networked services (e.g. by modifying the inetd.conf file in the case of Linux/Unix based operating systems). Figure 2 illustrates the operation and interaction of the Notifier and Handler daemons.

# 3 Implementation

The prototype system we present here was developed using the JXTA protocol [15]. JXTA is a partially centralized p2p protocol implementation introduced in early 2001, designed for maximum peer autonomy and independence. It allows applications to be developed in any language, it is independent of operating system type and is not limited to the TCP/IP protocol for data transfer. This allows an application such as NetBiotic to be easily ported to various operating systems, which is crucial to its operation, as its effectiveness will depend on the size of the peer group that will adopt it. An additional benefit of JXTA is its availability under an open source software license agreement, similar to the Apache License [1].

Due to the nature of our application, security issues are of particular interest. Security provisions are usually incorporated in p2p architectures by means of various cryptographic mechanisms such as the information dispersal algorithm [30] or Shamir's

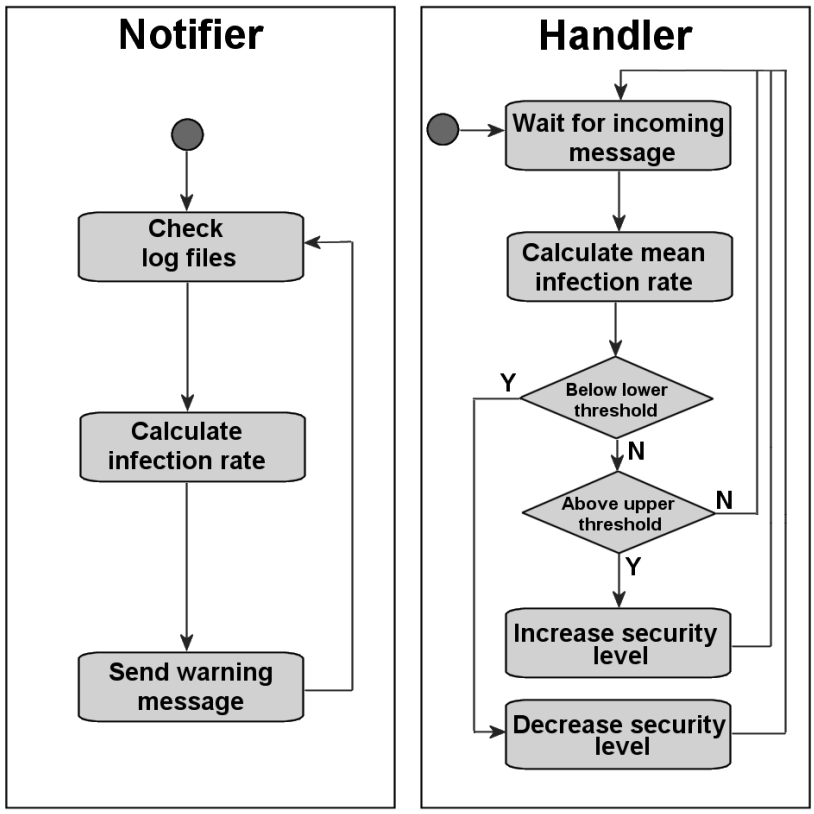


Figure 2: Operation of the Notifier and Handler daemons

secret sharing code [33], anonymous cryptographic relays [32], distributed steganographic file systems [11], erasure coding [19], SmartCards or various secure routing primitives [7].

JXTA peers function under a role-based trust model, whereby individual peers function under the authority of third-party peers to carry out specific tasks. Public key encryption of the messages exchanged, which may be in XML format, as well as the use of signed certificates are supported, providing confidentiality to the system. The use of *message digests* provides data integrity, while the use of *credentials* — special tokens that authenticate a peer's permission to send a message to a specific endpoint — provide authentication and authorization. JXTA also supports the use of secure pipes based on the TLS protocol. Further work is being carried out based on the security issues of the JXTA system, notably the implementation of a p2p based web of trust in the Poblano Project [4], which will be discussed in the future work Section.

Our system was implemented in Java (java2 version 1.4.0_02) using JXTA version 1.0, and uses the winreg [36] tool to administer the windows registry and modify the

security settings of the various applications. The main advantages of Java are its compatibility with most operating systems as well as the fact that it is one of the most secure programming languages.

