Global Information Assurance Certification Paper

Security Implications of "Peer-To-Peer" Software
GSEC Practical Assignment Version 1.4

By: Kelvin Choi, July 2002

In the halcyon days of dot-com hype (circa 2000), Peer-to-Peer computing joined the illustrious list all-important applications that would change the way people related to the universe. In a post-Napster world circa 2002, peer-to-peer computing no longer occasions overwrought jeremiads. This has not changed the rapid, often uncontrolled adoption of p2p applications by enterprise users. IDC predicts $133 million to be spent on Instant Messaging tools this year, and $1 billion by 2005.[1] This rapid integration of p2p type software into the enterprise has serious security implications that need to be addressed.

Peer-to-peer is a term that is used promiscuously. Loosely speaking, it most often refers to decentralized computing modes that do not rely on a traditional client-server model. In practice, the term encompasses instant messaging applications, corporate collaboration suites, file sharing applications, and non-traditional distributed computing. These diverse software types share a common ideology of eliminating or reducing the importance of any central node for data storage, processing power and the like. It is just this non-hierarchical, decentralized model that allows for both the advantages and dangers of p2p.

Hype aside, many of the promises of p2p technology are real. There is no denying that p2p applications can be useful in the corporate environment. They have the ability to streamline workflow, enhance communication, and reclaim idle hardware resources. PDI/DreamWorks realized real productivity gains by utilizing desktop/workstation processing power after business hours to render scenes from *Shrek*.[2] The convenience to a distributed workforce of instant messaging, (and peer based file sharing to a lesser extent) is undeniable. This immediate convenience is one of the main reasons for IM's rapid adoption, notwithstanding its ubiquity outside the enterprise:

---

[1] *Economist pp. 5-6*

[2] PDI/Dreamworks. It should be noted that properly designed security is like an Ogre, and "Ogres are like onions." They have layers.

> IM is rapidly transforming into an essential business communication tool—so much so that workers with IM accounts are routinely creating parallel, unmanaged communication universes to facilitate important business processes.[3]

Still, the current potential of distributed workspaces to enhance productivity, both within and between enterprises, remains largely untapped.

As an emerging technology, peer-to-peer computing presents serious security challenges for the corporate environment. The "parallel, unmanaged communication universe" is a prime example. While these challenges are not unique to peer-to peer computing, the rapid deployment of p2p, often uncontrolled by corporate IT, mandates serious deliberation regarding it's deployment and integration into the enterprise. This paper focuses primarily on instant messaging and file-sharing tools, as they are the most common, and the most likely to be installed outside the control mechanisms of corporate IT. Historical and theoretical vulnerabilities are discussed, as is general security strategy for dealing with some p2p application types.

Peer-To-Peer Application Types & Their Vulnerabilities

File-Sharing Tools

The file-sharing tool Napster remains the most infamous p2p application. Napster relied on central servers to index mp3 titles for users, so it wasn't strictly peer-to-peer. This relatively trivial cataloguing by Napster's central server also rendered it permanently and fatally vulnerable to the "Subpoena DoS" attack.[4] The open-source Gnutella has proven the popular successor, with BearShare, LimeWire, Morpheus, Gnewtellium, Newtella, Mutella, Phex, Qtella, and other front-end clients. Gnutella's true peer-to-peer architecture and distributed development model have shielded it from legal challenges to date.

---

[3] Gartner

[4] Bellovin (webcast)

File sharing applications such as this present multiple exposure opportunities for the enterprise. Issues of intellectual property are paramount. Companies bear some measure of liability for employees trading and storing copyrighted works in the office. Equally distressing is the opportunity for unintentionally sharing proprietary or delicate information through carelessly or improperly configured clients. Allowing documents to be shared without explicit permissions is an easy mistake for the unwary user, and users have been known to unintentionally share entire disc volumes. This "information leakage"[5] could be the most expensive security issue faced by the enterprise, as it has can have the greatest legal liability. This is exacerbated when employees install and configure file-sharing software outside a defined security process and infrastructure.

