GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at http://www.giac.org/registration/gcih

KaZaA Media Desktop Virus: W32/kwbot

By: Rita J Will
October 2, 2002
GCIH Practical Assignment V2.1 (revised April 8 2002)
Option 2 – Support for the Cyber Defense Initiative

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Table of Contents

Introduction....3
Targeted Port....3
Targeted service....4
Description....4
Protocol....6
- TCP....7
- UDP....8

Vulnerabilities....9
Specific Exploit....12
Exploit Details....12
Name:....12
Aliases:....12
Variants:....12
Operating System:....12
Protocols/Services:....12
Brief Description:....12
Worm or Virus?:....12
Description of Variants:....13
Protocol Description....13
How the exploit works....26
Diagram....31
How to use the exploit....34
Signature of the attack....35
How to protect against it....38
Source code/Pseudo code....44
Additional Information....44
Appendixes....45
A....45
B....49
References....52

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

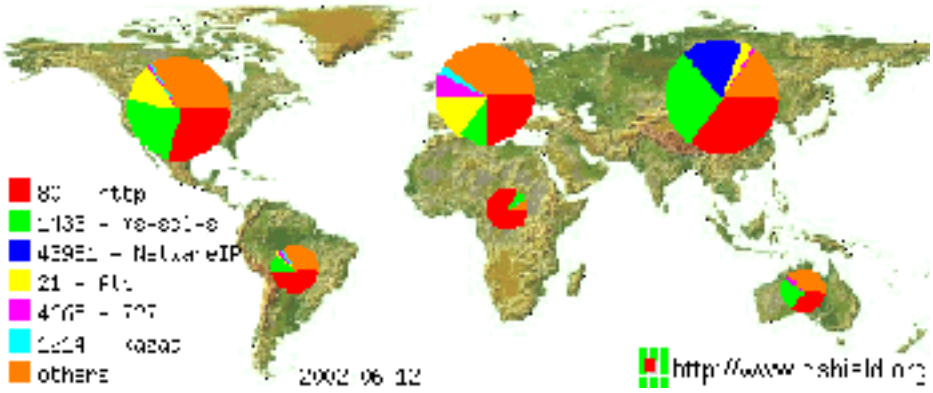
Introduction

"Let's Download Some Mariachi Music for the Cinco De Mayo Party!"

I never paid that much attention to music downloads until one day I wanted to download some mariachi music for a Cinco De Mayo Party and a friend of mine, that has a teenager, showed me how to do it. I couldn't get the idea out of my head that people were downloading and sharing files with everyone on the Internet directly from their hard drives. Not knowing who is accessing their shared folders or what they were getting when they downloaded a file. Are you only sharing what you intended? Is it really a media file you just downloaded? File sharing networks would be a good vehicle to compromise a number of home systems to use for a Distributed Denial of Service (DDOS) attack or to just penetrate a network. To me this seems to be a vulnerability "petree dish" and more exploits were on their way.

KaZaA UDP/TCP 1214 was on the top ten list and I decided to look for an exploit there. I found the KaZaA worm "Benjamin", then "K0wbot.1.3.B" came along, and I decided to use that as my topic. The KaZaA virus scratched the surface of what can be done in this open file-sharing environment if carefully planned and executed. This paper discusses how the KaZaA file sharing system works and how it could be used as a means to compromise numerous hosts and an example of an exploit that could have been used for this purpose.

Targeted Port



There are 65535 ports used by Internet Protocols for communication. They are defined in three groups.

Ports 1-1023 are the "well-known ports" assigned by Internet Assigned Numbers Authority (IANA) to ensure all use the same ports for common services. [1]"Well known

[1] Mastering Network Security by Chris Brenton

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

ports are de facto standards used to insure that everyone can access services on other machines without needing to guess which port number is used by the service".

Ports 1024-49151 are known as the "registered ports". They can be assigned by IANA for specific applications and services.

Ports 49152 through 65535 are "Dynamic" and/or "Private Ports". They are primarily used for return communication also called ephemeral ports. The ephemeral ports are typically chosen by the client's operating system when initiating a connection to a server/application.

Port 1214 is assigned by IANA to KaZaA for the Media Desktop communications.

Targeted service

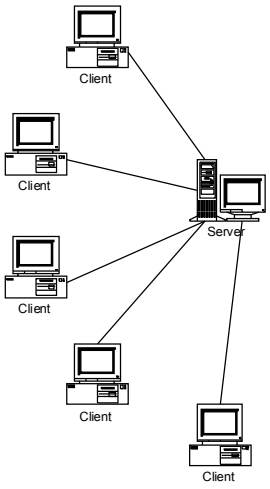
KaZaA is a client application referred to as "Media Desktop" used to connect to the FastTrack file-sharing network. Users are connected to thousands of other KaZaA users in a large Centralized/Decentralized file-sharing network where users can share files such as movies, music, documents, software, viruses, etc. Anything you choose to share with the outside world, and possibly some things that you didn't intend to share. Similar applications are Morpheus and LimeWire. Each client application is used to connect to a file-sharing network call Peer-to-Peer (P2P). Like KaZaA, the Morpheus client also connects to the FastTrack P2P. LimeWire is used to connect to a different type of P2P called Gnutella.

The Internet Assigned Numbers Authority (IANA) assigned UDP/TCP port 1214 for KaZaA P2P. This doesn't mean that an application developer cannot use that port for their application; it just means that IANA has officially assigned it for KaZaA.

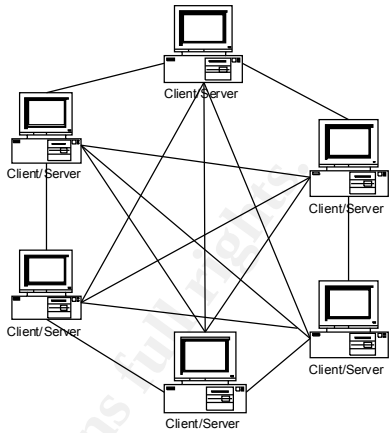
Description

There are two basic types of file sharing networks, client/server and P2P. In the client/server model, clients access the same server to share files. In the P2P model, all computers can act as client or server and can all access each other.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.



Client/Server



P2P Network

Early on the Internet performed as a Peer-to-Peer network. All computers could access each other directly and each could play the role of client or server. Then the Internet moved more into the client/server model. This happened when Firewalls, Network Address Translation (NAT) came on the scene. Recently, P2P networks have become popular again.

There are several file sharing networks available today using different topologies. Three of those topologies are Centralized, Decentralized or a combination of the two. Below is a brief description of each.

- Centralized topology would be similar to the Client/Server model shown above. There are a few media sharing networks that use this topology, one example is SETI@Home. In a centralized topology, users connect to a centralized sever where they can be registered and/or files can be subject to content filtering. The server would have all of the shared files, which would make that a single point of failure. If something happened to that server, the file-sharing system would be worthless.
- In the truly decentralized topology such as the Gnutella network, which is used by LimeWire, there is no centralized server, therefore, no registration, logging, or content filtering. However, this type of network is more scalable and fault-tolerant. If a Node shutdowns, it will not impact the rest of the network. The same file could be offered in many places throughout the network.
- The combination of centralized/decentralized is used by KaZaA and Morpheus. They are both based on FastTrack technology. A centralized server is still responsible for user registration and some logging. In this topology, the user first registers with a centralized server but downloading files, is requested from other peers like in a decentralized architecture.

KaZaA is a company in Amsterdam that developed the FastTrack P2P technology in March 2001. KaZaA "Media Desktop" and Morpheus are actually clients used to access

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

the FastTrack P2P. The developers took the Gnutella decentralized file-sharing concept and improved on it by adding "Supernodes" to perform as indexing servers. Along with Supernodes, there is also another participant in the FastTrack P2P, Peers. A Peer is everyone participating in the FastTrack network. A Supernode is a Peer that is designated as a Supernode because of bandwidth or processing power. Any Peer on the network could be used as a Supernode.

The FastTrack network grew quickly because of the already existing MusicCity users. MusicCity originally used the OpenNap file-sharing network but OpenNap was shutdown by the RIAA in 2001. MusicCity was then reintroduced to it's users as Morpheus using the FastTrack network. Morpheus was a modified version of the KaZaA client

In November 2001, KaZaA was sued by the International Federation of the Phonographic Industry for distributing copyrighted materials. To avoid the charges and possible shutdown, KaZaA was sold to Sharman Networks in Australia in January 2002. A couple of months later, it was determined that the users themselves were responsible for the illegal distribution of copyrighted material since they are shared and downloaded peer to peer. Therefore, the November ruling was overturned.

The most recent version of KaZaA can be downloaded at http://www.KaZaA.com. The user downloads it, installs it and they are on the KaZaA network sharing files whether they like it or not. Using the default settings shares a folder called "My Shared Folder" in the "Program files\Kazaa" directory. In the earlier version, KaZaA would search your hard drive for all media files to automatically share. It would share them regardless of where they were in the directory structure.

KaZaA and Morpheus are the most popular file-sharing clients available right now for a couple of reasons. 1) They allow the users to share more than just media files, you can share any file with any file extension. Many file sharing systems like Napster and AudioGalaxy, share only mp3 files. 2) They provide "SmartStream" and "FastStream" technologies. SmartStream addresses incomplete file downloads. For example, during a file download there is a network connection issue or the peer serving the file disconnects, SmartStream will connect to another user offering the same file or reinitiate the download when it is able. FastStream deals with download speed. If there is more than one peer that is offering the file for download, it will use the list of peers to download the file instead of just one. This increases the speed of the download. Other file-sharing systems like Napster and Gnutella were unable to perform these functions which caused users to be disappointed because of frequent incomplete file transfers and slow downloads.

Protocol

There are two Internet protocols used by the KaZaA Desktop for communication and file sharing. They are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

- **TCP**

TCP is considered a connection-oriented communication, which can provide a reliable means to deliver packets. The reason it is referred to as connection-oriented, it establishes a connection between two hosts and that connection remains until it is torn down when the communication is complete.

To establish a connection, first a "three-way handshake" is performed. This is the Client sending a "syn" packet to the Server saying "I am requesting a connection with you". If the Server receives the "syn" packet, he will respond with a "synack" packet saying "I got your request to initiate a connection with me." The Client will respond to the "synack" with an "ack" to let the Server know that he received the "synack". The connection is then established.

The Client and the Server keep track of their individual communications with numerous hosts by using source IP and source port with destination IP and destination port.

Another piece of important information contained in the TCP communication is the sequence numbers. When the Client sends his initial "syn" packet, it includes a sequence number. The "synack" returned from the Server, acknowledges the sequence number provided by the Client incrementing it by one, and provides a sequence number of his own. The "handshake" is completed and they can start passing data back and forth.

Here is an example of a complete TCP connection. User A wants to download file from Server B. User A would send a "syn" packet requesting a connection to be established. Server B would respond with a "synack" to acknowledge the "syn". User A would return an "ack" packet to acknowledge the "synack" and compete the handshake.

Next Server B would send a “push” packet probably asking for a Username. User A would provide his Username in a “push” packet back to Sever B. Each “push” packet is acknowledged by the recipient. Most times the “ack” will be sent with the next “push” packet but there are cases where just an “ack” packet will be sent to acknowledge receipt of the data. User A might send a data packet or "push" packet with an "ls" command. The "ls" command would show User A a list of files that are available for download. Server B would respond with an "ack" acknowledging receipt of the bytes in the "push" packet and then send a "push" packet with the list of files as requested. It is possible that the data in the "push" packet be transferred in more than one packet. For example, the list of filenames might be transferred in two packets. TCP will keep track of the packets of data by keeping track of the bytes transferred and sequence numbers in order to put them back together in the proper order. User A asks to download a particular file and Server B will send a series of packets each containing a chunk of the data until the entire file has been transferred. User A acknowledges the bytes as they are received. After all of the bytes have been received, User A can close the

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

connection if he does not want to download another file. To do that, he sends a "fin" packet to Server B. Server B responds with an "ack", acknowledging the "fin" and sending another "fin" packet to User A agreeing that he is also ready to close the connection. User A acknowledges Server B "fin" packet and the connection is closed gracefully.

In addition to keeping track of the bytes and acknowledging them, TCP also offers another service, a time-out mechanism to detect if data has been lost. If the next byte of data has not been received in a certain time or data has not been acknowledged, TCP will ask that packet to be retransmitted. This makes TCP the most reliable Internet Protocol to transfer data.

- **UDP**

UDP is a connectionless communication. No handshake is required for this protocol to communicate. UDP uses ports to distinguish between several connections as apposed to sequence numbers. It provides no reliability or packet recovery as TCP does. UDP packets are smaller and use less resources than TCP. It is used where reliability is not required.

For downloads and searches, data is being transferred between Supernodes and peers, therefore TCP is used for reliability.

UDP port 1214 is used for "peer to Supernode" and "Supernode to Supernode" communication. Since KaZaA has many Supernodes that a peer can communicate with, it is not necessary to establish connection with all of them. So in order to do a quick search to see if a Supernode is available, it uses UDP. If a packet gets lost, it's not important.

When the "KaZaA Media Desktop" has been started at either window start up or when the user double clicks on the proper desktop icon, the communication begins with a KaZaA server on port 80. The KaZaA server receives identification information from the user (peer) and then provides information to the peer about nearby Supernodes (peers designated as Supernodes). The peer then uses UDP 1214 to check for those Supernodes. It follows up with TCP 1214 for any Supernode from whom the peer receives a UDP 1214 return packet. If the TCP 1214 handshake sequence is completed, the peer knows that Supernode is available for a search task.

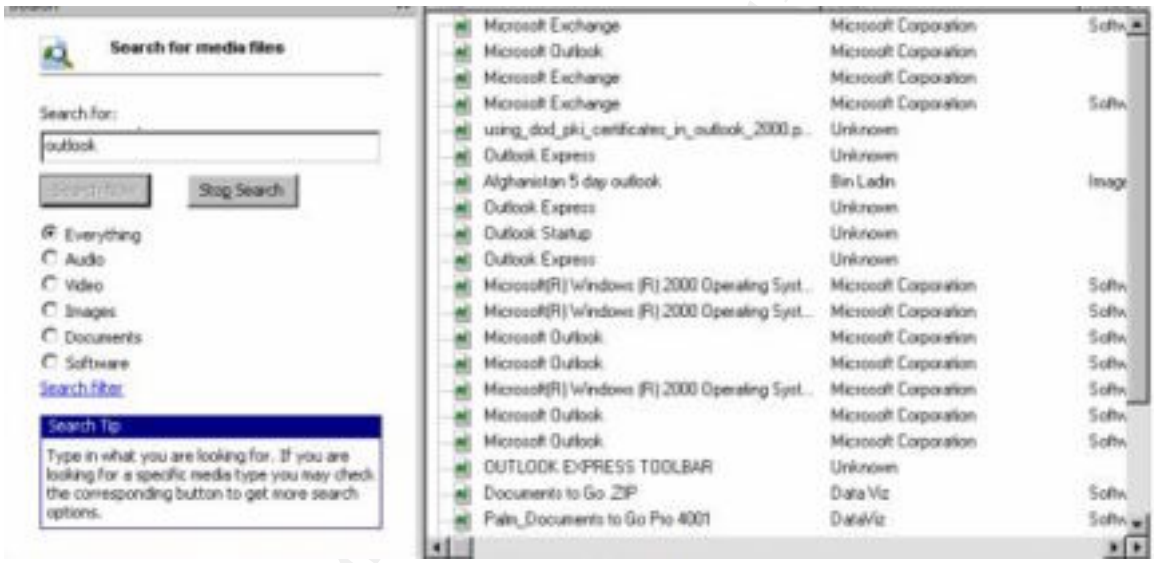
When the peer wants to search for a file, the request will go to the Supernode using TCP 1214 and ask to find a string, the Supernode will contact other Supernodes and peers to find a list of files that contain the string and provide a list to the requesting peer. The peer can choose from that list to download the desired file(s).

