Global Information Assurance Certification Paper

Fooling the Firewall

Understanding and blocking applications that circumvent firewall policy to provide connections from the Internet to the protected network

Karl Kranich
GSEC Practical Assignment v1.4
August 27, 2002

Abstract
Many firewalls are configured to allow outbound connections (from the protected network to the Internet), but deny inbound connections (see figure 1). However, there is a growing list of applications that effectively provide inbound remote control and file retrieval ability, if an agent is running on a machine on the protected network. If firewall rules allow outbound connections (even just web browsing), then these applications will allow outside machines to communicate with the protected machines. This paper examines GoToMyPC (remote control) and peer-to-peer file sharing applications, discussing their risks and showing how to block their use with Checkpoint's Firewall-1. A "defense in depth" strategy is also discussed.

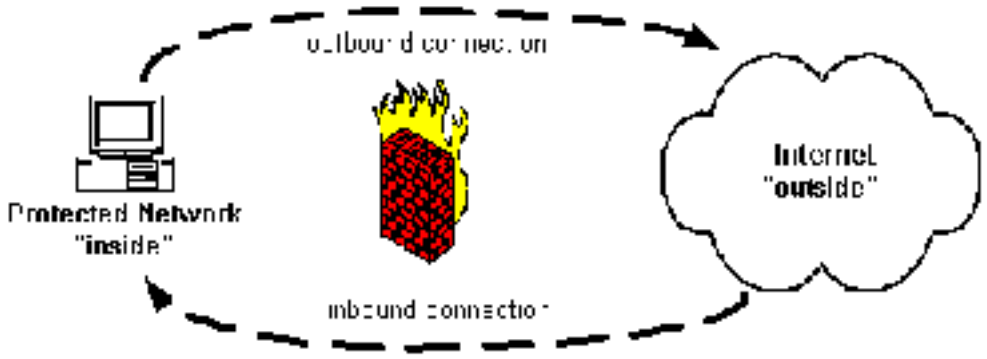

Figure 1: Connection direction

Introduction

Computers on the protected network can be shielded in several ways from connections from the Internet. First, firewalls can easily track the direction in which a TCP connection is initiated. The first packets of the TCP three-way handshake are uniquely identified by the flags they contain, and firewall rules can use this information to ensure that certain connections are initiated in only one direction. For example, firewall rules might specify that users can browse from their PCs to a web server on the Internet, but an outside user on the Internet cannot browse to the protected user's PC. Second, Network Address Translation usually works to protect inside users from inbound connections. Special reserved ip addresses may be used on the inside, and Internet machines can't direct traffic to the reserved addresses. When an inside user attempts an outbound connection, however, the firewall substitutes a valid ip address for the reserved ip address, and the connection is successful.

For these reasons, outside machines usually cannot connect to inside machines. However, GoToMyPC and the peer-to-peer file sharing applications create a clever way around this restriction. When run on an inside machine, they create an allowed, outbound connection to a machine on the Internet. Then, *other* machines on the Internet can access resources on the inside machine via the

outbound-initiated channel. Some of these applications will function even if the firewall rules only allow web browsing—they will automatically use the TCP port (80) that is usually reserved for http.

We will first examine GoToMyPC, then the peer-to-peer file sharing applications, and finally discuss a defense in depth protection strategy. Be sure to test the suggested Firewall-1 changes in a test environment before modifying production systems.

GoToMyPC

"Access your PC from Anywhere – Download Now" is the subject of the email advertising this remote control tool from Expertcity, Inc. In their own words, "GoToMyPC is a hosted service that enables secure browser-based access to any Internet-connected Windows PC" (GoToMyPC, "Making Life Simpler). GoToMyPC is an impressive tool, with very useful features and strong security. However, any user can set it up on his own, and then his PC is available from any web browsing user that knows the correct passwords.

How it works: a user installs a 1.4 MB service on the machine he wants to control (like his desktop at work-I'll call it the "host" for this discussion). He creates a password on the host (at least 8 characters, with letters and numbers). He then registers the host with GoToMyPC, creating a separate password there. The GoToMyPC servers never know the password set on the host. The host then "pings" (actually, a small TCP packet or HTTP request) the GoToMyPC "broker" server every 15 seconds, checking if there has been a connection request. The host initiates all of the communications, which is why it can get through firewalls and NAT devices.