Vulnerabilities within file sharing tools can also expose data or allow for arbitrary code execution. Gnapster and Knapster versions prior to 1.3.9 would not check to see if a file was explicitly shared (Bugtraq ID 1186, http://www.securityfocus.com). This allows the attacker to view any file, and could be exploited via a relatively simple Perl script. Gnut, another of the myriad Gnutella clients, is vulnerable to cross-site scripting when an attacker shares a file with the script tags embedded in the file name.[6] Failure to disallow cross-site scripting is not unique to Gnut.

File-sharing applications are not immune to worms and Trojans either. VBS.Gnutella and W32.Gnuman.Worm are two of the most recent examples. W32.Gnuman.Worm was capable of changing its name to match a users query string.[7] Properly deployed anti-virus solutions should guard against known worms and viruses, but it is trivial for a malicious user to rename malicious payloads to spoof search results. This makes user education critical, just as with destructive email attachments.

The argument could be made that the Gnutella protocol is inherently insecure. Its very simplicity is the culprit. The packet structure:

---

[5] Berg
[6] See CVE CAN2001-1004, http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-1004
[7] http://service4.symantec.com/SARC/sarc.nsf/html/W32.Gnuman.Worm.html#technicaldetails

1. 16 byte MessageID: Essentially a Windows GUID, this should be unique to the network in question, where 2 recipients will not generate the same MessageID, and the source machine will not reuse a MessageID.
2. 1 byte FunctionID: Declares the purpose of the packet. There are few functions –
    a. Initialization (0x00)/ Initialization Response (0x1),
    b. Search (0x80)/ Search Response (0x81)
    c. Client-Push (0x40). No response, the data is the response.
3. 1 byte RemainingTTL: Decremented node-by-node, typical default is 5, but this can be changed.
4. 1 byte HopsTaken: This is incremented node by node.
5. 4 byte DataLength: Size of the data left in the packet.

The Initialization (ping) packet floods the network. Those nodes that choose send a pong that includes their IP address and amount of data shared. This amounts to a topology giveaway.[8] Since there is no provision for login or authentication, there is no way for a recipient to determine who originated a packet beyond assumptions based on HopsTaken. This raises the specter of Man-in-the-Middle attacks.

The search-response message will include the node's speed, file names matching the query, port, and its own MessageID. The querying node will then request the data, and transfer occurs over HTTP. This highlights the well-known difficulties of dealing with this type of traffic with a firewall, as port 80 would have to be shut down to stop data transfer. Most popular p2p software such as file-sharing and IM tools are, in fact, designed specifically to get around NAT and firewall blockage. Directly passing data through port 80, with or without HTTP tunneling, works so well that enterprise-level p2p applications also use this method. Groove (www.groove.net) can tunnel its proprietary protocol through port 80, while NXT3 (www.nextpage.com) is actually based on XML.[9]

[8] Bellovin
[9] Hurwicz

Nodes outside a firewall may very well be able to retrieve files from within the firewall by using a rendezvous server and the Client-Push trigger:

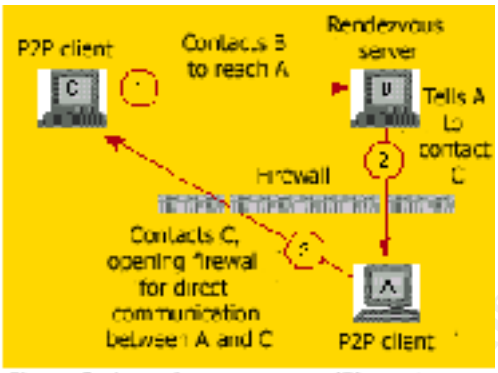

Figure 2. A rendezvous server (B) receives a request from a P2P client outside the firewall (C) to facilitate a connection with a P2P client inside the firewall (A). A initiates the conversation, which opens the firewall to direct traffic from C. [10]

Rendezvous servers could be blocked, but specific IP addresses must be known. The client inside the firewall must be able to receive traffic from the rendezvous nodes, but peers themselves can act as rendezvous servers, so a large pool of potential proxies will likely exist. This is helpful within the context of properly controlled and configured software, as it allows inter-network messaging and file sharing while maintaining some control over points of entries into and between disparate networks. Known, hardened rendezvous servers can be defined, with limited ports open to the outside world, through which messaging and data can be routed.