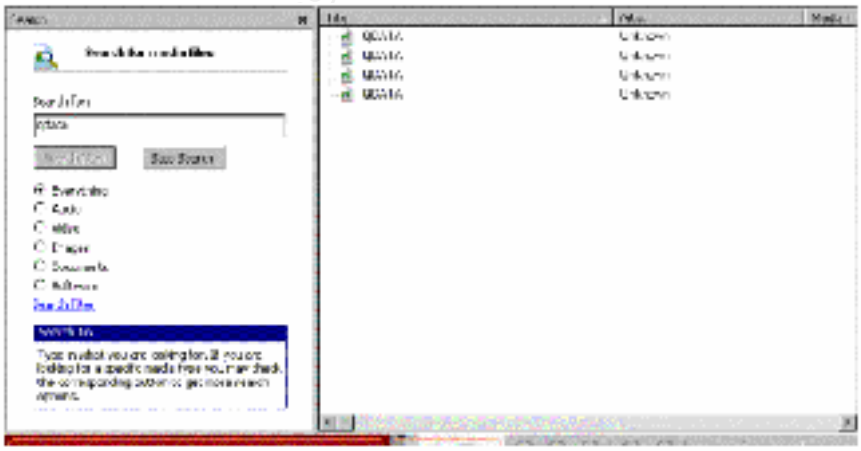
Downloading is done on TCP 1214 and it is performed peer-to-peer. The Supernode is out of the communication at this point and just stands by for the next search task.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Vulnerabilities

Having KaZaA up and running on your desktop makes you somewhat vulnerable. KaZaA users typically are not aware what files folders they are sharing or how to stop sharing. A laboratory study on the usability of KaZaA discovered [2]"…the majority of the users in our study were unable to tell what files they were sharing, and sometimes incorrectly assumed they were not sharing any files when in fact they were sharing all files on their hard drive." This is true with many applications and software, not enough information or knowledge of the application can result in vulnerabilities due to configuration errors. It would be easy to inadvertently share files or file folders that may have personal information, finances, e-mail, cookies with passwords, credit card information, etc. When you choose to allow users to access files directly from your computer, you are putting yourself at a considerable risk. Below are screen-prints of a search using the word "outlook" another one when searching for "qdata" (Quicken default filename).





[2] Usability and privacy: a study of Kazaa P2P file-sharing, by Nathaniel S. Good and Aaron Krekelberg

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

The search found some users that are sharing their financial information from Quicken and some that are sharing their e-mail with the outside world as well. The quicken files could include bank account numbers.

Using the default configuration of the most recent version of KaZaA, c:\Program Files\KaZaA\My Shared Folder is shared by default. There are three ways in KaZaA to choose to share other folders, which to some is a little confusing on what is selected. If you click the check box for the "My Documents" folder, it may not be clear to many users that all folders and files under that are now going to be shared. There is only one place to remove folders from being shared and it is a menu item called "Finding Media to Share". It's not clear that you can see what you are sharing or that you might be able to change the settings.

Downloading software or media files from the Internet is a risk no matter how it's accomplished, but I would consider downloading files directly from unknown users to be a much greater risk. As mentioned in the introduction, users are downloading files from someone they don't know and aren't exactly sure what they are getting. KaZaA has a warning on their website stating [3]"All files that are accessible using KaZaA Media Desktop (KMD) originate from other users of KaZaA. This means that there will always be the risk of irresponsible users introducing viruses." They say they are trying to do something about that, but they really can't help in this environment without placing themselves in the middle and scan everything that goes back and forth. Currently KaZaA has a centralized server that users connect to each time they bring up KaZaA media desktop. They have to register to that server, which means there is some sort of logging and accountability when users first come on to the network. However, the user information can be bogus and may not provide an accurate security audit trail. Once the registration is complete, the KaZaA server is out of the communication. Unless the user asks to refresh the page or clicks start to reconnect. Since the Supernodes are just Peers, there is no logging or content filtering being done at that point.

KaZaA comes bundled with some anti-virus evaluation software in an effort to protect their users. You can choose to download and use the software as a trial or for a price.

Some Peer-to-Peer filing sharing systems only allow sharing of mp3 files (i.e. AudioGalaxy). This would require a hacker to successfully insert malicious code into an mp3 file and figure a way to execute it. There has only been one report that I could find of a buffer overflow using an mp3 file and Winamp player. KaZaA on the other hand, allows users to share any type of file with any file extension. This would allow a hacker to easily inject a virus into the KaZaA Peer-to-Peer network.

Another reason KaZaA users are vulnerable and a potential target for compromise is the fact that they are usually home users with no virus detection, firewalls, or Intrusion Detection Systems (IDS). Typically, home users are not aware of the risk involved with

[3] http://www.kazaa.com/en/help/virus.htm

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

a fast Ethernet connection to their computer. No knowledge of what is open from the Internet by default or how active the hacker community really is.

Anti-virus software is usually supplied with the home computer software suite, but the user must authorize the update to take place in most cases. The anti-virus software can be useless, if the signature list is not updated regularly. If the anti-virus software is not included, the user must buy a copy for their personal use. There are a few free ones out there, but are not as effective as the top ones such as McAfee and Norton.

A firewall on a home computer or at the perimeter could prevent inbound connections from the Internet. Most home users are not hosting a website, so there would be no need for any connections to be initiated from the Internet to their computer. So in the case of a virus that opens a backdoor for a hacker, the firewall would not allow the inbound connection to be initiated and therefore the backdoor would be useless. Another point to make about the Network Firewall at the perimeter preventing inbound connections, no other KaZaA user would be able to access your share folder. When a user requests to download a file from you, they attempt to initiate a connection and a network perimeter firewall would stop that attempt.

Computer based firewalls on the other hand, like BlackIce or Zone Alarm, restrict access by application. This would still allow other KaZaA users to access your shared folder. This is still not going to prevent the user from downloading a virus or sharing some personal files inadvertently.

Intrusion Detection Systems (IDS) in front of the home network would benefit if the user did download a virus and opened a backdoor if the IDS rules were properly configured to pick up particular attack signatures. If the anti-virus software did not detect the virus, possibly the IDS system would alert the user that it detected some unusual activity.

There are free firewalls and IDS systems available, but the point is that the average home user would not be concerned with the possibility that someone would want to access their computer and use it for malicious activity. Many would not take the extra steps to secure their home computer. There is also the problem of improperly configuring these security devices and creating a false sense of security, allowing viruses to go undetected for long periods in the home user environment. This opens them up for a patient hacker that wants to start the distribution of a virus and let it propagate at it's own rate. When enough computers are under his control, use them for a possible distributed attack.

So far, we have only talked about using the virus to infect many computers to for a DDOS, but the virus could also be used to penetrate a network. All that is needed is an internal user to download the virus and infect his workstation. If users were allowed to use IRC chat from inside their network, the hacker would be free to access that system through his channel that he created with the virus. If the system successfully sent a message with information about the infected system, the hacker would know it was there and available. If the hacker has access to a workstation on the inside, it wouldn't be difficult to map out most of the network by looking at the workstation configuration,

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

routes, host files, DNS, etc. An example is the hack into Monkey.org in March of 2002 (article can be found at http://lwn.net/Articles/1486/. An administrator downloaded IRSSI chat client at the request of the users, and installed a code that was already contaminated. It contained a "backdoor" that was used to gain root access to a server at Monkey.org to contaminate some popular freeware with some malicious code. The malicious code in the freeware also opened a backdoor. One user on the internal network, permitted to download, install, or use open chat channels, can compromise the entire network.

One last point about KaZaA users vulnerability is their unlawful distribution of copyrighted materials. As mentioned above, the Dutch courts have determined that it is the users responsibility to properly handle copyrighted materials. The P2P client companies may no longer be held responsible and the users themselves will have to pay the price if the law decides enforce.

Specific Exploit

Exploit Details

Name:

W32.K0wbot.1.2/1.3.a/1.3.b

Aliases:

BackDoor-AGT
WORM_KWBOT.A
W32.Kwbot.Worm
W32/Kwbot-A

Variants:

Worm.Kazaa.Benja or W32/Benjamin

Operating System:

Windows 95/98/NT/ME/2000/XP

Protocols/Services:

TCP/UDP Port 1214 and reports of port 6667 for the IRC backdoor.

Brief Description:

The victim downloads the virus from another KaZaA user and it replicates itself 150 times disguised as popular media files. Some report that it contains an IRC client that waits for commands from the attacker.

Worm or Virus?:

There are many conflicting descriptions of worms vs. viruses. From the Sans Incident Handling Course Material, it appears that this exploit doesn't really fit into either

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

category neatly. It fits more closely to the virus description. [4]"Malicious code is called a worm when it requires no specific action on the part of the user to enable infection and propagation. It just spreads. If the code requires the user to open an e-mail or load a screen saver or take some action, then it is called a virus." This malware requires the user to download the file from an infected machine and then execute it in order for it to infect the machine. It does not propagate on it's own. Therefore, for the remainder of this paper, Kwbot will be referred to as a virus or malware.

Description of Variants:

The first virus to infect the KaZaA community was known as the Benjamin virus. Both put a copy of itself in the Windows/System directory and modified the registry to load at startup. Like Kwbot, Benjamin replicated itself in the share folder with many different names to attract more victims. However, Benjamin replicated itself 1000+- times in the share folder and Kwbot only created approximately 150 all the same file size (21KB). Benjamin padded the files with garbage bytes so the file sizes were inconsistent. The virus itself was 216KB. Since it created so many copies of different size files, it filled up the users hard drive or crashed the computer while processing. Some users reports 100 percent of their processing cycles were being consumed by the virus.

Benjamin was reported to have opened a \Window\temp\system32 directory as a share in KaZaA. It was determined by F-Security, that the virus was written to make money because it opened a web page in Germany that contained advertisements.

Protocol Description

NOTE: For packets captures and screen prints in the following sections, IP addresses have been changed to private address space and the Usernames changed to hide the identity of the real users.

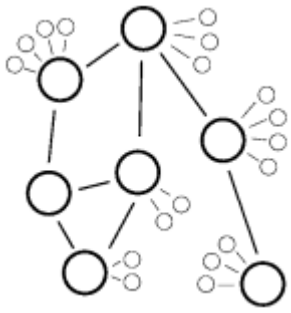
KaZaA uses the FastTrack P2P technology, which is considered a centralized and decentralized file sharing system. The reason it is consider a combination of those types is the peer has a centralized relationship with a Supernode. The Supernodes then work together as a decentralized network and the peers download directly from each other as in a decentralized file-sharing network. The Supernodes propagate queries among themselves in order to fulfill their peer's requests. For example, a KaZaA user decides to search for a file with the word "working", the request goes to a known Supernode and the Supernode initiates a search. If the Supernode does not have a sufficient index of files with that string it will ask another Supernode and that Supernode will possibly pass that request on to another. To avoid long searches using resources, the search will be limited by a TTL (time to live), meaning it will only propagate so many hops before it quits looking. The Supernodes can be any KaZaA user on the network, they are usually chosen for their bandwidth and processing power.

Below is an illustration of the Centralized/Decentralized file-sharing network.

[4] Sans Institute. Incident Handling Foundation, Malicious Software. Sans 2001, 2002.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

[5]



Once KaZaA is downloaded and installed, it creates an icon on the desktop called "Media Desktop". Double click is required to open the media desktop. However, KaZaA is running after boot up in the background. If the media desktop window is closed it will still be minimized in the system tray.

When starting the "media desktop" with all of it's advertisements, it doesn't launch the default web browser. It does make use of http (port 80) for making sure it has a connection to the Internet and can contact KaZaA. The normal TCP handshake is completed and the user is connected.

After the user was connected, nmap was used to determine what ports were open on the test box with the KaZaA Media Desktop client is active. Nmap is a free tool used here to perform port scans and vulnerability testing. The scanning box was running Linux OS and was located on the same network as the target system and complete scan of 65535 UDP and TCP ports was initiated. Below is the result of an nmap scan. Nmap shows the IP address of the box that was scanned, 192.168.1.102. The next line shows that 131065 ports scanned are closed. That means that we scanned 65,535 UDP and TCP ports totaling 131,070 and 5 ports are listed in a state of open which leaves 131,065 ports as closed.

Ports UDP/TCP 1214 are open and shown as unknown. These are the ports that are opened by KaZaA for the normal activity of the KaZaA user. This means that the computer is listening on these ports for any incoming communication using those ports. For example, if another KaZaA user wants to download a file from this box, they would send a "syn" packet to this computer on TCP port 1214 and this computer would accept that request and respond accordingly. Then there are the typical netbios ports for windows, ports 137, 138 and 139. All windows operating systems open these netbois ports by default. These ports are used for mapping network drives; windows name resolution; and other windows services.

The next thing nmap will indicate is how long it took to complete the scan. In this case it took 86 seconds. Scans such as this would take longer across the Internet, this was done on a small internal network.

[5] http://www.openp2p.com/pub/a/p2p/2001/12/14/topologies_one.html

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.1.102):
(The 131065 ports scanned but not shown below are in state: closed)
Port       State       Service
137/udp    open        netbios-ns
138/udp    open        netbios-dgm
139/tcp    open        netbios-ssn
1214/tcp   open        unknown
1214/udp   open        unknown


Nmap run completed -- 1 IP address (1 host up) scanned in 86 seconds
```

Port 80 does not show in the nmap scan above. That is because it only uses it to make an outbound connection to the KaZaA server. The ports that are listed are open, which means that are waiting to accept inbound traffic on those ports and are tied to services or applications running on the box. The port 80 communications from the peer to the KaZaA server, is an outbound connection only.

KaZaA uses port 80 for the user registration as shown below. The first two packets show the "syn", "synack" and the "ack" to complete the TCP handshake highlighted below. The fourth packet shows the KaZaA user, 192.168.1.102 sending a request for the media desktop page shown in bold below.

```
17:48:27.833129 192.168.1.104.1903 > kazaa.server.http: S [tcp sum ok]
20223617:20223617(0) win 8192 <mss 1460,nop,nop,sackOK> (DF) (ttl 128, id
58391, len 48)
0x0000   4500 0030 e417 4000 8006 991f c0a8 0168        E..0..@........h
0x0010   xxxx xxxx 076f 0050 0134 9681 0000 0000        .t...o.P.4......
0x0020   7002 2000 461a 0000 0204 05b4 0101 0402        p...F...........
----------------------------------------------------------------------
17:48:27.978816 kazaa.server.http > 192.168.1.104.1903: S [tcp sum ok]
1816788769:1816788769(0) ack 20223618 win 65535 <mss 1460> (ttl 48, id 12336,
len 44)
0x0000   4500 002c 3030 0000 3006 dd0b xxxx xxxx        E..,00..0....t..
0x0010   c0a8 0168 0050 076f 6c49 ff21 0134 9682        ...h.P.olI.!.4..
0x0020   6012 ffff 0fa5 0000 0204 05b4 40d7             `...........@.
----------------------------------------------------------------------
17:48:27.978974 192.168.1.104.1903 > kazaa.server.http: . [tcp sum ok]
ack 1 win 8760 (DF) (ttl 128, id 60183, len 40)
0x0000   4500 0028 eb17 4000 8006 9227 c0a8 0168        E..(..@....'...h
0x0010   xxxx xxxx 076f 0050 0134 9682 6c49 ff22        .t...o.P.4..lI."
0x0020   5010 2238 052a 0000                            P."8.*..
----------------------------------------------------------------------
17:48:27.979749 192.168.1.104.1903 > kazaa.server.http: P [tcp sum ok]
1:320(319) ack 1 win 8760 (DF) (ttl 128, id 60439, len 359)
0x0000   4500 0167 ec17 4000 8006 8fe8 c0a8 0168        E..g..@........h
0x0010   xxxx xxxx 076f 0050 0134 9682 6c49 ff22        .t...o.P.4..lI."
0x0020   5018 2238 51d7 0000 4745 5420 2f65 6e2f        P."8Q...GET./en/
0x0030   6b6d 6473 7461 7274 2e68 746d 3f63 6c69        kmdstart.htm?cli
0x0040   656e 743d 6b6d 6426 7665 723d 3137 3220        ent=kmd&ver=172.
0x0050   4854 5450 2f31 2e31 0d0a 4163 6365 7074        HTTP/1.1..Accept
0x0060   3a20 2a2f 2a0d 0a41 6363 6570 742d 4c61        :.*/*..Accept-La
0x0070   6e67 7561 6765 3a20 656e 2d75 730d 0a41        nguage:.en-us..A
0x0080   6363 6570 742d 456e 636f 6469 6e67 3a20        ccept-Encoding:.
0x0090   677a 6970 2c20 6465 666c 6174 650d 0a49        gzip,.deflate..I
```


© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
0x00a0   662d 4d6f 6469 6669 6564 2d53 696e 6365        f-Modified-Since
0x00b0   3a20 4672 692c 2032 3720 5365 7020 3230        :.Fri,.27.Sep.20
0x00c0   3032 2030 393a 3533 3a34 3620 474d 540d        02.09:53:46.GMT.
0x00d0   0a49 662d 4e6f 6e65 2d4d 6174 6368 3a20        .If-None-Match:.
0x00e0   2234 3135 6465 2d34 6135 642d 3364 3934        "415de-4a5d-3d94
0x00f0   3261 6161 220d 0a55 7365 722d 4167 656e        2aaa"..User-Agen
0x0100   743a 204d 6f7a 696c 6c61 2f34 2e30 2028        t:.Mozilla/4.0.(
0x0110   636f 6d70 6174 6962 6c65 3b20 4d53 4945        compatible;.MSIE
0x0120   2034 2e30 313b 2057 696e 646f 7773 2039        .4.01;.Windows.9
0x0130   3829 0d0a 486f 7374 3a20 6465 736b 746f        8)..Host:.deskto
0x0140   702e 6b61 7a61 612e 636f 6d0d 0a43 6f6e        p.kazaa.com..Con
0x0150   6e65 6374 696f 6e3a 204b 6565 702d 416c        nection:.Keep-Al
0x0160   6976 650d 0a0d 0a                              ive....
```

The KaZaA servers then sends the complete Media Desktop page, as we will see below.