When the user wants to control his host, he browses to the GoToMyPC web server and logs in. He is then presented with a list of hosts to which he has access. Then, he authenticates to the host with the password that only he and the host know. The GoToMyPC servers relay messages between the web client and the host, with the endpoints (host and client) initiating outbound connections to the GoToMyPC "middleman" servers. The communications are compressed, encrypted, and signed.

GoToMyPC will work in most cases without any reconfiguration of the corporate firewall. GoToMyPC hosts first try to contact the broker server over TCP port 8200. If that fails, HTTP GET requests are sent to port 80 on the broker server. If web browsing is allowed, the connection will succeed. "GoToMyPC adjusts itself to the firewall" (GoToMyPC, "Making Life Simpler").

Blocking the use of GoToMyPC: Despite the powerful features of GoToMyPC, there are reasons for corporations to restrict its use. Although it is built around a strong security model, GoToMyPC depends only on email addresses and passwords for authentication. A user could set both passwords (the web login

password and host password) the same, which would limit the strength of the security. Corporate policy may not allow for remote access, or may provide for it in a different way. Expertcity describes two ways to control the use of GoToMyPC in their FAQs (GoToMyPC, "FAQs"):

1. Sign up for GoToMyPC's Authorization Management Service (this is free), whereby administrators can control which machines can be GoToMyPC hosts. However, this option requires that the hosts have unique, public ip addresses. If all of the organization's PCs are hidden behind a single public ip address, this option doesn't work.
2. Block access to the broker server "poll.gotomypc.com". With Firewall-1's Policy Editor[1], this is as simple as creating a workstation object to represent poll.gotomypc.com, and then creating a rule like the following:

Figure 2: Firewall-1 GoToMyPC rule

When creating a workstation object such as poll.gotomypc.com, it is possible to use the "Get address" feature of the Policy Editor. However, this would not be a complete solution if there were multiple ip addresses for the dns name "poll.gotomypc.com". I recommend using "nslookup" to get the ip address. The following example shows how nslookup in Windows 2000 returns information for a dns name with one ip address (poll.gotomypc.com), and a dns name with multiple ip addresses (www.yahoo.com):

[1] The following assumptions relate to the Firewall-1 examples: Firewall-1 is version 4.1, and the object "CorpNet" represents the protected network.

```
C:\>nslookup
Default Server:  UnKnown
Address:  192.168.0.1

> poll.gotomypc.com
Server:  UnKnown
Address:  192.168.0.1

Non-authoritative answer:
Name:    poll.gotomypc.com
Address:  10.11.12.13

> www.yahoo.com
Server:  UnKnown
Address:  192.168.0.1

Non-authoritative answer:
Name:    www.yahoo.akadns.net
Addresses:  10.14.15.16, 10.14.15.17, 10.14.15.18, 10.14.15.19,
          10.14.15.20, 10.14.15.21, 10.14.15.22, 10.14.15.23
Aliases:  www.yahoo.com
```

Figure 3: nslookup output (ip addresses changed to protect the innocent)

If poll.gotomypc.com had returned multiple ip addresses, we would create multiple workstation objects in the Policy Editor, and then put them in a group or list them all in the rule.

I would like to commend Expertcity for openly providing guidance on blocking the use of GoToMyPC. However, in my opinion, designing applications that adjust themselves to firewall rules shows a lack of concern for corporate information security.

Peer-to-Peer File Applications

The goal of peer-to-peer (P2P) applications is to remove the distinction between client and server. Instead of running web browsers that can only request information from web servers, users run an application that can contribute files or resources in addition to requesting them. The idea of P2P is as old as the Internet itself—the original Internet hosts acted as both server and client for applications such as telnet and ftp. The big difference today is in the kind of nodes that make up P2P networks—individual users' PCs, primarily using dynamically assigned ip addresses (Oram, 22).

P2P applications can generally be divided into three categories (Berg, 40):
1. Distributed processing. Projects like SETI@home (http://setiathome.ssl.berkeley.edu/) take a large computing task and use Internet-connected PCs to divide and conquer.

2. Instant Messaging (IM). IM applications were created to allow people to chat with each other in real-time, but now also contain features like file transfer and voice message transfer.
3. File sharing. With P2P file sharing, each peer can simultaneously search for and offer files to other peers on the network. These applications exploded in popularity in 1999 when Napster was introduced to swap mp3 music files.

Each type of application introduces capabilities and risks. We will focus here on the concerns raised by the file sharing category.