For unauthorized file-sharing applications however, there is an effective hole created in the perimeter defenses, making a destructive payload, worm, or man-in-middle attack more unconscionable. The facile subversion of NAT and firewall defenses represents the most worrisome security exposure from file-sharing software, partly because it is so difficult to combat, and because it in fact magnifies the danger of the application's other, more common vulnerabilities, allowing them into the enterprise.

[10] ibid.

Once subverted, the firewalled network is also more vulnerable to risks associated with the prolific spy ware now commonly found in file-sharing applications. One recent story highlights the comprehensive exposure it is possible to create. Brilliant Digital Entertainment had been installing its ad software within Kazaa for some time before it announced it had "plans to remotely 'turn on' people's PCs, welding them into a network of its own."[11] In fact, Brilliant Digital's Altnet Secureinstall can connect to p2p networks and ad servers outside of Kazaa, as well as auto-update. Much of the attention generated by this scheme has centered on privacy issues and the manner in which Altnet Secureinstall was distributed, but most worrisome is the comprehensively capable payload that was unwittingly downloaded and installed by Brilliant Digital and Kazaa's victims.

Instant Messaging

Instant messaging, since its popular acceptance with ICQ in 1996, has propagated virally across the Internet, reaching inside the corporate perimeter. The convenience of IM is undeniable through its delivery of instant communication. Like file-sharing tools, Instant Messaging tools come in multiple flavors, including the most popular freeware tools (AOL Instant Messenger, MSN Messenger, Yahoo IM, ICQ, etc.). They are also being built into file-sharing clients such as the Gnutella based Morpheus, and are in integral part of many corporate collaboration suites such as Groove, MS NetMeeting, and Lotus Sametime.

Instant messaging applications are, of course, vulnerable to worms and viruses. W32.Hello is the most notable example. W32.Hello was a relatively harmless example of IM worms, being easily detected by the presence of Hello.EXE in the Windows startup folder. As with file-sharing software, proper deployment and maintenance of anti-virus solutions will ameliorate this exposure. Of more concern are existing vulnerabilities that arise from flaws in either the architecture of the product or from coding flaws.

[11] Borland

In 2001, AOL had up to 80% of the IM market.[12] A survey of the software vulnerabilities and authentication and encryption found in popular freeware like AIM and MSN Messenger suggests that IM will be relatively insecure barring the replacement of freeware by more controllable and securable, enterprise-level tools. Below are some of the recent CVE candidates for AOL Instant Messenger (http://www.cve.mitre.org):

CVE-2000-1000	Format string vulnerability in AOL Instant Messenger (AIM) 4.1.2010 allows remote attackers to cause a denial of service and possibly execute arbitrary commands by transferring a file whose name includes format characters.
CVE-2000-1094	Buffer overflow in AOL Instant Messenger (AIM) before 4.3.2229 allows remote attackers to execute arbitrary commands via a "buddyicon" command with a long "src" argument.
CVE-2002-0005	Buffer overflow in AOL Instant Messenger (AIM) 4.7.2480, 4.8.2616, and other versions allows remote attackers to execute arbitrary code via a long argument in a game request (AddGame).
CAN-2000-0190	** CANDIDATE (under review) ** AOL Instant Messenger (AIM) client allows remote attackers to cause a denial of service via a message with a malformed ASCII value.
CAN-2000-0383	** CANDIDATE (under review) ** The file transfer component of AOL Instant Messenger (AIM) reveals the physical path of the transferred file to the remote recipient.
CAN-2002-0362	** CANDIDATE (under review) ** Buffer overflow in AOL Instant Messenger (AIM) 4.2 and later allows remote attackers to execute arbitrary code via a long AddExternalApp request and a TLV type greater than 0x2711.
CAN-2002-0457	** CANDIDATE (under review) ** Cross-site scripting vulnerability in signgbook.php for BG GuestBook 1.0 allows remote attackers to execute arbitrary Javascript via encoded tags such as <, >, and & in fields such as (1) name, (2) email, (3) AIM screen name, (4) website, (5) location, or (6) message.
CAN-2002-0591	** CANDIDATE (under review) ** Directory traversal vulnerability in AOL Instant Messenger (AIM) 4.8 beta and earlier allows remote attackers to create arbitrary files and execute commands via a Direct Connection with an IMG tag with a SRC attribute that specifies the target filename.
CAN-2002-0472	** CANDIDATE (under review) ** MSN Messenger Service 3.6, and possibly other versions, uses weak authentication when exchanging messages between clients, which allows remote attackers to spoof messages from other users.
CAN-2002-0592	** CANDIDATE (under review) ** AOL Instant Messenger (AIM) allows remote attackers to steal files that are being transferred to other clients by connecting to port 4443 (Direct Connection) or port 5190 (file transfer) before the intended user.