Ethereal was used to piece the entire conversation together and make packets dumps easier to read to see where the peer registers with the KaZaA server using his username. Ethereal is a free tool that will string together the TCP conversation and it's data, allows for easy filtering, and real time packet captures on Windows or Unix platforms.

The first section is the data that the peer sent to the KaZaA server. On the first line, we see the request for the Media Desktop (kmdstart) and then there is information about the peer, for example, that it is a Windows 98 OS. The KaZaA server tells the peer what type of system it is and what version of Apache it's using. All the same information included in the packets dumps, Ethereal just puts in a little easier to read format. Still no clear indication of where the registration takes place.

```
GET /en/kmdstart.htm?client=kmd&ver=172 HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
If-Modified-Since: Fri, 27 Sep 2002 09:53:46 GMT
If-None-Match: "415de-4a5d-3d94-2aaa"
User-Agent: Mozilla/4.0 (compatible; MSIE 4.01; Windows 98)
Host: desktop.kazaa.com
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Sun, 29 Sep 2002 20:45:09 GMT
Server: Apache/1.3.26 (Unix) PHP/4.2.2
Last-Modified: Fri, 27 Sep 2002 09:53:29 GMT
ETag: "1df992-4a5d-3d94-2a99"
Accept-Ranges: bytes
Content-Length: 19037
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>



..<head>
...<meta http-equiv="content-type" content="text/html;charset=iso-8859-1">
...<meta name="generator" content="Adobe GoLive 6">
...<title>Kazaa</title>

....<link rel="STYLESHEET" type="text/css" href="/stylesheets/kazaa.css">
```

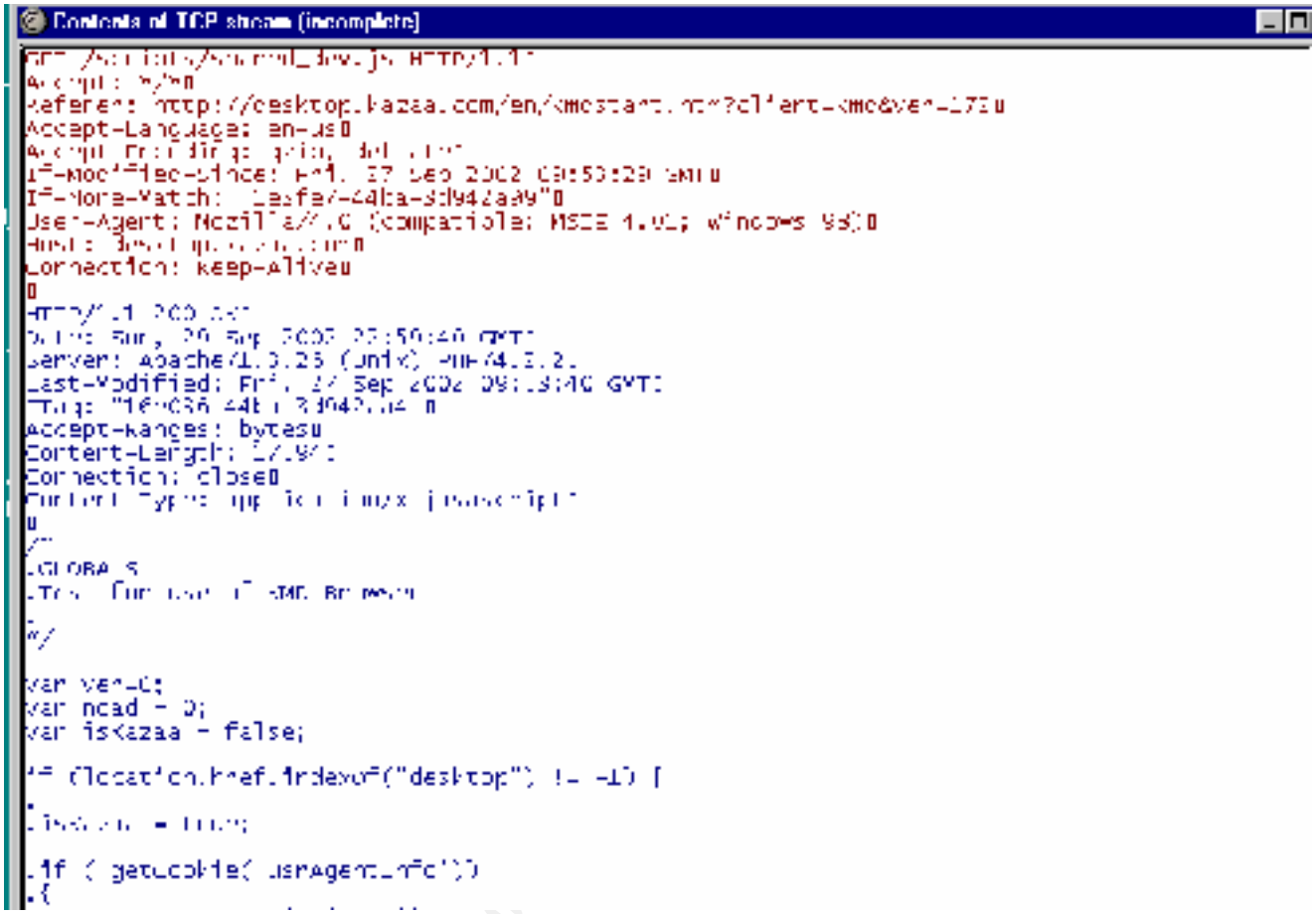
© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

The next screen print shows where each time the Media Desktop is started, there is another communication with another server at KaZaA called the update server. Here we see information about the user is passed to server to check for any changes since the last time. The first line shows the e-mail address that is configured when the KaZaA client is first installed, whether the user chooses to get the newsletter or not, and the version of the KaZaA client. Still doesn't look like the user registration is being done here.

Further analysis of the packet captures and using Ethereal has proved that the Username registration is being done with a cookie. The Username information is located in the c:\Windows\user.dat file. Here is a screen print of the Ethereal TCP stream that led to that conclusion.

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.



In the last line of the screen dump, it says "getCookie('usrAgentInfo')". The "usrAgentInfo" string is located in the c:\Windows\user.dat file and contains the Username and share folders. The above was a TCP stream captured between the KaZaA server and the peer after the initial KaZaA Media Desktop page had been sent.

Once the user registration is completed, a series of UDP 1214 packets are transmitted from the KaZaA Media Desktop to various addresses on the Internet. In the example below, the names have been changed to "Supernode1", Supernode2", etc, in order to hide the identity of the real users. Remember Supernodes are just Peers with large bandwidth and fast processors, so these Supernodes are in most cases home users like our test box. These are the Supernodes provided by KaZaA. The peer, 192.168.1.102, is letting the Supernodes know that he is online and the peer is finding out which Supernodes are online and available.

Below is an example of the UDP 1214 communication. A UDP 1214 packet goes out to each one of the Supernodes on the Peers list. In the first packet the Peer, 192.168.1.102 sends a UDP 1214 packet to Supernode1.optonline.net. Note that the UDP packets do not have a "syn" label or "ack" label as the TCP packets do. As discussed in the Protocol section, the UDP is a connectionless protocol, so it doesn't perform the handshake.

```
08:37:19.147179 192.168.1.102.1214 > Supernode1.optonline.net.1214:
[udp sum ok] udp 12 (ttl 128, id 20524, len 40)
```


© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
0x0000   4500 0028 502c 0000 8011 35a3 c0a8 0166        E..(P,....5....f
0x0010   xxxx xxxx 04be 04be 0014 0a11 2700 0000        ...+........'...
0x0020   2980 4b61 5a61 4100                            ).KaZaA.
```

```
08:37:19.147630 192.168.1.102.1214 > Supernode2.client2.attbi.com.1214:  [udp
sum ok] udp 12 (ttl 128, id 20780, len 40)
0x0000   4500 0028 512c 0000 8011 03de c0a8 0166        E..(Q,.........f
0x0010   xxxx xxxx 04be 04be 0014 d94b 2700 0000        ...-.......K'...
0x0020   2980 4b61 5a61 4100                            ).KaZaA.
```

```
08:37:19.148043 192.168.1.102.1214 > Supernode3.hot.rr.com.1214:  [udp sum ok]
udp 12 (ttl 128, id 21036, len 40)
0x0000   4500 0028 522c 0000 8011 3084 c0a8 0166        E..(R,....0....f
0x0010   xxxx xxxx 04be 04be 0014 06f2 2700 0000        ............'...
0x0020   2980 4b61 5a61 4100                            ).KaZaA.
```

```
08:37:19.148454 192.168.1.102.1214 > Supernode4.cfl.rr.com.1214:  [udp sum ok]
udp 12 (ttl 128, id 21292, len 40)
0x0000   4500 0028 532c 0000 8011 381c c0a8 0166        E..(S,....8....f
0x0010   xxxx xxxx 04be 04be 0014 0f8a 2700 0000        ...S........'...
0x0020   2980 4b61 5a61 4100                            ).KaZaA.
```

```
08:37:19.148863 192.168.1.102.1214 > Supernode5.stny.rr.com.1214:  [udp sum ok]
udp 12 (ttl 128, id 21548, len 40)
0x0000   4500 0028 542c 0000 8011 8424 c0a8 0166        E..(T,.....$...f
0x0010   xxxx xxxx 04be 04be 0014 5c92 2700 0000        ._........\.'...
0x0020   2980 4b61 5a61 4100                            ).KaZaA.
```

If the peer receives a UDP port 1214 packet back from one of the Supernodes, it sends a TCP 1214 Sync packet to the Supernode to establish a connection. If the Supernode responds with a "synack", the Peer sends an "ack" to complete the TCP handshake and the connection is established. Once he establishes a connection with a Supernode he does not continue to find one to connect. On the other hand, if the Supernode sends a reset, the peer knows he isn't available and he moves on to the next Supernode on the list.

After the peer has established a communication with a Supernode, there communication stays alive until one of the two hosts cancels the connection by logging off KaZaA, shutting down their computer, network problems, etc. The communication is kept alive with a periodic communication with the Supernode initiated by the Peer so another handshake is not necessary for them to communicate. The first packet shows the Peer, 192.168.1.102 sending a packet to Supernode.nc.rr.com. The second packet is a "push" packet and an acknowledgement of the first packet sent by the Peer. The third packet shows the Peer acknowledging the "push" packet from the Supernode and the keep-alive is completed. In the event that the Supernode would not respond, the KaZaA Peer would establish a connection with another Supernode on his list provided by KaZaA in the start up communication.

```
08:44:54.085475 192.168.1.102.2826 > Supernode.nc.rr.com.1214: P [tcp sum ok]
211174616:211174617(1) ack 1351745324 win 8743 (DF) (ttl 128, id 40241,
len 41)
```


© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
0x0000   4500 0029 9d31 4000 8006 884e c0a8 0166        E..).1@....N...f
0x0010   xxxx xxxx 0b0a 04be 0c96 44d8 5091 ff2c        .X........D.P..,
0x0020   5018 2227 cf60 0000 38                         P."'.`..8
```

```
08:44:54.141125 Supernode.nc.rr.com.1214 > 192.168.1.102.2826: P [tcp sum ok]
1:2(1) ack 1 win 64239 (DF) (ttl 115, id 6673, len 41)
0x0000   4500 0029 1a11 4000 7306 186f xxxx xxxx        E..)..@.s..o.X..
0x0010   c0a8 0166 04be 0b0a 5091 ff2c 0c96 44d9        ...f....P..,..D.
0x0020   5018 faef e996 0000 45e0 25c8 6eee             P.......E.%.n.
```

```
08:44:54.337880 192.168.1.102.2826 > Supernode.nc.rr.com.1214: . [tcp sum ok]
ack 2 win 8742 (DF) (ttl 128, id 40497, len 40)
0x0000   4500 0028 9e31 4000 8006 874f c0a8 0166        E..(.1@....O...f
0x0010   xxxx xxxx 0b0a 04be 0c96 44d9 5091 ff2d        .X........D.P..-
0x0020   5010 2226 0769 0000                            P."&.i..
```

Next, we will look at an example of a search. On the KaZaA Media Desktop, the user can search for a string and the Supernode will return a list of files containing that string. KaZaA will match that string to the filename, extension, information in the description, file size, user, etc. The search option is a very versatile. Whatever it can find with that string in someone's share folders, the user can download.

Here is the dump of the search request. Since the Peer already has an active connection with the Supernode, only a "push" packet is sent with the string the users wishes to search.

The first packet in the example below is the "push" packet containing the string sent from the Peer, 192.168.1.102 to the Supernode.nc.rr.com on TCP port 1214. The second packet is the Supernode acknowledging the request and sending any information he might have in his index in a "push" packet. The information would consist of filenames, usernames, file sizes, etc. All of the information he has about files that match that search request. The third packet shows the Peer, 192.168.1.102 acknowledging the receipt of the first chunk of data sent by the Supernode.