In our preliminary implementation, the Handler modifies the security settings of the Microsoft Outlook mail client and the Microsoft Internet Explorer web browser. These two applications were selected as they are often the target of viruses. The simple operation of increasing their security settings is therefore enough to provide effective protection to a large number of users.

Most anti-virus programs can be adjusted to produce log files with the attacks they intercept. By regularly monitoring such log files, the Notifier daemon is able to detect a security attack and notify the peers. To test our prototype system, we created a software tool which randomly appends supposed security attack entries to these log files.

The NetBiotic architecture is compatible with any IDS or anti-virus software that can be setup to record the security attacks against the system it is protecting in a log file. Our aim is to make the NetBiotic system as independent as possible from the IDS with which it cooperates and the underlying operating system. This independence, however, cannot be total, as the following factors will be unavoidably system dependent:

- Log files
  In its simplest form, the system can simply check the size of the log file. For a more sophisticated operation, though, it would be necessary to incorporate a parser that would extract specific information from the log files. Such a parser has to be specific to each different type of log file used.

- Countermeasures taken
  System independence cannot be achieved in the case of the countermeasures taken, which will depend on the operating system. Different scripts have to be used to modify the security levels of applications in different operating systems.

Our system has been tested in laboratory environment as well as in a peer group that was set up for this purpose, in which virus attacks were simulated on some peers, resulting in the modification of the security settings of Microsoft Outlook and Internet Explorer on other peer computers. No real viruses were deployed. A program was running on each of the peer computers and periodically edited the log file of the antivirus software, simply changing its size to simulate a security attack event. The average frequency with which these events were simulated was random and different for each computer. The exchange of messages, individual and overall average hit rates as well as the resulting changes in the security settings of the application were recorded and verified against our theoretical expectations.

Finally, since our system consists of two independent daemons, it is possible to only install one of the two on certain peer computers. For instance, the Notifier daemon would be particularly useful running on a large company server, and supplying the peers with information about the security threats it faces. The administrators of such a server may prefer not to install the Handler daemon, and instead manually take action in the event of security attacks.

Similarly, for a personal computer user who may not have adequate security measures and antivirus software installed (for either financial or other reasons), installing

the Handler daemon itself may provide an adequate level of protection. In this case, the Handler daemon would modify the local security level based on information received by the security focused peer group. The Handler would therefore operate relying on the trustworthiness of the information received from the peer group only, which may in some cases be a disadvantage.

# 4 Related Work

The research that is most relevant to our proposed system has been carried out within the framework of project Indra [14], with which we partially share a common philosophy. We agree on the basic principle of using p2p technology to share security attack information between computers in a network in order to activate security countermeasures if necessary.

We differ however in the circumstances under which specific countermeasures should be taken. According to the Indra project team, in the event that a security attack is detected countermeasures should be immediately initiated, by using the appropriate plugins to protect the computer system. A single security attack anywhere in the network is enough for them to generate a response. In short, Indra is designed to respond to every single security attack.

In contrast, our system's goal is to determine if there is a general increase in the virus or worm attacks in the network, or more importantly a virus or worm epidemic outbreak. Measures taken in this case, such as the increase in security settings of mail clients, web browsers and anti-virus programs will only be effective during the epidemic, and the system will return to its original state after it is finished. In our design, individual virus or worm attacks in the network are not considered separately. Furthermore, we believe that our design can be expanded to very large network sizes without considerably increasing the overall network traffic.

A number of highly distributed systems rely on peer communications. The Hummingbird system [28] is based on a cooperative intrusion detection framework that relies on the exchange of security related information between networks or systems in the absence of central administration. The structure of the Hummingbird system is significantly more complex and advanced than NetBiotic, using a combination of Manager-Hosts, Managed Hosts, Slave Hosts as well as Peer, Friend and Symbiote relationships for the exchange of security related information. The Hummingbird system includes advanced visualization tools for its configuration and monitoring of log files, and although it may require considerable effort and expert knowledge for fine tuning the cooperation of each host with the others, it is particularly effective for distributed security attacks (such as doorknob, chaining, loopback attacks etc.). A potential secondary use of the Hummingbird system, in our view, could also be in the detection of malcode.

Emerald [29, 25] is a system targeted towards the exchange of security incident related information between different domains or large networks. It consists of a layered architecture that provides a certain abstraction, and requires the adjustment of parameters relevant to the trust relationships between cooperating parties. We believe that Emerald, like Hummingbird, can be invaluable in protecting a computer system or net-

work against distributed and targeted attacks. NetBiotic may not be in the position to affront such attacks with the same effectiveness, as its goal is the seamless and automated creation of a network of peers for the fast exchange of information regarding rapid spread malcode activity, leveraging the benefits of peer-to-peer architectures and topologies, and providing basic protection to the participating peers.