[12] Schiller

Buffer overflows are expected, but note the wide array of other vulnerabilities. Denial of service attacks are possible, as are cross-site scripting attacks, and directory traversals with the concomitant ability to execute arbitrary code. Of greater impact for the corporate environment are vulnerabilities such as CAN-2002-0592, where man-in-the-middle attacks allow for the interception of messages and files through direct connection.

In fact, this highlights the greatest challenge for secure instant messaging. In the excellent Auerbach Analysis, "Security in electronic messaging systems," Duane E. Sharp points out that any message transfer agent is a potential point of attack.[13] As an example, MSN Messenger's infrastructure includes a Dispatch Server, Notification Server, and a Switchboard Server. Although communications will occur directly from node to node, the session is actually maintained by the Notification Server, and all messages will actually pass through the Switchboard Server. The Switchboard server also passes along all requests for services such as file transfer and voice chat, making it an especially attractive target.[14]

In fact, the MSN Messenger protocol has historically contained weak message identifiers, making it especially susceptible to message spoofing. According to the Microsoft 1.0 protocol, the transaction ID will be a simple numerical string between 0 and 4294967295.[15] This is not hashed or encrypted in any way, and the server will always send the same transaction ID to a client This can easily be sniffed. In fact, with each succesive iteration of client to server commands, the transaction ID is simply incremented, taking any guesswork out of spoofing these ID's once they are sniffed. In fact, the server treats the ID cavalierly, leaving it up to the client to ensure uniqueness for each transaction. It is unclear if this has been changed, as a new protocol document has not been forthcoming from Microsoft. Although there is some support for MD5 hashing, it is

[13] Sharp
[14] Mintz and Wilson
[15] Movva and Lai

unclear whether newer versions of this protocl have addressed this issue, as Microsoft has not posted an updated protocol document.

The potential for identity hijacking has historically plagued IM applications. In 1998, "Wumpus" posted an ICQ hack called "ICQ Hijack" which allowed a malicious agent to entirely take over an ICQ account. Shortly thereafter, Alan Cox posted "icqsniff" to Bugtraq, proving the ease of sniffing ICQ passwords off the network.[16] Architectural flaws make this type of connection hijacking or eavesdropping relatively attractive: Jack Schiller points out that both ends of a messaging connection must sign-off before the link is entirely severed.[17] This is very attractive to "marketers, crackers, or just snoopy people." Leaving an open connection where there are multiple points of intrusion into a messaging system represents clear exposure for the enterprise.

There is no robust encryption system with most implementations of freeware tools such as AIM or MSN Messenger. Of equal importance is non-repudiation of message delivery and confirmation of message integrity. Before IM can be trusted in critical and time-sensitive environments, such as financial vertical markets, it must provide some measure of CIA: confidentiality, integrity, and availability. Even when some form of authentication is included, as with MSN Messenger, message spoofing is possible due to a weak authentication implementation. This is clearly an unacceptable risk for time and content sensitive communications.