The communication continues with the Peer sending "ack" for each "push" packet the Supernode sends until the list of files the Supernode found are provided to the Peer. The last packet in the example is the Peer acknowledging the last chunk of data from the Supernode.

```
08:38:04.158653 192.168.1.102.2826 > Supernode.nc.rr.com.1214: P [tcp sum ok]
211160441:211160467(26) ack 1351733740 win 8760 (DF) (ttl 128, id 21550, len
66)
0x0000   4500 0042 542e 4000 8006 d138 c0a8 0166        E..BT.@....8...f
0x0010   xxxx xxxx 0b0a 04be 0c96 0d79 5091 d1ec        .X.........yP...
0x0020   5018 2238 9740 0000 e88f dba9 0066 fcce        P."8.@.......f..
0x0030   1bd9 0970 a9c6 4712 f837 8d62 1816 ffbd        ...p..G..7.b....
0x0040   5f96                                           _.
```

```
08:38:04.261472 Supernode.nc.rr.com.1214 > 192.168.1.102.2826: P [tcp sum ok]
1:366(365) ack 26 win 64144 (DF) (ttl 115, id 33194, len 405)
0x0000   4500 0195 81aa 4000 7306 af69 xxxx xxxx        E.....@.s..i.X..
```


© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
0x0010   c0a8 0166 04be 0b0a 5091 d1ec 0c96 0d93        ...f....P.......
0x0020   5018 fa90 418d 0000 cd3c 06e5 52c5 b264        P...A....<..R..d
0x0030   deea ca23 ac09 d65f fc5d 1cd4 f0ed 2cca        ...#..._.]....,.
```

```
08:38:04.396542 192.168.1.102.2826 > Supernode.nc.rr.com.1214: . [tcp sum ok]
ack 366 win 8395 (DF) (ttl 128, id 21806, len 40)
0x0000   4500 0028 552e 4000 8006 d052 c0a8 0166        E..(U.@....R...f
0x0010   xxxx xxxx 0b0a 04be 0c96 0d93 5091 d359        .X..........P..Y
0x0020   5010 20cb 6bde 0000                            P...k...
```

```
08:38:04.473440 Supernode.nc.rr.com.1214 > 192.168.1.102.2826: P [tcp sum ok]
366:574(208) ack 26 win 64144 (DF) (ttl 115, id 33212, len 248)
0x0000   4500 00f8 81bc 4000 7306 aff4 xxxx xxxx        E.....@.s....X..
0x0010   c0a8 0166 04be 0b0a 5091 d359 0c96 0d93        ...f....P..Y....
0x0020   5018 fa90 c7b5 0000 8f6a d955 7db1 a802        P........j.U}...
```

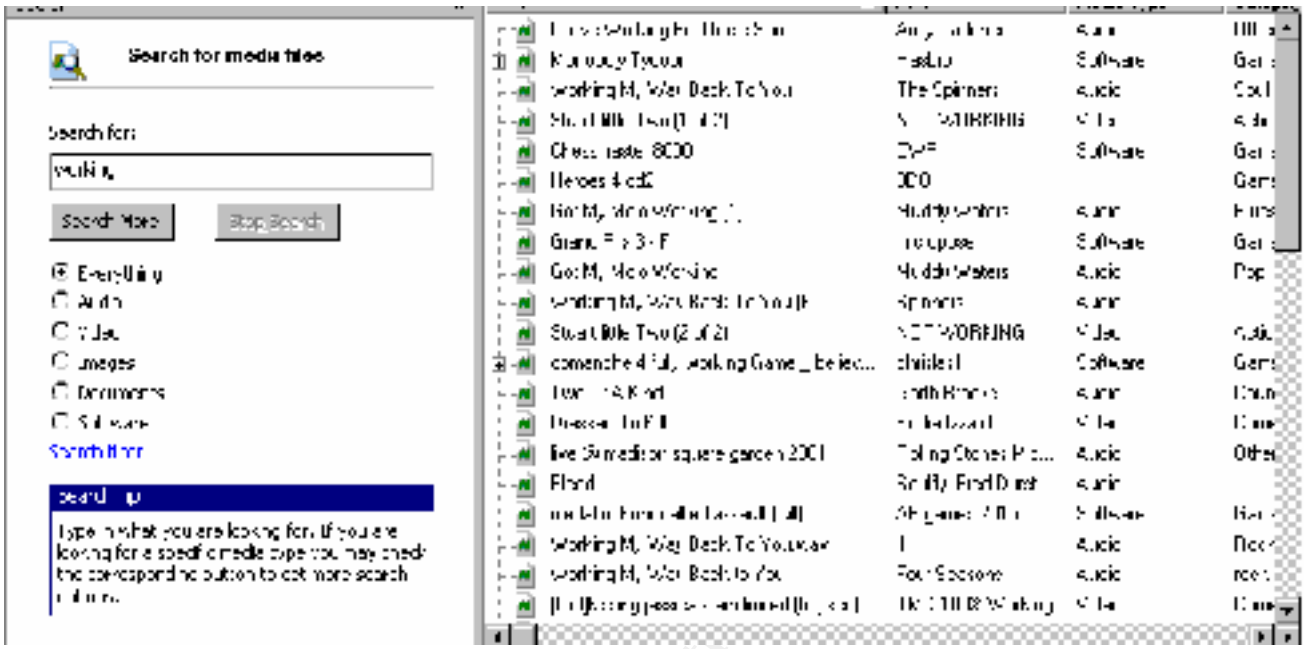
```
08:38:04.496213 192.168.1.102.2826 > Supernode.nc.rr.com.1214: . 26:1486(1460)
ack 574 win 8187 (DF) (ttl 128, id 22062, len 1500)
0x0000   4500 05dc 562e 4000 8006 c99e c0a8 0166        E...V.@........f
0x0010   xxxx xxxx 0b0a 04be 0c96 0d93 5091 d429        .X..........P..)
0x0020   5010 1ffb 774b 0000 d151 b109 23ff e0a8        P...wK...Q..#...
```

```
08:38:04.496355 192.168.1.102.2826 > Supernode.nc.rr.com.1214: .
1486:2946(1460) ack 574 win 8187 (DF) (ttl 128, id 22318, len 1500)
0x0000   4500 05dc 572e 4000 8006 c89e c0a8 0166        E...W.@........f
0x0010   xxxx xxxx 0b0a 04be 0c96 1347 5091 d429        .X.........GP..)
0x0020   5010 1ffb 5756 0000 42e2 fcc3 12ab 69ad        P...WV..B.....i.
```

Suppose the user searched for the word "working". The Supernode checks his index and returns a list of KaZaA users that are sharing files with the word "working", or it will send the request on to another Supernode to see if he has any entries that match the search request. Below is a screen print of the results of this search. In the screen print below, not all of the filenames have the word working. The word "working" must be in the name of the artist, or in the description. For example, look towards the top at Stuart Little Two (1 of 2), over in the next column, under artist; it says, "NOT WORKING". This explains why that particular file came up in the search request.

© SANS Institute 2000 - 2002 Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

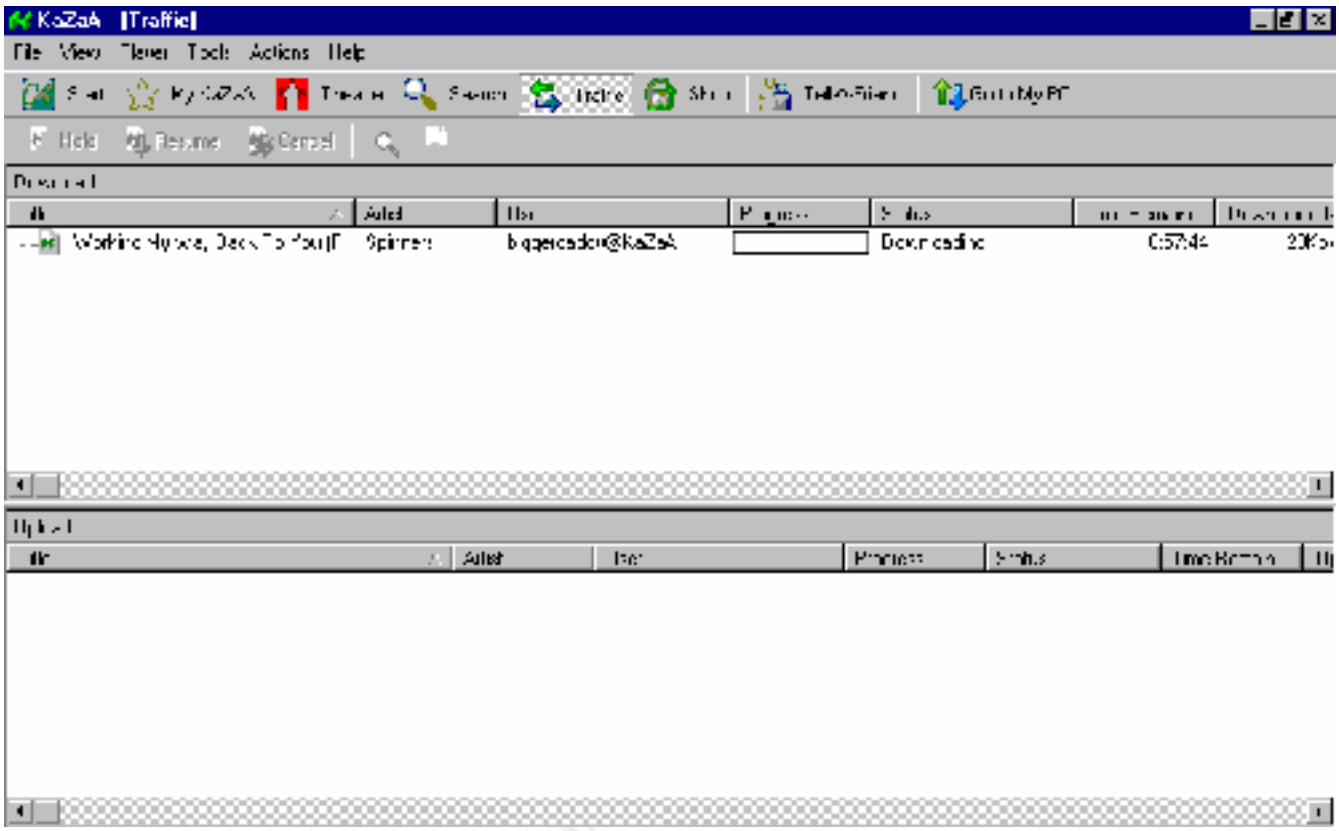


The user can double click on an entry that he wishes to download and the KaZaA media desktop attempts to contact that user to get the file download started. If he cannot make contact with that user, it will try another user that returned the same entry for the search. If KaZaA can make contact with more than one user offering the same file name, it will perform a multi-segmented download. That means it uses the two or more users that offer the same file and downloads the file from all of them simultaneously to ensure the quickest download time. In the example, the use of the + and – signs shows that there is more than one entry for the same file. Multi-segmented download is another reason the FastTrack network is popular, the quicker the download the better.

As soon as the user double clicks on the entry he wants to download, the download begins. The progress can be viewed on the "traffic" window.

The traffic viewer shows the bytes being transferred and whom (username) the file is coming from. In this example, the username is "biggerdaddy".

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.



The window labeled "Upload" at the bottom of the screen would show any downloads being performed from this users share folder. However, if you do not happen to look at the "traffic" window while online and close KaZaA, the entries will be cleared when KaZaA is reopened. In other words, the "Upload" log clears each time KaZaA is closed, and there would be no way of seeing who downloaded a file from your machine after the fact.

Below is an example of the download negotiation. These first three packets show the typical TCP handshake. The KaZaA user, 192.168.1.102 sends a "Sync" packet when he decides what file to download and double click on the file name.

```
08:41:15.205668 192.168.1.102.3066 > peer1.rochester.rr.com.1214: S [tcp sum
ok] 211385232:211385232(0) win 8192 <mss 1460,nop,nop,sackOK> (DF) (ttl 128, id
31535, len 48)
0x0000   4500 0030 7b2f 4000 8006 c492 c0a8 0166        E..0{/@........f
0x0010   xxxx xxxx 0bfa 04be 0c99 7b90 0000 0000        BB........{.....
0x0020   7002 2000 0f38 0000 0204 05b4 0101 0402        p....8..........
```

```
08:41:17.370064 peer1.rochester.rr.com.1214 > 192.168.1.102.3066: S [tcp sum
ok] 2698916057:2698916057(0) ack 211385233 win 17520 <mss 1460,nop,nop,sackOK>
(DF) (ttl 112, id 58858, len 48)
0x0000   4500 0030 e5ea 4000 7006 69d7 xxxx xxxx        E..0..@.p.i.BB..
0x0010   c0a8 0166 04be 0bfa a0de 30d9 0c99 7b91        ...f......0...{.
0x0020   7012 4470 18ff 0000 0204 05b4 0101 0402        p.Dp............
```

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
08:41:17.370272 192.168.1.102.3066 > peer1.rochester.rr.com.1214: . [tcp sum
ok] ack 1 win 8760 (DF) (ttl 128, id 31791, len 40)
0x0000   4500 0028 7c2f 4000 8006 c39a c0a8 0166        E..(|/@........f
0x0010   xxxx xxxx 0bfa 04be 0c99 7b91 a0de 30da        BB........{...0.
0x0020   5010 2238 67fb 0000                            P."8g...
```

Next, the requesting user sends information about himself to the other peer offering the file. The first thing we can identify is the "GET" command. This is telling the Peer providing the file what we want to download from their share folder. In this example, it is an mpeg of a portion of a soccer match between Italy and France. Below that in bold is the Host IP of the user that has the file we are asking to download, 10.10.100.100. The requesting users name is lancemanion. Below that in bold is the Kazaa-IP of 192.168.1.102. This is the IP address of the Peer. The user also provides the IP address of the Supernode we used to find the file, 10.11.111.111.

```
08:41:17.371160 192.168.1.102.3066 > peer1.rochester.rr.com.1214: P [tcp sum
ok] 1:323(322) ack 1 win 8760 (DF) (ttl 128, id 32047, len 362)
0x0000   4500 016a 7d2f 4000 8006 c158 c0a8 0166        E..j}/@....X...f
0x0010   xxxx xxxx 0bfa 04be 0c99 7b91 a0de 30da        BB........{...0.
0x0020   5018 2238 b542 0000 4745 5420 2f31 3131        P."8.B..GET./111
0x0030   3536 2f45 5552 4f25 3230 3230 3030 2532        56/EURO%202000%2
0x0040   3049 7461 6c79 2d46 7261 6e63 6525 3230        0Italy-France%20
0x0050   312d 3025 3230 2532 3844 656c 7665 6368        1-0%20%28Delvech
0x0060   696f 2532 392e 6d70 6567 2048 5454 502f        io%29.mpeg.HTTP/
0x0070   312e 310d 0a48 6f73 743a 20xx xxxx xxxx        1.1..Host:.10.10
0x0080   xxxx xxxx xxxx xxxx 3a31 3231 340d 0a55        .100.100:1214..U
0x0090   7365 7241 6765 6e74 3a20 4b61 7a61 6143        serAgent:.KazaaC
0x00a0   6c69 656e 7420 4d61 7920 3238 2032 3030        lient.May.28.200
0x00b0   3220 3030 3a32 333a 3532 0d0a 582d 4b61        2.00:23:52..X-Ka
0x00c0   7a61 612d 5573 6572 6e61 6d65 3a20 xxxx        zaa-Username:.la
0x00d0   xxxx xxxx xxxx xxxx xx0d 0a58 2d4b 617a        ncemanion..X-Kaz
0x00e0   6161 2d4e 6574 776f 726b 3a20 4b61 5a61        aa-Network:.KaZa
0x00f0   410d 0a58 2d4b 617a 6161 2d49 503a 2031        A..X-Kazaa-IP:.1
0x0100   3932 2e31 3638 2e31 2e31 3032 3a31 3231        92.168.1.102:121
0x0110   340d 0a58 2d4b 617a 6161 2d53 7570 6572        4..X-Kazaa-Super
0x0120   6e6f 6465 4950 3a20 xxxx xxxx xxxx xxxx        nodeIP:.10.11.11
0x0130   xxxx xxxx xx3a 3132 3134 0d0a 436f 6e6e        1.111:1214..Conn
0x0140   6563 7469 6f6e 3a20 636c 6f73 650d 0a58        ection:.close..X
0x0150   2d4b 617a 6161 2d58 6665 7249 643a 2039        -Kazaa-XferId:.9
0x0160   3133 3332 3831 0d0a 0d0a                       133281....
```

After the requesting Peer sends his information, the Peer offering the file responds. In this example, the offering Peer sends his IP address (10.10.100.100) and his user name of "mestuffisyoustuf". We also see the offering Peer's Supernode address of 10.11.11.11. Below that is information about the file being requested.

```
08:41:18.721560 peer1.rochester.rr.com.1214 > 192.168.1.102.3066: .
1:1461(1460) ack 323 win 17198 (DF) (ttl 112, id 58903, len 1500)
0x0000   4500 05dc e617 4000 7006 63fe xxxx xxxx        E.....@.p.c.BB..
0x0010   c0a8 0166 04be 0bfa a0de 30da 0c99 7cd3        ...f......0...|.
0x0020   5010 432e 9069 0000 4854 5450 2f31 2e31        P.C..i..HTTP/1.1
0x0030   2032 3030 204f 4b0d 0a43 6f6e 7465 6e74        .200.OK..Content
0x0040   2d4c 656e 6774 683a 2037 3938 3732 340d        -Length:.798724.
0x0050   0a41 6363 6570 742d 5261 6e67 6573 3a20        .Accept-Ranges:.
0x0060   6279 7465 730d 0a44 6174 653a 2054 6875        bytes..Date:.Thu
0x0070   2c20 3031 2041 7567 2032 3030 3220 3131        ,.01.Aug.2002.11
```


© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
0x0080   3a34 383a 3031 2047 4d54 0d0a 5365 7276        :48:01.GMT..Serv
0x0090   6572 3a20 4b61 7a61 6143 6c69 656e 7420        er:.KazaaClient.
0x00a0   4d61 7920 3238 2032 3030 3220 3030 3a32        May.28.2002.00:2
0x00b0   333a 3532 0d0a 436f 6e6e 6563 7469 6f6e        3:52..Connection
0x00c0   3a20 636c 6f73 650d 0a4c 6173 742d 4d6f        :.close..Last-Mo
0x00d0   6469 6669 6564 3a20 5468 752c 2032 3720        dified:.Thu,.27.
0x00e0   4a75 6e20 3230 3032 2031 353a 3535 3a35        Jun.2002.15:55:5
0x00f0   3020 474d 540d 0a58 2d4b 617a 6161 2d55        0.GMT..X-Kazaa-U
0x0100   7365 726e 616d 653a 20xx xxxx xxxx xxxx        sername:.mestuff
0x0110   xxxx xxxx xxxx xxxx xx0d 0a58 2d4b 617a        isyoustuf..X-Kaz
0x0120   6161 2d4e 6574 776f 726b 3a20 4b61 5a61        aa-Network:.KaZa
0x0130   410d 0a58 2d4b 617a 6161 2d49 503a 203x        A..X-Kazaa-IP:.1
0x0140   xxxx xxxx xxxx xxxx xxxx xxxx 3a31 3231        0.10.100.100:121
0x0150   340d 0a58 2d4b 617a 6161 2d53 7570 6572        4..X-Kazaa-Super
0x0160   6e6f 6465 4950 3a20 xxxx xxxx xxxx xxxx        nodeIP:.10.11.11
0x0170   xxxx xx3a 3132 3134 0d0a 582d 4b61 7a61        .11:1214..X-Kaza
0x0180   6154 6167 3a20 3133 3d32 3838 0d0a 582d        aTag:.13=288..X-
0x0190   4b61 7a61 6154 6167 3a20 353d 390d 0a58        KazaaTag:.5=9..X
0x01a0   2d4b 617a 6161 5461 673a 2032 313d 3635        -KazaaTag:.21=65
0x01b0   320d 0a58 2d4b 617a 6161 5461 673a 2034        2..X-KazaaTag:.4
0x01c0   3d53 6f63 6365 7220 2d20 4555 524f 2032        =Soccer.-.EURO.2
0x01d0   3030 3020 4974 616c 792d 4672 616e 6365        000.Italy-France
0x01e0   2031 2d30 2028 4465 6c76 6563 6869 6f29        .1-0.(Delvechio)
0x01f0   0d0a 582d 4b61 7a61 6154 6167 3a20 363d        ..X-KazaaTag:.6=
0x0200   466f 6f74 6261 6c6c 2047 7265 6174 2067        Football.Great.g
0x0210   6f61 6c73 0d0a 582d 4b61 7a61 6154 6167        oals..X-KazaaTag
0x0220   3a20 3134 3d53 706f 7274 730d 0a58 2d4b        :.14=Sports..X-K
0x0230   617a 6161 5461 673a 2032 363d 676f 616c        azaaTag:.26=goal
0x0240   732c 2067 6f6c 732c 2067 6f61 6c2c 2067        s,.gols,.goal,.g
0x0250   6f6c 0d0a 582d 4b61 7a61 6154 6167 3a20        ol..X-KazaaTag:.
0x0260   3138 3d56 6964 656f 2043 6c69 700d 0a58        18=Video.Clip..X
0x0270   2d4b 617a 6161 5461 673a 2033 3d3d 4d31        -KazaaTag:.3==M1
0x0280   386f 4a77 784e 735a 5267 3746 7156 5969        8oJwxNsZRg7FqVYi
0x0290   3134 5573 542f 6e55 493d 0d0a 436f 6e74        14UsT/nUI=..Cont
0x02a0   656e 742d 5479 7065 3a20 7669 6465 6f2f        ent-Type:.video/
0x02b0   6d70 6567 0d0a 0d0a 0000 01ba 2100 0100        mpeg........!...
0x02c0   1180 0d41 0000 01bb 0009 800d 4103 21ff        ...A........A.!.
0x02d0   e0e0 2e00 0001 be07 dfff ffff ffff ffff        ................
0x02e0   ffff ffff 0fff ffff ffff ffff ffff ffff        ................
```

After the negotiation and the file is identified, the sending peer begins to send the file as shown in the sample below using TCP port 1214. The first two packets show the sending Peer, peer1.rochester.rr.com sending 1460 bytes of data and then a second packet with another 1460 bytes of data. The third packet is the receiving Peer, 192.168.1.102 acknowledging that he has received the data up to byte 5841 (ack 5841). Two packets of data where sent and one acknowledgement confirming that the bytes that have been sent have been received thus far.

Fourth packet shows another 1460 bytes of data being sent and it continues in this manner until the transfer is completed. They would then tear down the connection gracefully with "fin" and "finack" as in a normal TCP connection.

```
08:41:19.709100 peer1.rochester.rr.com.1214 > 192.168.1.102.3066: .
2921:4381(1460) ack 323 win 17198 (DF)
```

```
08:41:19.755456 peer1.rochester.rr.com.1214 > 192.168.1.102.3066: .
4381:5841(1460) ack 323 win 17198 (DF)
```

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

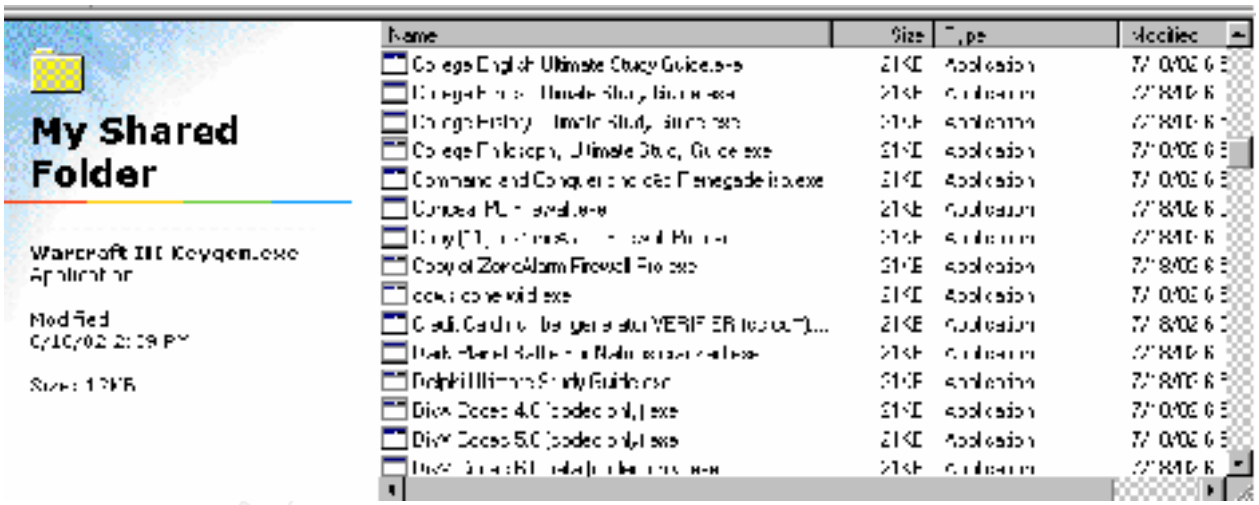
```
08:41:19.755616 192.168.1.102.3066 > peer1.rochester.rr.com.1214: . ack 5841
win 8760 (DF)
```

```
08:41:19.766096 peer1.rochester.rr.com.1214 > 192.168.1.102.3066: .
5841:7301(1460) ack 323 win 17198 (DF)
```

A special feature the KaZaA client offers that others do not is the resume feature. If for some reason a download was not able to complete, when KaZaA Media Desktop is started it tries to complete or initiate any previous unsuccessful downloads the last time the user was online.

How the exploit works

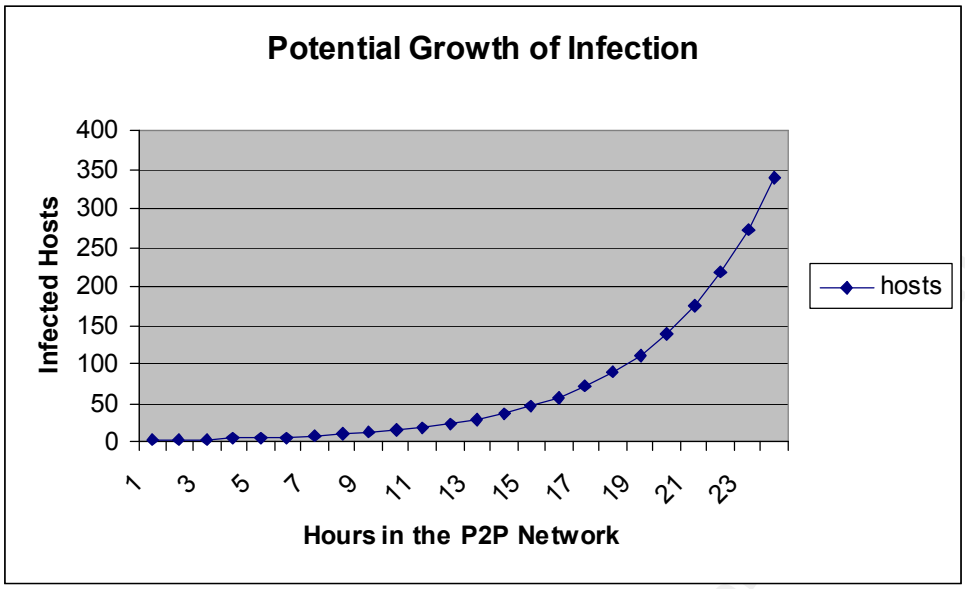
Once you download the executable file, you must try to open it in order to get the code to execute. It will replicate itself 150+ times in the shared folder with different names and subjects that might appeal to everyone or come up in several string searches. Below is an example of the files that were created by the virus that contain the virus. Every file is 21 bytes in size. This should stand out to the user attempting to download a movie or even College History Ultimate Study Guide; they should be larger than 21KB. The average KaZaA user probably doesn't pay attention to the file size or the file extension (i.e. exe).



With the big variety of filenames the virus uses, it could easily spread. For this paper, a box was infected with the virus, and was online for less than 15 minutes to capture some logs and someone downloaded the virus from the test machine before is could be cancelled or taken offline.

If someone downloaded the virus from the test box in a 15-minute timeframe, the virus could spread rapidly. Realistically, not all infected machines will give the virus to another machine every 15 minutes. Below the graph is based on ¼ of the infected machines giving the virus to another machine, how quickly the virus can grow. Starting with one machine infecting another and so on, by the end of 24 hours theoretically 339 machines would be infected.

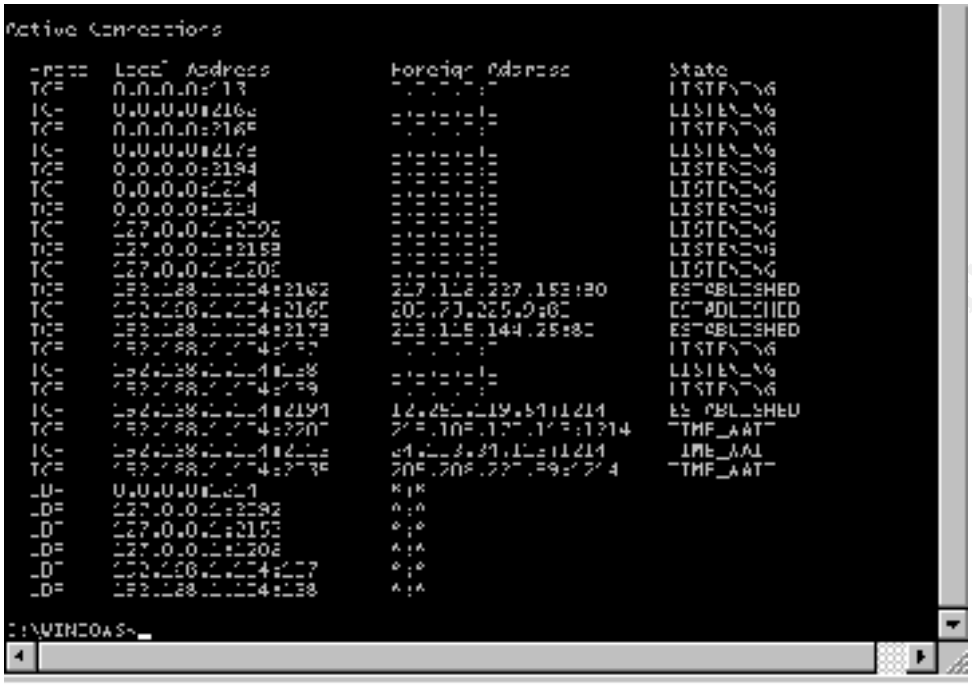
© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.



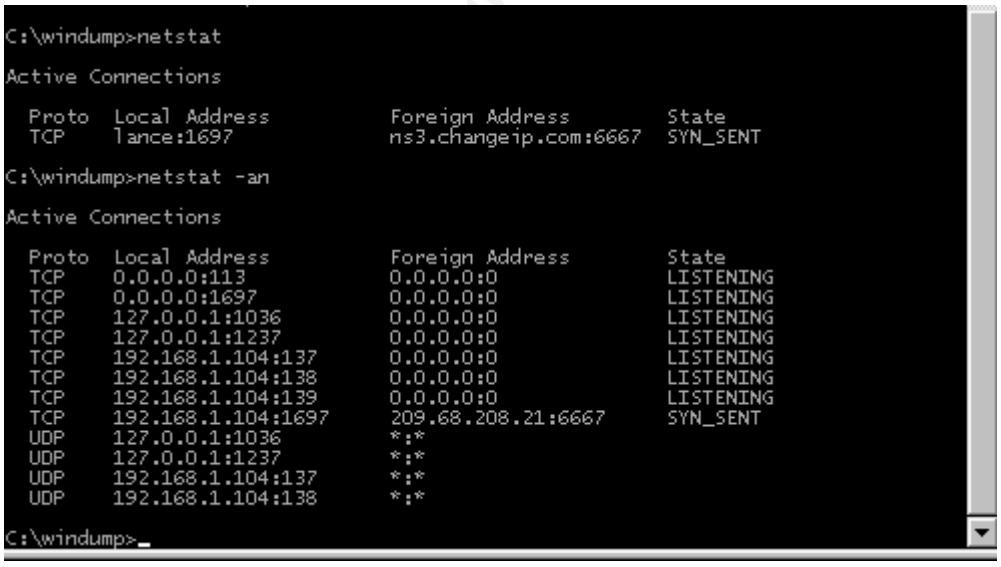
There were reports that the virus opened up an IRC channel listening for commands from the hacker. At first, there was no indication that this was true. Running Windump on the Windows 98 box that was running KaZaA media desktop and capturing packets with Snort at the perimeter, the virus was executed. After analyzing the captures, there was not indication of an IRC Channel. There was no unusual traffic captured. Here is the netstat -an on the infected box and could not see where port 6667 was listening.

It is listening on TCP port 1214 because KaZaA media desktop is running. When an application is running that accepts inbound connection, the port will show as "LISTENING". The windows Netbios ports (137, 138, and 139) show as listening as they do on all windows operating systems unless specifically turned off. The established connections shown here are all on port 80, one would be the KaZaA server and the other two are probably advertisements. Towards the top, you see some ports listening that have no address in the "foreign address" column. Four of those, 2162, 2165, 2178, and 2194 correspond with the connections below that are established or waiting. The communication is ongoing or closing and the computer leaves that port open and listening for return traffic on that port until the connection is closed or times out.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.



The next time the box was infected, it was left connected to the Internet for approximately ½ hour or more and the test box started sending "Syn" packets on port 6667. Here is the screen print of the netstat and a netstat –an. The infected box is not actually listening on port 6667; it is sending a periodic "Sync" packet to the address shown in the screen print below. The "Sync" packets continued without a response.



Below are the packets captures of that traffic. We are only showing a small portion of the packet to clearly identify the interval at which the "Sync" packets are sent. Notice that the source port changes after five tries. In the first section using port 3518, we see that the second attempt is after 3 seconds, then the third 6 seconds from that one and the fourth 12 seconds later, then 26 seconds later the pattern starts again. This is normal for a

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

TCP connection. It will try four times to send a "Syn" packet and if it doesn't receive a "Synack" it will timeout and stop trying or in this case, it will attempt again 26 seconds later. When it tries again after five tries, the source port changes because it is a new attempt and the source port is chosen at random by the OS of the client. In this case, our test box is the client.

So by identifying the intervals are normal, we can conclude that this traffic is not a response to a stimulus. In other words, the attacker is not actively prompting the box to make these attempts, the virus is instructing it to do so.

18:43:39.770546 192.168.1.104.3518 > ns3.changeip.com.ircd: S 109949109:109949109(0)
18:43:42.761991 192.168.1.104.3518 > ns3.changeip.com.ircd: S 109949109:109949109(0)
18:43:48.761780 192.168.1.104.3518 > ns3.changeip.com.ircd: S 109949109:109949109(0)
18:44:00.771293 192.168.1.104.3518 > ns3.changeip.com.ircd: S 109949109:109949109(0)

18:44:26.793778 192.168.1.104.3519 > ns3.changeip.com.ircd: S 109996134:109996134(0)
18:44:29.785178 192.168.1.104.3519 > ns3.changeip.com.ircd: S 109996134:109996134(0)
18:44:35.784957 192.168.1.104.3519 > ns3.changeip.com.ircd: S 109996134:109996134(0)
18:44:47.784475 192.168.1.104.3519 > ns3.changeip.com.ircd: S 109996134:109996134(0)

18:45:13.792038 192.168.1.104.3520 > ns3.changeip.com.ircd: S 110043134:110043134(0)
18:45:16.783366 192.168.1.104.3520 > ns3.changeip.com.ircd: S 110043134:110043134(0)
18:45:22.783147 192.168.1.104.3520 > ns3.changeip.com.ircd: S 110043134:110043134(0)
18:45:34.782664 192.168.1.104.3520 > ns3.changeip.com.ircd: S 110043134:110043134(0)

18:46:00.802307 192.168.1.104.3521 > ns3.changeip.com.ircd: S 110090146:110090146(0)
18:46:03.781579 192.168.1.104.3521 > ns3.changeip.com.ircd: S 110090146:110090146(0)

The infected box may be trying to initiate a connection with this machine in order to supply information about itself to the hacker. One piece of information that is evident is the listening port. In the screen print below, the "Syn" attempt is using that the source port is 1697. The infected box listens on that port for the "Synack" or "Reset" response.