Bakos and Bert [2] presented a system for the detection of virus outbreaks. The fastest spreading worms use scanning techniques for identifying potential target computers. As a result, they also scan a large number of addresses that do not correspond to actual computers. The routers that intercept such scanning messages usually reply with a *ICMP Destination Unreachable* (also known as *ICMP Type 3* or *ICMP-T3*) message. The authors propose that a carbon copy message be sent by the routers to a central collector system, which will be responsible for collecting, correlating and analyzing this data. Bakos and Bert have implemented such a system by modifying the kernel of the Linux operating system to act as a router. The central collector receives the messages and forwards them to an analyzer system, which extracts the valuable information. It should however be examined whether the time required for the entire processing prohibits the use of this system for fast spreading worms, as described by Staniford [38].

Systems that use an extended network to gather information yet rely on a centralized client/server model were also examined. DeepSight [6] is a system developed by Symantec based on a client/server architecture, whereby centralized servers collect and re-distribute security attack information. Since it is a commercial system it is not available for scientific research, however it does include a very widespread data collection network.

An approach similar to DeepSight is taken by DShield, in which hundreds of computers communicate with central servers and transmit their IDS log files. The servers process the data and announce in a web site information about the currently active malware, the IP addresses from which most attacks originated and other useful information. Through the incorporation of different parsers, DShield supports various different IDS systems. DShield has been active for more than two years, with a significant number of users. A disadvantage of the system is that the large volume of data collected requires considerable processing time for extracting useful information. The theoretical times taken by the Flash and Warhol worms as well as the measured times for the Slammer worm [22, 38] to spread through the Internet are probably beyond the ability of DShield to react.

Both DeepSight and DShield aim at providing a global view of the Internet security status, however they are both subject to the disadvantages of the client/server architecture they follow: their dependence on a single server for their operation and their lack of adaptability makes them vulnerable to targeted attacks. An original approach taken by the AAFID [35], whereby agents are used to collect virus attack information also follows a centralized control structure. The same holds for the GrIDS system [39], which uses activity graphs to control large scale networks and identify suspicious activities, based on the judgment of a System Security Officer.

Finally, the following two approaches propose different ways of monitoring the overall security state and threat level of a network: In the DIDS system [34], the overall security state of a network under observation is represented by a numerical value ranging between 0 (safest) and 100 (least safe), while a clearly visual approach to rep-

resenting the network security state has been proposed [42, 8]. We find both approaches very descriptive and useful to a System Security Officer. In our prototype NetBiotic implementation, however, we are currently adopting a much simpler approach which consists of choosing between three different security states (regular, low risk and high risk), as described in Section 2.

# 5 Future Work

The NetBiotic system is an evolving research prototype. It is currently being extended in a number of ways as discussed below, in order to subsequently be released as open source software to allow the collaboration with other research groups working in similar directions.

At this stage, our goal is to propose an architecture, accompanied by a basic implementation for proof-of-concept purposes, which, based on a p2p network infrastructure can provide security services for computer systems. Although our prototype performed well in the situation in which we tested it, it is not suitable for performing large-scale testing.

We expect that, before more advanced versions of our application will be implemented, the scientific community will examine the use p2p networks in security applications from a theoretical standpoint and provide insight into the advantages and disadvantages of such an approach.

The following conceptual and implementation improvements are currently being considered:

- Vulnerability to malicious attacks
  A major drawback of our current design is its inability to effectively verify the information transmitted in the network. If one or more malicious users manage to introduce in the peer network a large number of false hit rate indications, the result may be the unwanted decrease of the security measures of the computers in the network, rending them vulnerable to virus attacks.

  We propose that all members of the security peer group will have to be authenticated and verified, probably through the use of certificates, to enforce a consistent authentication and authorization policy.

  At the implementation level, to confront the problem of malicious users introducing false information we further propose the following approaches, based on the capabilities offered by JXTA:

  1. JXTA supports the exchange of encrypted messages based on the TLS algorithm secured pipes [3], which will be used for the transmission of warning messages.
  2. JXTA *message digest* will be used for data integrity purposes.
  3. Other research groups are involved in the creation of a p2p-based web of trust. We intend to study these systems to examine to what extent they can be used to enhance the NetBiotic architecture.