Risks with P2P File Sharing
P2P applications are extremely popular on college campuses. And even with the high speed Internet links of many universities, **bandwidth utilization** can become a concern when music files (and even entire movies) are changing hands[2].

Another issue is that of creating yet another path for **viruses** to enter the corporate network. Gateways that scan email attachments and downloaded executables will probably not notice files downloaded through P2P applications. Antivirus software needs to be kept up-to-date on every client.

What could be even more dangerous to a business is the **unintended sharing of corporate information**. A misconfigured P2P application could make proprietary files available to the world. A study at HP Laboratories Palo Alto found that users "sometimes incorrectly assumed they were not sharing any files when in fact they were sharing all files on their hard drive" (Good). Remote users mapping drives to the corporate network with a VPN could accidentally share the corporate data on the Internet (Berg, 42). Most P2P applications don't include any form of authentication, so work-related collaboration over these networks is risky.

A related issue is that of **bugs** in the P2P software. One actual example was a worm that was spreading through the KaZaA network, sharing itself from machine to machine. "One modification of the worm (Worm.P2P.Duload.a) also downloads from an Internet site several Trojan programs designed to establish the unauthorized remote management of victim computers" (Kaspersky). Other P2P applications, including Gnut and eDonkey2000, have included buffer overflow and arbitrary script execution vulnerabilities (eGlobal and SecuriTeam).

Because of these risks and others, corporations may desire to block the use of P2P file sharing applications. We will look at several specific P2P "networks", and discover how to block their use with Firewall-1.

[2] There are traffic-shaping products, such as PacketShaper from Packeteer and Floodgate-1 from Checkpoint, that limit applications to specific amounts or percentages of available bandwidth.

Peer-to-Peer Protocols

There is an increasing number of P2P protocols with which peers communicate. All of the peers that use the same protocol to exchange messages with each other constitute a P2P "network". For some networks, there are many P2P applications available—especially when the network's protocol definition is openly published. www.afternapster.com lists 87 file sharing applications[3] at the time of this writing. Some P2P networks still contain dedicated servers that facilitate searches, but the file transfers happen directly from peer to peer. The peers join the network by connecting to one or more servers. Other networks are truly peer-to-peer, and a machine wanting to join the network needs some way to discover another peer running the same protocol. In these cases, host caches are usually maintained, and web sites are used to add to the caches.

Napster

Northeastern University student Shawn Fanning created Napster in 1999 to swap mp3 music files. The service was incorporated, grew quickly, and gained notoriety as billions of files were exchanged and several lawsuits were filed. Napster uses servers to maintain a central index of files--as users connect and disconnect from the network, the song list is updated automatically. But the design was revolutionary because the file transfer happens directly between the two users' machines. The servers allow searches to happen more quickly than in completely decentralized systems. However, when the Recording Industry Association of America and some artists decided to sue over copyright infringement, that central point of contact was found to be responsible for facilitating music piracy (RIAA). At this time, Napster is still shut down while arrangements are being made for a subscription service.

Although Napster, Inc, is gone as a free file sharing service, the Napster protocol lives on. The OpenNap project at SourceForge (http://opennap.sourceforge.net/) has created an open source Napster server, and Napigator maintains a list of servers and information about each one. There are over 180 servers in the Napigator Server List[4] (Napigator). These servers share any file type, not just mp3.

The original Napster servers were confined to six ip address blocks (Welch-Abernathy), and blocking access to those ip addresses would disable the service. Now that OpenNap servers can be found anywhere, a better strategy is to block the TCP ports that are used by the OpenNap servers. By looking through the Napigator Server List, we can see that the current Napster servers are running on 33 different ports. However, many of the ports are only used by one or two

[3] P2P applications are often called "clients", despite the fact that they serve both functions—client and server.

[4] Napigator offers an downloadable program that creates a Napster server list. When run in "standalone" mode, napigator.exe displays a server list that can be sorted in various ways or exported to a comma separated value file.

servers. The following ten TCP ports covers over 90% of the files reported by Napigator: 2345, 3456, 4444, 5555, 6565, 6666, 7777, 8888, 8899, and 9999.

To configure Firewall-1 to block most attempts to use Napster, create a TCP service for each port number (for example: "Napster2345", "Napster3456",etc.). Then create a service group called "Napster", and put all ten of the Napster TCP service objects into the group. Blocking Napster then becomes as simple as the following rule:

Figure 4: Firewall-1 Napster rule

Gnutella

Like OpenNap, Gnutella is used to share files of all types. In contrast to Napster, Gnutella is purely peer-to-peer. There are no central servers in this network. Gnutella isn't actually a piece of software, but is the underlying protocol that the peers use to communicate. Gnutelliums (http://gnutella.wego.com/) lists 15 different Gnutella applications.