```
C:\windump>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    lance:1697             ns3.changeip.com:6667  SYN_SENT

C:\windump>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:113            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1697           0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1036         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1237         0.0.0.0:0              LISTENING
  TCP    192.168.1.104:137      0.0.0.0:0              LISTENING
  TCP    192.168.1.104:138      0.0.0.0:0              LISTENING
  TCP    192.168.1.104:139      0.0.0.0:0              LISTENING
  TCP    192.168.1.104:1697     209.68.208.21:6667     SYN_SENT
  UDP    127.0.0.1:1036         *:*
  UDP    127.0.0.1:1237         *:*
  UDP    192.168.1.104:137      *:*
  UDP    192.168.1.104:138      *:*

C:\windump>_
```


© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

If the destination box had responded with a "Synack" and the handshake was allowed to continue, the attacker would have an open connection to the infected box. Below is the Windump output for those connection, there is no "Synack" or "Reset" received from 209.68.208.21.

```
16:32:38.385429 192.168.1.104.1714 > 209.68.208.21.ircd: S [tcp sum ok]
15673505:15673505(0) win 8192 <mss 1460,nop,nop,sackOK> (DF)
(ttl 128, id 22542, len 48)
0x0000  4500 0030 580e 4000 8006 3f4f c0a8 0168      E..0X.@...?O...h
0x0010  d144 d015 06b2 1a0b 00ef 28a1 0000 0000      .D........(.....
0x0020  7002 2000 b567 0000 0204 05b4 0101 0402      p....g..........
-------------------------------------------------------------
16:32:41.380202 192.168.1.104.1714 > 209.68.208.21.ircd: S [tcp sum ok]
15673505:15673505(0) win 8192 <mss 1460,nop,nop,sackOK> (DF)
(ttl 128, id 22798, len 48)
0x0000  4500 0030 590e 4000 8006 3e4f c0a8 0168      E..0Y.@...>O...h
0x0010  d144 d015 06b2 1a0b 00ef 28a1 0000 0000      .D........(.....
0x0020  7002 2000 b567 0000 0204 05b4 0101 0402      p....g..........
-------------------------------------------------------------
16:32:47.379998 192.168.1.104.1714 > 209.68.208.21.ircd: [tcp sum ok]
15673505:15673505(0) win 8192 <mss 1460,nop,nop,sackOK> (DF)
(ttl 128, id 23566, len 48)
0x0000  4500 0030 5c0e 4000 8006 3b4f c0a8 0168      E..0\.@...;O...h
0x0010  d144 d015 06b2 1a0b 00ef 28a1 0000 0000      .D........(.....
0x0020  7002 2000 b567 0000 0204 05b4 0101 0402      p....g..........
```

Continuous "Syn" packets with no response can be caused by several things, the box is down, the box cannot respond, or the test server cannot reach the box. If the box was up and had this port open, we would receive a "Synack". If the box was up and didn't have the port open, it would send a "Reset". Could be the attacker wants the machine, 209.68.208.21, to accept the "Syn" packets, log them, and not respond causing the infected machine to keep trying and in the process keeps a port or the channel opened for the attacker when he is ready.

Below is another nmap scan to see if port 1697 is open for communication from everyone. The results show that port 1697 does not appear to be open but is listening according to netstat. This means that it is open to 209.68.208.21 for the "Synack" that TCP requires to complete the handshake.

```
Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.1.102):
(The 1540 ports scanned but not shown below are in state: closed)
Port       State       Service
113/tcp    open        auth
139/tcp    open        netbios-ssn


Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

When the virus was executed it generated an error message, "Cannot import c:\moo.reg: The specified file is not a registry script. You can import only registry files." This file, according to Bitdefender at

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

http://www.bitdefender.com/virusi/virusi_descrieri.php?virus_id=87, contains a list of subfolders in the Program Files folder. It appears that this file is not successfully created.

With an open IRC channel, the attacker would be able to take control of the machine and initiate commands. Here are the actions that could be performed through the IRC channel as reported by Symantec at http://www.sarc.com/avcenter/venc/data/pf/w32.kwbot.worm.html

[6]

- Manage the installation of the backdoor
- Control the IRC client on the compromised computer
- Dynamically update the installed Trojan
- Send the Trojan to other IRC channels to attempt to compromise more computers
- Download and execute files
- Deliver system and network information to the hacker
- Perform Denial of Service (DoS) attacks against a target that is defined by the hacker
- Uninstall itself completely by removing the relevant registry entries.

In addition to replicating itself in the shared folder for others to download, it creates a file called explorer32.exe in the \windows\system directory.

It creates two entries in the registry in order to execute the file at windows start up.

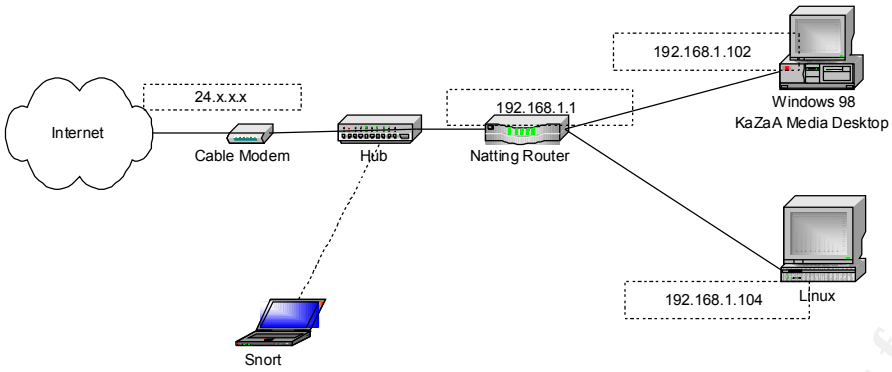
HLM\Software\Microsoft\Windows\CurrentVersion\Run\Windows Explorer Update Build 1142=explorer32.exe
HLM\Software\Microsoft\Windows\CurrentVersion\RunServices\ Explorer Update Build 1142=explorer32.exe

Diagram

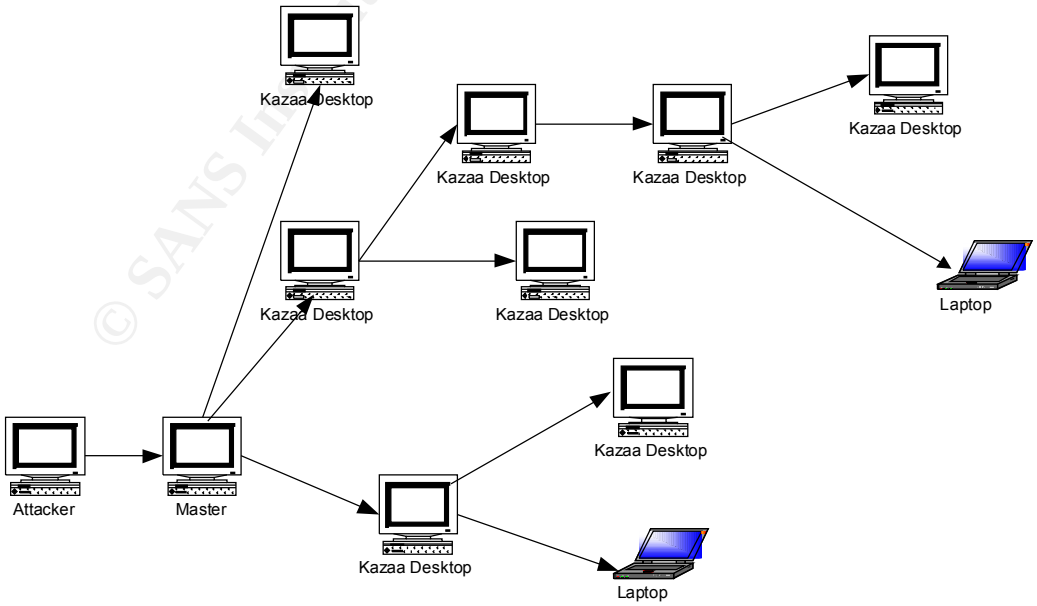
Below is a basic network diagram of the test environment. The natting router is set up to allow all TCP and UDP ports inbound to the 192.168.1.102 Windows 98 box. All outbound traffic is permitted. The Linux box was used for the nmap scans. Windump and Ethereal was used to capture traffic to and from the on the Windows 98 box during each test. The laptop was running Snort at the perimeter to capture all traffic to and from the network during testing and to test the Snort IDS rules.

[6] http://www.sarc.com/avcenter/venc/data/pf/w32.kwbot.worm.html

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.



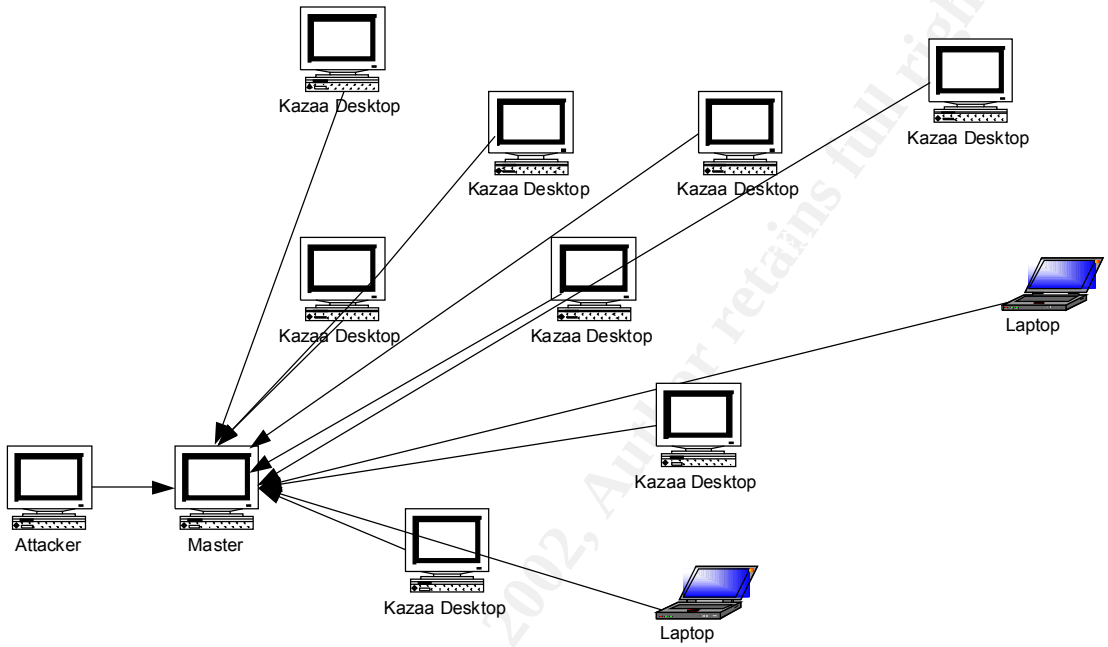
The following diagram shows the Attacker putting the virus on a host that he compromised in order to conceal his identity when the virus starts to propagate. The host marked "master" could be the host that the test box was attempting to contact on port 6667 in the previous captures. The virus would be waiting in the share folder to be downloaded by other KaZaA users and so on.



© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

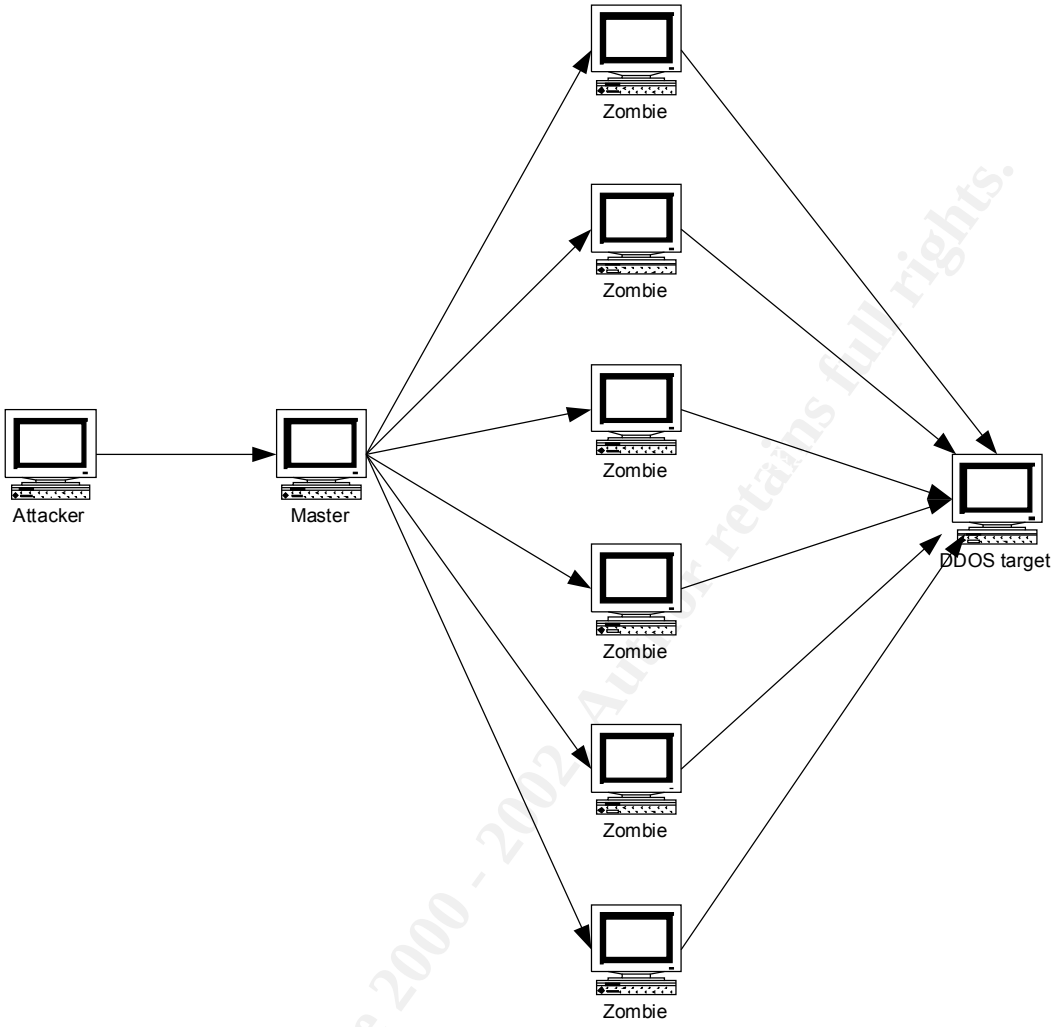
This diagram shows what was probably intended, to send information back to the master about each system and open an IRC Channel for the attacker to use when he has control of enough hosts.