- Use of epidemiological models
  We believe that the incorporation of mathematical epidemiological models for the detection of epidemic outbreaks in the network and determining the threshold for initiating security level modifications should significantly enhance the robustness of our system. A key point in our future research will be the selection of the thresholds for modifying security policies. These thresholds will be variable and will depend on each system's characteristics and on an analysis of the attack data collected.

  Studies [9, 18, 16, 17] show that there is a correlation between the patterns of spread of biological viruses and computer viruses. These studies were mainly limited to closed local area networks. P2p models are ideal for gathering large scale network virus information, which can subsequently be processed and adapted to epidemiological models, leading to decision tools for concluding, or perhaps even predicting, whether there is — or is likely to be — an epidemic outbreak in the network.

- Choice of appropriate security policy
  In conjunction with other factors, such as the role of the system being protected, our system should be able to effectively choose the most appropriate security policy for the specific period of time. In this way, single incidents of virus attacks may not be the cause of any concern, yet the detection of epidemic outbreaks would initiate a modification of the security policies.

- Platform porting
  In porting our system to Unix/Linux platforms, the operating system could be instructed to launch or halt applications, or automatically request updates. The configuration of these operating systems can be edited through plain text files, which is an additional benefit for our system.

# 6 Conclusions

Even the best protected organizations, companies or personal users are finding it difficult to effectively shield themselves against all malicious security attacks due the increasing rate with which they appear and spread.

Antivirus applications, as well as IDS systems, identify the unknown malware by employing behavioral based heuristic algorithms. These algorithms are particularly effective under a strict security policy, however they tend to produce an increased number of false alarms, often disrupting and upsetting the smooth operation of a computer system and the organization or users it supports. On the other hand, if the security policy is relaxed, the threat of a virus infection becomes imminent.

We propose a platform based on p2p technology in which the computers participating as peers of a network automatically notify each other of security threats they receive. Based on the rate of the warning messages received, our system will increase or decrease the security measures taken by the vulnerable applications running on the computer.

Our approach automates elements of the process of choosing the appropriate security policy, based on data useful for adjusting the security levels of both the operating system (by launching and terminating related applications) and the security applications (by modifying the security parameters of the heuristic algorithms they employ). An important aspect of our design is that the traffic introduced in the network by the peer nodes as a result of the transmission of hit rate information is minimal.

We believe that, with the inclusion of the future extensions we are currently working on, our approach may lead to operating systems, antivirus programs, IDS software and applications that will be able to self-adjust their security policies.

# References


[1] Apache license: Current on-line (June 2003): http://httpd.apache.org/docs/license.

[2] G. Bakos and V. Berk. Early detection of internet worm activity by metering icmp destination unreachable messages. In *Proceedings of the the SPIE Aerosense*, 2002.

[3] Wilson B.J. *JXTA*. New Riders, Indianapolis, IN, USA, June 2002.

[4] R. Chen and W. Yeager. Poblano: A distributed trust model for peer-to-peer networks. Technical report, Sun Microsystems.

[5] Code Red CRv2. Current on-line (June 2003): http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml.

[6] Deepsight threat management system: Current on-line (June 2003): http://www.securityfocus.org.

[7] P. Druschel and A. Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, May 2001.

[8] R. Erbacher, K. Walker, and D. Frincke. Intrusion and misuse detection in large-scale systems. *IEEE Computer Graphics and Applications*, 22(1), 2002.

[9] S. Forrest, S. Hofmeyr, and A. Somayaji. Computer immunology. *Communications of the ACM*, 40(10):88–96, 1997.

[10] W32.gnuman.worm: Current on-line (June 2003): http://service1.symantec.com/sarc/sarc.nsf/html/w32.gnuman.worm.html.

[11] S. Hand and T. Roscoe. Mnemosyne: Peer-to-peer steganographic storage. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, MIT Faculty Club, Cambridge, MA, USA, March 2002.

[12] Icsa labs 2002 computer virus prevalence survey. Current on-line (June 2003): http://www.trusecure.com/download/dispatch/vps2002.pdf.

[13] Code Red II. Current on-line (June 2003):
http://www.eeye.com/html/research/advisories/al20010804.html.

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

[15] Project jxta v2.0 java programmer's guide: Current on-line (June 2003): http://www.jxta.org/docs/jxtaprogguide_v2.pdf.