Gnutella nodes function by sending messages within peer groups. The only thing that a Gnutella node (node A) needs to start up is the address of one other Gnutella node (node B). Node A sends a connect message to node B (called a "ping"). Node B forwards the ping on to the other nodes in its peer group, and responds to A with a ping reply. Eventually all of the nodes on the network send a ping reply to Node A[5]. Node A begins to form a peer group of its own, which will change over time.

When Node A wants to search for a file, it sends a search request to its peer group. The peer group forwards the request on, and also responds to A with any matches. Each search message contains a unique identifier so that the nodes can tell if they've already seen and handled the message. When node A wants to download a file (say, from node F), it connects directly to node F and requests the file. If node F is behind a firewall and node A can't communicate directly with node F, node A sends a PUSH request that is passed on to node F, and node F pushes the file to A.

Gnutella can also deal with firewalls that restrict outbound connections to certain ports (like port 80). The node behind the firewall just needs to find a node somewhere on the Internet running on port 80 and pass messages through it (Knowbuddy).

[5] Actually, the ping contains a Time To Live field, and will probably not make it to every node on the network.

How does a Gnutella node find its first peer? Companies that distribute Gnutella applications may run Gnutella nodes and distribute the software pre-configured to look for those nodes, or ip addresses can be found on web pages. Gnutella clients may retain lists of every node that they've heard of.

Blocking Gnutella is difficult, because there are no central servers or fixed port numbers. One way to recognize Gnutella traffic is by looking at the TCP data in the connection setup packets. The first thing a node does to establish a link with another node is to create a TCP connection and send the following ASCII encoded string: `GNUTELLA CONNECT/<protocol version>\n\n`

The other node responds with: `GNUTELLA OK\n\n` (LimeWire)

Firewall-1 is built around a packet inspection language called "INSPECT". INSPECT code can be written to handle special tasks that can't be accomplished with the GUI Policy Editor. The file `$FWDIR/lib/user.def` is the place to put "site specific INSPECT definitions" (Check Point, 381).

The following code should be added after the initial comment lines in `user.def` (save a copy of `user.def` before editing):

```
// reject TCP packets with first 4 payload chars = 'GNUT'
reject tcp, [TCPDATA,b]=0x474e5554;
```

After saving the file, install the security policy.

Explanation: The first line is a comment. The second line says that the packet being inspected should be rejected if it matches a certain condition. `TCPDATA` refers to the data portion of the TCP packet (the first 4 bytes, unless otherwise specified). The `b` refers to "big endian", meaning that the most significant byte comes first. The hex codes 47 4e 55 54 is the ASCII representation for "GNUT" (see Check Point, 291-317).

It is unlikely that the characters "GNUT" would happen to show up at the beginning of a non-Gnutella packet. However, you may want to look for the whole string "GNUTELLA" like this:

```
reject tcp, [TCPDATA,b]=0x474e5554, [TCPDATA+4,b]=0x454c4c41;
```

KaZaA

KaZaA is a music sharing application that uses the FastTrack network, which combines some of the features of Napster and Gnutella. There are no central servers, but nodes with sufficient processor power and Internet bandwidth can become "supernodes". Nodes upload their file lists to supernodes, and supernodes communicate with each other to perform searches (Sharman).

The FastTrack protocol is proprietary, but the KaZaA web page indicates that nodes talk to supernodes via port 1214. I have found that blocking TCP port 1214 does not prevent KaZaA from connecting to the FastTrack network, but it does prevent files from being transferred.

To configure Firewall-1 to block FastTrack applications, create a TCP service for port 1214. Then create a rule such as:

Figure 5: Firewall-1 KaZaA rule

The Need for Defense in Depth

These examples illustrate the need for more than just a firewall. With P2P applications appearing regularly, relying completely on technical solutions at the network perimeter is insufficient. As firewalls become more sophisticated, P2P developers design their protocols to look more like http. An organization that depends completely on a firewall can also be severely damaged by something as simple as a virus-infected software installation disk. A defense in depth strategy includes multiple layers of protection. With respect to the applications mentioned in this paper, the following layers are especially worth considering:

1. Written Policies. Every site should have an Acceptable Use Policy (AUP) that explains how information technology resources may be used. As new P2P applications are designed to work around firewall rules, relying completely on technical means of control becomes impossible (Berg, 50). Policies don't just serve notice to users who would like to misuse resources; they also let honest users know what applications they may or may not run. Security administrators need policies so that they know what consequences should result when violations occur, and so that they know that they are acting consistently toward all users.
2. Antivirus software. Especially critical in the Windows world, antivirus software should be deployed on every machine and kept up-to-date. A central console that gathers log messages and pushes updates is very useful.
3. Workstation standards. Having a standard workstation image can shorten an incident recovery dramatically. Additionally, an AUP that prohibits users from installing software of their choice protects the enterprise from applications that adapt to firewall rules.
4. Firewalls. Firewalls provide the first line of defense (with the exception of Internet router access lists). Firewalls are used to implement corporate policies that govern what kinds of Internet communications are allowed or prohibited. The protection offered by a firewall can be compromised by the applications described in this paper, and by poorly designed policy.

Conclusion

New applications that may not belong in your organization are constantly being developed. Information Security Officers and Firewall Administrators need to be aware of how these applications function and the risks accompanying them. Guided by organization policy and with some study of the protocols involved, an effective in-depth defense is possible.

List of References

Berg, Al. "P2P, or Not P2P?" Information Security. Feb. 2001 (2001): 38-51.

Check Point Software Technologies Ltd. Check Point Firewall-1 Architecture and Administration: Version 4.0. Check Point Software, 1998.

eGlobal Technology Services. "Gnut Gnutella Client Arbitrary Script Code Execution Vulnerability." 30 Aug. 2001. URL: http://www.safermag.com/html/safer40/alerts/06.html (25 Aug. 2002).

Gnutelliums LLC. "Gnutelliums." 2002. URL: http://gnutella.wego.com/ (25 Aug. 2002).

Good, Nathaniel S. and Aaron Krekelberg. "Usability and privacy: a study of Kazaa P2P file-sharing." 5 June 2002. URL: http://www.hpl.hp.com/techreports/2002/HPL-2002-163.pdf (25 Aug. 2002).

GoToMyPC. "GoToMyPC: Making Life Simpler for Teleworkers and Travelers." 2001. URL: https://www.gotomypc.com/downloads/pdf/GoOvrvew.pdf (23 Aug. 2002).

GoToMyPC. "FAQs – For Your Security." URL: https://www.gotomypc.com/help2.tmpl (23 Aug. 2002).

Kaspersky Lab. "Once Again A Virus Targets The KaZaA Network - Duload ." 11 Aug. 2002. URL: http://www.kaspersky.com/news.html?id=954839 (25 Aug. 2002).

"Knowbuddy's Gnutella FAQ." URL: http://www.rixsoft.com/Knowbuddy/gnutellafaq.html (26 Aug. 2002).

LimeWire. "The Gnutella Protocol Specification v0.4: Document Revision 1.2." URL: http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf (26 Aug. 2002).

Napigator. "Napigator Server List." URL: http://www.napigator.com/list.php (26 Aug. 2002).

"OpenNap: Open Source Napster Server." 15 Nov. 2001. URL: http://opennap.sourceforge.net/ (25 Aug. 2002).

Oram, Andy, ed. Peer-To-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly & Associates, Inc, 2001.

Recording Industry Association of America. "Napster Lawsuit Q&A." URL: http://www.riaa.com/Napster.cfm (26 Aug. 2002).

SecuriTeam. "eDonkey 2000 URL Buffer Overflow." 6 Aug. 2002. URL: http://www.securiteam.com/securitynews/5TP082K7FG.html (25 Aug. 2002).

SETI@home. "SETI@home: The Search for Extraterrestrial Intelligence." 2001. URL: http://setiathome.ssl.berkeley.edu/ (26 Aug. 2002).

Sharman Networks Limited. "KaZaA Frequently Asked Questions." 2002. URL: http://www.kazaa.com/en/help/faq/supernodes.htm (26 Aug. 2002).

Walsh, Lawrence M. "Blocking Napster Isn't Elementary: Academia Grapples with Bandwidth Abuse and Security Risks Posed by P2P Services." Information Security. Feb. 2001 (2001): 44-45.

Welch-Abernathy, Dameon D. "How Do I Block Access to Napster?" PhoneBoy's Firewall-1 FAQ's. 2002. URL: http://www.phoneboy.com/faq/0386.html (26 Aug. 2002).