Once in control a large number of hosts, the attacker could launch a Distributed Denial of Service (DDOS) attacker on a targeted machine. The "zombies" would be instructed by the attacker via the master, what the target host is and what type of Denial of Service attack to initiate.

The diagram below is a typical DDOS. If the virus with the backdoor was not used and propagated using KaZaA media desktop, the virus could still have been propagated through e-mail, other file-sharing systems that do not restrict by file type, or installed by the attacker through some other exploitable vulnerability on the host. Once the host has been compromised and the attacker can control the machine, he can open up an IRC Channel for accepting commands and/or load a "Trojan" that can be triggered through this channel, or other means, and effectively initiate a DDOS.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.



How to use the exploit

The attacker could easily gain control of many machines using the IRC Channel. Machines that belong to users that do not concern themselves with security, do not have intrusion detection devices running at the perimeter, do not have firewalls protecting them, and may not notice the virus.

[7]"Any attacker who can control 100,000 machines is a major force on the internet, while someone with a million or more is currently unstoppable: able to launch massively diffuse DDOS attacks, perform needle in a hayfield searches, and commit all sorts of other mayhem."

DDOS, Distributed Denial of Service, attacks using compromised workstation called "zombies". A popular "zombie" is a home computer on a cable modem or DSL connection to the Internet. Cable and DSL connection are rapidly increasing. A "virus" containing a "backdoor" is one way to compromise a number of hosts to use for a DDOS.

[7] http://www.cs.berkeley.edu/~nweaver/0wn2.html

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

A "backdoor" is a hidden way for a user to access a compromised system for malicious purposes or a programmer leaving a means to access a system for recovery or troubleshooting.

The reports stated that Kwbot would create a file and send it to the attacker or somewhere the attacker can retrieve it, with information about the compromised system. It would also open an IRC Channel, which would wait for the attacker to issue a command. The command could be to launch a denial of service attack on another host, maybe one of more interest or importance. For example, to take a firewall or intrusion detection device out of commission in order to secretly access bank records, trade secrets, school grades, or just to feel the accomplishment of taking down the targeted system or key network device.

First, the attacker would take the w32.kwbot virus and make it available for download. Obviously, he will want to keep his own identity secret so it cannot be traced back to him. Therefore, he would compromise a home users machine with KaZaA Media Desktop. This would be easy to find with a simple scan for port 1214. He makes the virus available, from this compromised machine, for download disguised with different filenames to appeal to a wide range of users and come up in many searches. Once downloaded by a certain number of users, patiently wait for the virus to propagate.

Users download the virus with the backdoor and they execute the file, thinking they are going to view an "A+ Certification Study Guide", porn, etc. Part of the malicious code sends a file to another compromised system that the attacker checks regularly. This would tell him that he had another system that he could possibly use for his bidding.

Once he gets enough systems to use for his "zombie", he can wake them up and run a malicious attack from the compromised hosts in unison at the targeted system.

Signature of the attack

There are several signatures easily detected on the host.

- It copies itself to the Windows\System32 directory as explorer32.exe. Normally there is no explorer.exe in that directory.

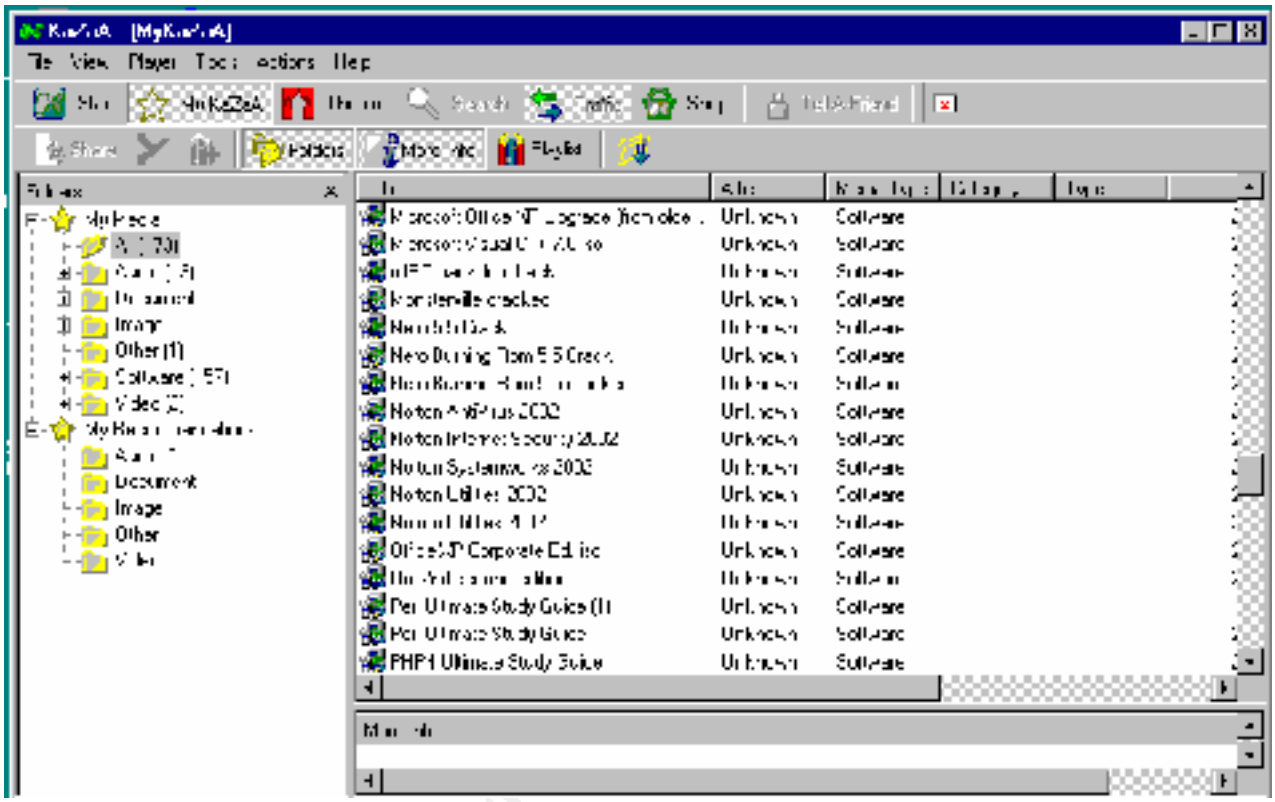
- It creates 2 registry key values to run explorer32.exe at startup:

HLM\Software\Microsoft\Windows\CurrentVersion\Run\Windows Explorer Update Build 1142=explorer32.exe
HLM\Software\Microsoft\Windows\CurrentVersion\RunServices\ Explorer Update Build 1142=explorer32.exe

- 150+ copies of the virus will be located in the "C:\Program Files\KaZaA\My Shared Folder" disguised as media files, software, and documents with popular subjects. For example porn, password crackers, study guides, hacker exploits, and current movies. Below is a snapshot of the less offensive filenames. A

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

complete list of files names are available at http://vil.mcafee.com/dispVirus.asp?virus_k=99555.



The network signature would be a little more difficult depending on the normal business policy for outbound connections. Some companies allow their users outbound access on any port to anywhere they only restrict inbound traffic. The Cisco PIX for example is designed for that type on environment. With that type of configuration, the users would be allowed to use chat, participate in P2P for file sharing, among many other things. IDS could be used to look for particular signatures of KaZaA or Morpheus in order to track an infection back to the source. For example, the test network was using Snort at the perimeter to capture packets. It could be set up to alert on certain signatures of attacks. You can get rules already written for known attack signatures at http://www.snort.org. Snort.org did not have a rule written specifically for this exploit but they did have one to alert if anyone tries to download a file using KaZaA or Morpheus client on the Fastrack P2P. The snort rule looks like this.

[8]alert tcp $EXTERNAL_NET any -> $HOME_NET 1214 (msg:"P2P Fastrack (kazaa/morpheus) traffic"; flow:to_server,established; content:"X-Kazaa-Username"; reference:url,www.kazaa.com; classtype:protocol-command-decode; sid:1699; rev:2;)

[8] http://www.snort.org/cgi-bin/sigs-search.cgi?sid=kazaa

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

It will alert when any external host on any source port tries to send a packet to any host on the internal network with a destination port of 1214. It also searches for "X-Kazaa-Username" in the payload. From the dumps in the "Protocol Description" section, anytime users requests to download a file, the payload would contain "X-Kazaa-Username" just before the Username itself. This is true for Kazaa and Morpheus, they both user the same keywords before the actual username.

Because this happens anytime a user downloads a file using Kazaa or Morpheus, this would alert if an internal user was getting a file from someone or if an external user was getting a file from someone internal. Both sending and receiving users send their Username.

Another possibility is to alert each time a user brings up the KaZaA Media desktop by changing the above rule slightly. In the section above, we show that the registration of the user happens on port 80 and we send a string with the user name similar to the rule above. The new rule would look like this.

alert tcp $HOME_NET any -> $EXTERNAL_NET 80 (msg:"Start Kazaa Media desktop)"; content:"GET"; depth: 4; content:"kmdstart.htm"; nocase; reference:url,www.kazaa.com; classtype:protocol-command-decode;

Here is a more specific rule from snort.com, that just alerts when an external users is requesting a download from one of the internal users. The difference is the destination port number, port 1214 is used for file transfers.

[9]alert tcp $EXTERNAL_NET any -> $HOME_NET 1214 (msg:"P2P Fastrack kazaa/morpheus) GET request"; flow:to_server,established; content:"GET "; depth:4; reference:url,www.musiccity.com/technology.htm; reference:url,www.kazaa.com; classtype:protocol-command-decode; sid:1383; rev:3;)

If the internal users are authorized to share files using KaZaA and Morpheus, then the rule above could generate too many alerts. We could take it one step further and only alert if it involves an executable file, which is the type of file containing this virus. Most times executable files are a risk to download, if you don't know what they are going to do when opened. Some companies restrict e-mail attachments that have the ".exe" extension. Here is a Snort rule that would alert just if our internal user downloads an executable file using KaZaA or Morpheus.

Alert tcp $HOME_NET any -> $EXTERNAL_NET 1214 (msg:"EXE download using kazaa") GET request"; content:"GET"; depth: 4; content "X-Kazaa-Username"; content ".exe"; nocase; reference:url,www.kazaa.com;

[9] http://www.snort.org/cgi-bin/sigs-search.cgi?sid=kazaa

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

This could easily be changed to alert even if an external users downloads an executable from an internal user. This might be something that the Incident Response Team wants to see, where infection might have come from and where it is has been spread.

Alert tcp any any -> any 1214 (msg:"EXE download using kazaa"; GET request"; content:"GET"; depth: 4; content:"X-Kazaa-Username"; content:".exe"; nocase; reference:url,www.kazaa.com;

This particular exploit does not have a snort rule available at snort.com, but there are rules available for IRC channels. They will look for ports 6666-7000. This exploit uses port 6667, so it might be wise to look specifically for that port so it can be labeled accordingly in the snort logs.

alert tcp $HOME_NET any -> $EXTERNAL_NET 6667 (msg:"IRC Channel for KWBOT virus")

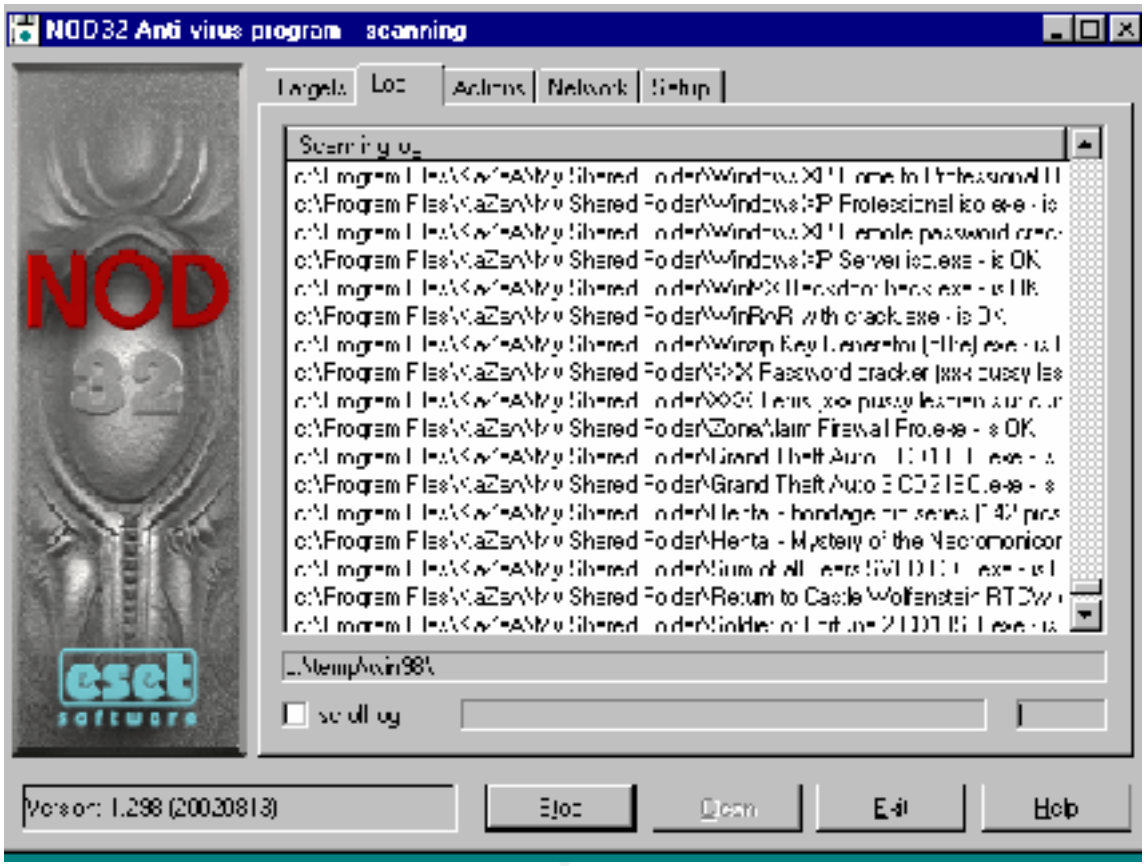
How to protect against it

We ran two different AntiVirus products on the infected machine, one found it and one did not. Neither could clean the virus.

Below is a screen print of the NOD32 AntiVirus scan output showing that it did not detect the virus. It marked some of the files that contained the virus in the share folder as "OK".

© SANS Institute 2000 - 2002 Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.



This screen print is from Symantec Norton AntiVirus software, identified the virus but was not able to clean it. It did give an option later to delete the files, which would have put a stop to it since it did detect the "explorer32.exe" file as well. However, the registry entries would still be there and cause an error at start up.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

The AntiVirus products would help identify that the virus existed, then the user could find out what they needed to do to clean out the registry. To protect against getting the virus infection would be simple, don't use KaZaA. That probably isn't the right answer, but that would sure help.