[16] Glave
[17] Schiller

Coders with a rebel ethos have historically been the developers of freeware file-sharing tools. Worse still, the general user of these applications has absorbed the Napster paradigm, where short shrift is given to the copyright holders of proprietary material. Attendant to this is the "us vs. the firewall administrator" attitude that has promoted the subversion of perimeter defenses. This mindset has been exacerbated by the litigious behavior of "content providers" such as the RIAA and their myriad attempts to direct the consumer towards a "pay-for-play" model. Whatever the cause of a content provider's legal agitation, hosting rogue file-sharing applications in the corporate environment is ill advised when they are used for trading non-work related materials, especially with entities outside the corporate perimeter.

It behooves the enterprise to engage in serious cost-benefits-needs analysis to determine the worth of file-sharing applications. For some industries, the ability to easily share data outside the corporate firewall (with suppliers, customers, etc.) is an efficiency boon. In these cases, enterprise level software integrating multiple peer-to-peer applications such as file sharing and IM is the best solution. This allows for the creation of trusted virtual networks spanning individual organizations. The substantial costs and political difficulties of involving multiple organizations within a circle of trusted partners demand a careful ROI study. Tools such as Windows Messenger is a less expensive alternative, but for it to be even marginally secure, SIP proxies must be deployed and traffic forced through them to leverage SSL.[18]

In order address the need for confidentiality, integrity, and availability, a robust method of authentication, encryption, and non-repudiation must be built in. Integration with public key infrastructure is the most obvious solution, often mentioned by p2p commentators. However, a "lack of immediate hard-dollar demand"[19] for PKI has hampered its widespread corporate adoption. The general

[18] Fout
[19] Hurwicz

integration of corporate collaborative tools such as Groove into the enterprise infrastructure of authentication, encryption, and perhaps validation (LDAP or Active Directory, existing PKI, and enterprise wide certificate authorities) could provide the most seamless solution. This is imminent for some vendors (Groove), while some standards give it marketing attention (JXTA).

In the meantime, authentication may need to take place outside an IM or file sharing application. Email, voice chat, or phone could suffice. With the exception of voice chat (why not just telephone?), all outside authentication methods make instant messaging less than "instant." A more robust and comprehensive solution is needed, and Communicator Inc's Hub Instant Messaging protocol application may point the way. Made to span organizations, Hub provides for a "gated community" enclosing diverse financial organizations.[20] Eschewing the use of aliases, everyone must sign in with a real name, providing some accountability. Defined nodes are the only entrance point to the IM network in this implementation, making this more of a closed system and decreasing the opportunity for man-in-the-middle attacks. All traffic is logged and stored, partially addressing the legal issues around non-repudiation.

This type of logging is vulnerable to information leakage and must be accounted for within a mature security policy. Sam Jain, CEO of eFront is the highest profile victim of promiscuous data leakage.[21] Logs revealing detailed discussions between Jain and others regarding eFront's employees, business partners, were posted on the web. The content was exceptionally problematic for eFront and Jain in particular, as this issue damaged or severed eFront's relationship with its customer's and partners. This highlights the need for any p2p application, whether installed and managed by the IT department or not, to be fully addressed within an organization's security policy. Mr. Jain may be wishing that his messages had conformed to some professional standard of appropriate discussion, and that his messaging logs had been properly secured!

---

[20] *Economist* pp. 5-6
[21] Festa

Any useful security policy must define appropriate use of managed software. It must also address the acceptability of non-managed applications such as AOL Instant Messenger or Gnutella within the organization. If managed collaboration tools are to exist, or if freeware messaging or trading programs are allowed, best security practices should be defined for security configuration as well as data and message handling. The consequences of installing inappropriate software, or trading inappropriate content must also be clearly defined. The response of the IT or infosec department must also be well defined.

User training will be essential to address improper data sharing or improper messaging practices. Aside from issues of configuration, a properly defined security policy will address levels of training appropriate to the p2p application. More generally, the security policy should state whether or not users have control over software installation. It may or may not be desirable to fully lock down workstations and servers for a subset of employees, but this is virtually the only way to ensure that unmanaged file sharing and IM applications are not invited into the enterprise.