[16] J Kephart. How topology affects population dynamics. In *Proceedings of Artificial Life 3*, Santa Fe, New Mexico, June 1992.

[17] J. Kephart, D. Chess, and S. White. Computers and epidemiology. *IEEE Spectrum*, May 1993.

[18] J. Kephart and S. White. Directed-graph epidemiological models of computer viruses. In *Proceedgings of IEEE Computer Society Symposium on Research in Security and Privacy*, pages 343–361, Oakland, CA, 1991.

[19] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, S.R. Gummadi, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS*. ACM, November 2000.

[20] A. Mackie, J. Roculan, R. Russell, and VanVelzen M. Nimda worm analysis - incident analysis report version ii. September 2001.

[21] J. Miller, J. Gough, B. Konstanecki, J. Talbot, and J. Roculan. Deepsight threat management system threat alert - microsoft DCOM RPC worm alert.
Current on-line (August 2003):
https://tms.symantec.com/members/analystreports/030811-alert-dcomworm.pdf.

[22] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. The spread of the sapphire/slammer worm. Current on-line (June 2003):
http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html. Technical report, 2003.

[23] D. Moore, G. Voelker, and S. Savage. Internet quarantine:requirements for containing self-propagating code. In *Proceedings of the 2003 IEEE Infocom Conference, San Francisco California, USA*, April 2003.

[24] Microsoft security bulletin ms03-026. Current on-line (August 2003):
http://www.microsoft.com/technet/treeview/default.asp?
url=/technet/security/bulletin/ms03-026.asp.

[25] P. Neumann and P. Porras. Experience with EMERALD to date. In *First USENIX Workshop on Intrusion Detection and Network Monitoring*, pages 73–80, Santa Clara, California, April 1999.

[26] Current on-line (June 2003):
http://www.incidents.org/react/nimda.pdf.

[27] Current on-line (June 2003):
http://www.f-secure.com/v-descs/nimda.shtml.

[28] Polla, D., J. McConnell, T. Johnson, J. Marconi, D. Tobin, and D. Frincke. A framework for cooperative intrusion detection. In *Proceedings of the 21st National Information Systems Security Conference*, pages 361–373, October 1998.

[29] P. Porras and P. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *Proceedings of the National Information Systems Security Conference*, October 1997.

[30] M.O. Rabin. Efficient dispersal of information for security, load balancing and fault tolerance. *Journal of the ACM*, 36(2):335–348, April 1989.

[31] Code Red. Current on-line (June 2003):
http://www.eeye.com/html/research/advisories/al20010717.html.

[32] A. Serjantov. Anonymizing censorship resistant systems. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, MIT Faculty Club, Cambridge, MA, USA, March 2002.

[33] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, November 1979.

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

[35] E. Spafford and D. Zamboni. Intrusion detection using autonomous agents. *Computer Networks*, (34):547–570, October 2000.

[36] D. Spinellis. Outwit: Unix tool-based programming meets the windows world. In *Proceedings of the USENIX 2000 Technical Conference*, pages 149–158, San Diego, CA, USA, June 2000.

[37] D. Spinellis. Reliable identification of bounded-length viruses is np-complete. *IEEE Transactions on Information Theory*, 49(1):280–284, January 2003.

[38] S. Staniford, V. Paxson, and N. Weaver. How to own the internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, 2002.

[39] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS – A graph-based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, 1996.

[40] VBS.Gnutella. Current on-line (June 2003): http://service1.symantec.com/sarc/sarc.nsf/html/vbs.gnutella.html.

[41] VBS.Gnutella. Current on-line (June 2003): http://vil.nai.com/vil/content/v_98666.html.

[42] G. Vert, J. McConnell, and D. Frincke. A visual mathematical model for intrusion detection. In *Proceedings of the 21st National Information Systems Security Conference*, pages 329–337, October 1998.

[43] C. Wang, J.C. Knight, and M.C. Elder. On computer viral infection and the effect of immunization. In *Annual Computer Security Applications Conference (ACSAC)*, pages 246–256, December 2000.

[44] V. Yegneswaren, P. Barford, and J. Ullrich. Internet intrusions: Global characteristics and prevalence. In *Proceedings of ACM SIGMETRICS*, June 2003.

[45] R.L. Ziegler. *Linux Firewalls*. New Riders Publishing, Indianapolis IN, USA., 2002.