An alternative solution is to actively monitor what applications are running throughout the enterprise and automatically kill them as necessary. As an example, NetIQ Corporation's Security Manager (http://www.netiq.com) is capable of poling running processes on a timed basis and killing either any application that matches or does not match a defined list of processes. An appropriate alert is notifying the security principles that a rogue application was terminated. Tthe software can be configured to send a message to the offender referring them to the appropriate section of the corporate security policy, a particularly effective trick. Because Security Manager integrates with some firewall solutions (Cisco PIX and Checkpoint Firewall One), it can also alert on and monitor when defined ports are in use and notify the firewall to shut these ports if necessary, whether incoming or outgoing. However, given the fact that many p2p applications are flexible with their port usage, or pass data via HTTP, stopping the application at the host level is a more effective solution.

Conclusion

Peer-to-peer applications take diverse forms. The most common p2p applications such as Gnutella clients or AIM often have a development legacy that either intentionally subverts corporate perimeter defenses, or gives short shrift to concerns of confidentiality, integrity, and availability. The risk associated with p2p applications are not substantially different from those suffered by other network centric application types, including worms and viruses, buffer overflows, and weak or non-existent data/identity security. Although there can be real value to integrating p2p technology within the enterprise, appropriate return-on-investment calculations are necessary before adopting enterprise level software. Alternatively, sane risk-benefits analysis must be considered when introducing freeware into a secured environment. The potential risks must be fully enumerated and compared to the real worth of a p2p application.

If selected, p2p tools must be integrated into the corporate security policy. The policy must address appropriate use, configuration, and user training. For enterprise level solutions, integration with existing security infrastructure, such as PKI, enterprise directory services, and certificate authorities, is just over the horizon. This is not true for the majority of p2p tools, and is especially not true of the type of tools that are likely to be user installed inside an organization's perimeter. Until freeware tools cleanly integrate with existing security infrastructure, a best practice is to ban them entirely from the environment. Given the difficulty of stopping this traffic at the firewall, a host based monitoring solution that kills the application itself is the most efficient defense.

Sources:

Bellovin, Steven. "Security Aspects of Security Aspects of Napster and Gnutella." Archived webcast, November 28, 2001. URL: http://technetcast.ddj.com/tnc_play_stream.html?stream_id=625 (June 22, 2002)

Bellovin, Steven. "Security Aspects of Napster and Gnutella." URL: http://216.239.51.100/search?q=cache:yC4EjZm7gusC:www.usenix.org/events/u

senix01/invitedtalks/bellovin.pdf+Gnutella+ping+pong&hl=en&ie=UTF-8 (June 22, 2002)

Bert, Al. “PP, or not P2P?” Information Security Online. February 2001. URL: http://www.infosecuritymag.com/articles/february01/cover.shtml (June 22, 2002)

Borland, John. “Stealth P2P network hides inside Kazaa.” CNET News.com, March 1, 2002. URL: http://news.com.com/2100-1023-873181.html (June 4, 2002)

Borland, John. “Gnutella viruses weaker than email bugs, experts say.” CNET News.com. June 5, 2000. URL: http://news.com.com/2100-1023-241440.html?legacy=cnet (June 4, 2002)

Ethereal Users’ Guide v. 1.1 for Ethereal 0.8.19, Appendix A: Gnutella Protocol. URL: http://www.ethereal.com/docs/user-guide/sidgnutellaprotocol.html

Festa, Paul. “ICQ logs spark corporate nightmare.” CNET News.com. March 15, 2001. URL: http://news.com.com/2102-1023-254173.html?legacy=cnet (June 4, 2002)

Fout, Tom. “Inside Windows Messenger – How it Communicates.” Microsoft TechNet. October 2001. URL: http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/prodtechnol/winxppro/evaluate/insid01.asp (June 4, 2002)

Gartner Group. “Commentary – Free IM-not to be trusted.” July 12, 2001. URL: http://www.techrepublic.com/article.jhtml?src=search&id=r00120010712ggp01.htm (June 22, 2002)

Glave, James. “Net Messaging Called 'Catastrophic.” Wired News, June 5, 1998. URL: http://www.wired.com/news/technology/0,1282,12758,00.html (June 22, 2002)

Gomes, Lee. “Is P2P plunging off the deep end?” The Wall Street Journal Online (from ZDNet News.com). April 3, 2001. URL: http://zdnet.com.com/2100-11-529159.html?legacy=zdnn (June 22, 2002)

Hurwicz, Michael. “Emerging Technology: Peer-to-Peer Networking Security.” Network Magazine. February 6, 2002. URL: http://www.networkmagazine.com/article/NMG20020206S0005 (June 4, 2002)

Jones, Christopher. “File Trading Roundup.” Webmonkey.com. March 19, 2002. URL: http://hotwired.lycos.com/webmonkey/00/35/index2a.html?tw=multimedia

Lai, William and Movva, Ramu. “MSN Messenger Protocol 1.0.” Microsoft Corporation. Posted on Hypothetic.org. URL:http://www.hypothetic.org/docs/msn/ietf_draft.php\ (June 22, 2002)

McDougall, Paul. “The Power of Peer-To-Peer.” InformationWeek.com News. August 28, 2000. URL: http://www.informationweek.com/801/peer.htm (June 6, 2002)

Mintz, Mike, and Wilson, Nat. “MSN Instant Messenger Protocol.” Hypothetic.org. URL: http://www. http://www.hypothetic.org/docs/msn/basics.php (June 6, 2002)

Olson, Lance “.NET P2P: Writing Peer-to-Peer Networked Apps with the Microsoft .NET Framework.” Microsoft Corp. URL: http://msdn.microsoft.com/msdnmag/issues/01/02/NetPeers/NetPeers.asp (June 4, 2002)

Osborne, Rick. “Knowbuddy’s Gnutella FAQ.” Rixsoft.com. URL: http://www.rixsoft.com/Knowbuddy/gnutellafaq.html (June 22, 2002)

PDI/Dreamworks. “The Tech of Shrek,” *Shrek: Two-Disc Special Edition DVD*, Dreamworks/SKG, 2001

Salamone, Salvatore. “Instant messaging threatens enterprise security.” TechRepublic.com. January 4, 2002. URL: http://www.techrepublic.com/article.jhtml;jsessionid=5JZYJNU5LPQLKQD23VUCFFQ?id=r00520011218sss01.htm&src=search&_requestid=24149

Schiller, Jack. “An Instant War, Just Add Chat: The Growth of Instant Messaging Technology.” SANS Reading Room, September 13, 2001.

Sharp, Duane E. “Security in electronic messaging systems.” TechRepublic.com, August 21, 2001. URL: http://www.techrepublic.com/article.jhtml?id=r00520010821aue01.htm&src=search&_requestid=127286 (June 22, 2002)

Slaton, Joyce. “Dare to Share.” On Magazine. June 2000. URL: http://on.magazine.se/pdf/6_2000/Peer%20to%20peer.On_6_2000.pdf (July 1, 2002)

Staff Writer. “A Chat with the Master of P2P.” BusinessWeek online. August 1, 2001. URL: http://www.businessweek.com/technology/content/aug2001/tc2001081_460.htm (June 4, 2002)

Staff Writer. “Instant Messaging Joins the Firm,” Staff Writer. *The Economist* Technology Quarterly, pp. 5-6. June 22-28, 2002.

Staff Writer. “Peer-to-Peer Technology Issues and Commentary.” Business2.0. URL: http://www.business2.com/webguide/0,1660,5551,FF.html (June 4, 2002)

Sun Microsystems. “Security and Project JXTA.” whitepaper. URL: http://www.jxta.org/project/www/docs/SecurityJXTA.PDF (June 22, 2002)

Winer, Dave. “What is P2P?” Davenet.userland.com. September 20, 2000. URL: http://davenet.userland.com/2000/09/20/whatIsP2p (June 22, 2002)