As KaZaA warns their users, downloading files from other users is a risk you are taking. They recommend keeping your AntiVirus software updated, but that doesn't really protect you from getting it. AntiVirus software companies only have the virus signatures to detect once the virus has been released into the outside world. Then it would be too late for some users.

The best suggestions for protecting against this virus for the home user:

- Do not download files with an .exe extension - If it is necessary to download an .exe file, do it with caution. Run it on an isolated workstation if possible or as mentioned below, make note of the file size.
- Be aware of the file size. If it supposed to be a movie, it is going to be larger than 21KB.
- Shutdown the computer when not in use. If the box is infected and an IRC Channel is open allowing the attacker to take control of the machine, it would be best to take the extra security step to shutdown the workstation to avoid being a non-willing participant in an attack.
- Use a personal firewall on the workstation or a firewall at the perimeter of the network with the following controls.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

    - Do not allow any inbound access. That would be anything initiated to your network or internal machine, if that is possible.
    - Use a personal firewall on the home workstation such as Zone Alarm, which restricts connections by application.
    - At least block IRC chat ports 6666-7000. In most business environment IRC chat and IM would not be allowed in and out of the network. As for home use and teenagers, it would be almost impossible to shut that down.
    - If possible, control the administrator access to the Windows desktop. Do not allow everyone to install software on the workstation.
- Use AntiVirus software and keep it up to date. Even though they can't guard against the virus until it has reached the outside world, most AntiVirus software companies will update their list of signatures as quickly as they can. It will at least detect it so it can be cleaned up before it is propagated or does any damage.

Most KaZaA users are home users sharing files for personal use, music, movies, software, etc. Not many companies would choose to use KaZaA to share files with other companies or customers on the Internet it would be too risky. Although there are those internal users that want to download something from a friend or get some music to listen to while working that might infect the network. Businesses should not allow these types of P2P network connections into their internal network. Here are some suggestions:

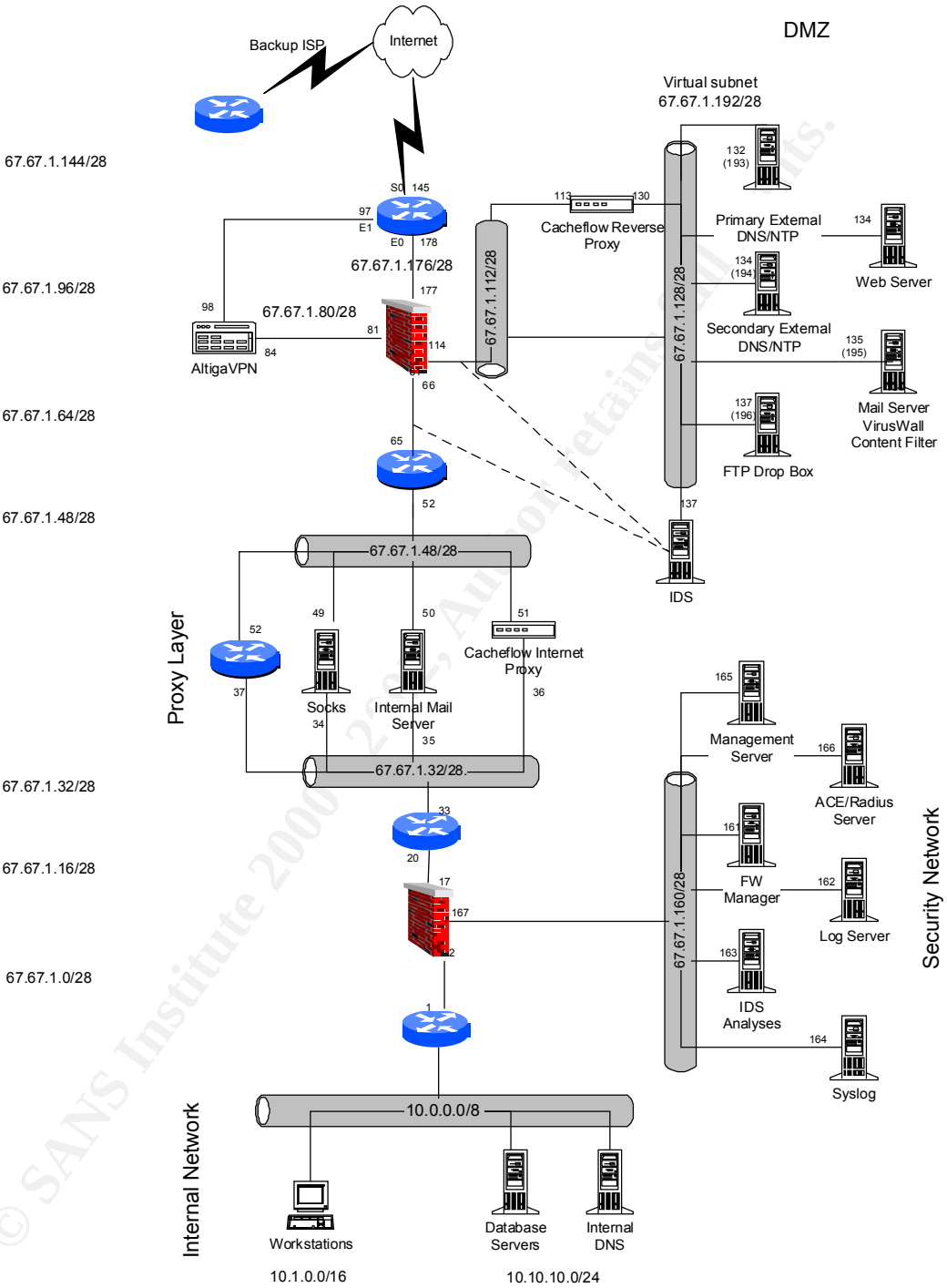
- Security Policy
    - Security Policy should be in place the clearly spells out that company computers should be used for business purposes only.
    - Only company-approved software should be installed on laptops or workstations. Sometimes locking down their workstations so they are unable to install software is not realistic. Therefore, it should be spelled out in the security policy.
    - No inbound traffic to workstations from Internet. This can be prevented with natting and firewalls.
    - Keep anti-virus software updated on workstations.
    - Consequences if you do not follow this policy.
    - Make sure the policy is available and all employees know it's there and sign something that they have read and will comply.

Firewalls and Security Network Design. A Secure network design could prevent the internal users from bringing in unwanted infections and intruders into the company network. Here is an example of a secure network design and some points about this design. This design may not be realistic for all companies, but there are some security features included that would discourage the infection of the internal network by the employees using IRC Chat, ftp, or P2P networks from the inside.

- More information can be found at the reference at footnote 8 and details about the configuration at http://www.giac.org/practical/Rita_Will_GCFW.zip.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

10



[10] Northcutt, Stephen; Zeltser, Lenny; Winters, Scott; Frederick, Karen Kent; Ritchey, Ronald W. Inside Network Perimeter Security. June 2002. Page 449 Case Study 4: A Complex E-Commerce Site

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

  - The Externally accessible servers are separated from the internal network. All Internet users are restricted to the DMZ. No traffic would be allowed from the Internet to the internal workstations, which is where the KaZaA Media Desktop would most likely be installed.
  - Use proxy servers for outbound access. An Internet proxy server can be restricted to outbound http and https only. The user would be able to get to the KaZaA site to download the software but wouldn't be able share files and retrieve files since KaZaA uses port 1214. They would however be able to connect to the KaZaA server, since we identified in the earlier section that is accomplished on http port 80.
  - The Proxy servers, e-mail relays, and the ftp servers could be configured to do content filtering. They could be restricting by file extension, possibly not allowing .exe files to be e-mailed into the company.
  - Firewalls should restrict inbound traffic to what is required for the business. That means just http and https to the web servers, ftp to the ftp servers, etc. All other ports should be blocked.
  - Block all outbound ports at the firewall that are not needed for business purposes. This would prevent the internal users from downloading files from other KaZaA users and allowing users to access their files.
  - The firewall or router should be hiding the internal network from the outside world. Users workstation and database servers should have private addressing and should not be provided to the Internet. If no proxy server, hide addresses should be used to prevent the private address of an internal workstation from being revealed.

- The best way is to restrict the internal users Internet access, if that isn't possible, then there are other issues to address. If they bring this virus into the internal network and it opens an IRC channel. It would be important to have IDS sensors located throughout the network, at least at the perimeter. The IDS sensors should be configured to watch for IRC ports 6666-6667. This would be the only signature of the virus that could be detected at the network. Snort rules are available for IRC detection or can be written to be specific to each exploit.
- If there is no firewall the router could be used to restrict access to particular ports. Block all inbound except to the specific ports to specific servers, for example port 80 to the web server and port 20/21 to the ftp servers. In this case, you might allow all outbound traffic, but just restrict particular ports such as the IRC ports 6666-6667 or all TCP and UDP high ports for that matter (anything above 1023). The router can be configured to allow those outbound connections using the high ports only if it is a response to an existing connection.
- Workstations and servers should have anti-virus protection and updated automatically.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Source code/Pseudo code

I took a copy of the executable file from the Windows/System directory, explorer32.exe and attempted to reverse engineer the code using Reverse Engineering Compiler (REC).

I ran REC with no special options since there is not much information about this code available. REC will attempt to disassemble it if it recognizes the executable. It didn't seem to have a problem; therefore, we let it use the default options. The following is the informational message provided by REC when running on explorer32.exe file. It appears that it recognized the executable and was able to generate code.

The results are provided at Appendix A. They really didn't give us anything useful.

```
explorer32.exe is an NT executable of 0x5400 (21504) bytes
Image base : 0x00400000,  Entry point : 0x0002a000
0x00025000 - 0x00029848  (  18504)
0x0002a000 - 0x0002a200  (    512)
0x0002a174 - 0x0002a1df  (    107)  Import table at offset 0x00000374
Validating strings...
Finding references...
```

I found a Win32 Disassemble program at http://www.geocities.com/~sangcho/ and ran this against the same executable. This time I got approximately 150 pages of assembler language. A sample of those results is provided at Appendix B.

I still wanted to try and get it to some kind of C code, so I took the output from the Win32 Disassemble program and ran it through Fenris. Didn't get the results I had hoped. Fenris generated many error messages and nothing that could be interrupted as C language.

NOTE: The reverse engineering results generated too many pages to attach, they can be provide upon request.

Additional Information

More information can be found at the following AntiVirus Software websites:
http://www.sarc.com/avcenter/venc/data/pf/w32.kwbot.worm.html
http://www.bitdefender.com/virusi/virusi_descrieri.php?virus_id=87
http://www.sophos.com.virusinfo/analyses/w32kwbota.hmtl
http://vil.mcafee.com/dispVirus.asp?birus_k=99555

The following are links to news articles about the virus:
http://www.techtv.com/news/print/0,23102,3385557,00.html
http://www.vnunet.com/News/1133129
http://resnet.albany.edu/news
http://www.theregister.co.uk/content/56/25945.html

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Appendixes

A

```
/*    This file was automatically created by
 *    Reverse Engineering Compiler 1.6 (C) Giampiero Caprino
      (Mar 31 2002)
 *    Input file: 'explorer32.exe'
 */

/*    Procedure: 0x0042A000 - 0x0042A006
 *    Argument size: 0
 *    Local size: 0
 *    Save regs size: 0
 */
__entry_point__()
{
    L0042A007();
    asm("Unknown opcode 0xfe");
    > ? L0042a066 : ;
}

/*    Procedure: 0x0042A007 - 0x0042A016
 *    Argument size: 0
 *    Local size: 0
 *    Save regs size: 0
 */

L0042A007()
{
    (restore)esi;
    goto L0042a00b;
    asm("das");
L0042a00b:
    (save)-534975669;
    L0042A017();
    asm("salc");
    goto L0042a077;
}

/*    Procedure: 0x0042A017 - 0x0042A028
 *    Argument size: 0
 *    Local size: 0
 *    Save regs size: 0
 */

L0042A017()
{
    (restore)edi;
    (restore)edx;
    goto L0042a01d;
    asm("int 0x20");
L0042a01d:
    ebx = 539342133;
    eax = L0042A029() + *ecx;
```

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
}
/*    Procedure: 0x0042A029 - 0x0042A03B
 *    Argument size: 0
 *    Local size: 0
 *    Save regs size: 0
 */
L0042A029()
{
    (restore)edi;
    ebx = ebx + edx;
    asm("adc edi,esi");
    eax = 244;
    edi = edi & ebx;
    L0042A03C();
    *(esi - 118) :: bl;
}
/* DEST BLOCK NOT FOUND: 0042a0dd -> 526f2796 */
/*    Procedure: 0x0042A03C - 0x0042A173
 *    Argument size: 11801
 *    Local size: 0
 *    Save regs size: 0
 */
L0042A03C()
{
    (restore)esi;
    cl = *ebx;
    edi = edi + 99 | 248;
    ecx = ecx ^ 1206001662;
    goto L0042a051;
    *(ebx + 32210921) = *(ebx + 32210921) + 1;
L0042a051:
    asm("xchg ebp,[edx]");
    asm("enter 0x2eb,0xcd");
    *(ecx + 843571953) = *(ecx + 843571953) & al;
    al :: 193;
    *ebx & 50;
    asm("enter 0xf90b,0x88");
    eax = eax | *(esi + 67);
    edi = edi - edx;
    eax = eax - 1;
    goto L0042a077;
    return;
    (fsave)(frestore) + *(ebx + -1133182744);
L0042a077:
    asm("ror esi,0xe");
    goto L0042a080;
    bh = 222;
L0042a080:
    asm("int 0x7c");
    eax :: 1971838591;
    edi = edi + 1;
    asm("Unknown opcode 0x8d");
    asm("loope 0x42a019");
    asm("aas");
    (save)ebx;
    asm("xchg eax,edi");
    esi = esi + 1;
```

SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
    asm("scasd");
    asm("xchg eax,ebx");
    asm("lahf");
    asm("popf");
    asm("lahf");
    asm("arpl [ebp+0x79f36858],si");
L0042a098:
    *edi = eax - *(ecx + -148393277);
    edi = edi + 1;
    return;
    asm("sbb esi,+0x79");
    *edi = eax;
    edi = edi + 4;
    (restore)edx;
    asm("out 0x9e,eax");
    asm("aam 0xde");
    edi = *(ecx + -576027290);
    asm("loop 0x42a12f");
    ebx = ebx - 1;
    asm("xchg eax,esi");
    esp = esp + 1;
    asm("fisubr dword [ebp-0x54]");
    (fsave) *(ebp + -1956935394);
    asm("aas");
    asm("scasb");
    al = al & 152;
    (restore)esi;
    asm("out dx,eax");
    asm("%c inc ecx");
    asm("aaa");
    asm("into");
    asm("Unknown opcode 0x8f");
    asm("xchg eax,ebp");
    asm("xchg eax,edi");
    asm("outsb");
    esi = esi - 1;
    al & 143;
    eax = eax ^ 1701274983;
L0042a0d6:
    esi = esi - 1;
    asm("xchg eax,ebp");
    asm("adc edi,[edi+edx*2]");
    goto L526f2796;
    *(edi + esi * 4) = *(edi + esi * 4) | 68;
    if(esi = bp + di) {
        goto L0042a098;
    }
    ecx = edi + -527064939;
    asm("popf");
    asm("lahf");
    asm("out dx,al");
    asm("cdq");
    asm("in eax,0xa3");
    (save)-1511364842;
    if(!(ah = ah & *esi)) {
L0042a0ff:
        al = al ^ 8;
```

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
        asm("repne mov eax,0xe3984ee");
        L0000aa3f();
        esp = esp | *(esi - 44);
        (restore)esp;
        *L766B894B = al;
        al = *esi;
        esi = esi + 1;
        asm("fidivr word [ebx]");
        *(ebx + 99) = ebp;
        goto L0042a0d6;
        edx = edx + 1 + 1;
        asm("outsb");
        ah = ah - *(esi + 1384995662);
        (restore)ecx;
        asm("jpe 0x42a0f8");
        goto L00009d62;
        edx = edx - 1;
    }
    (save)ds;
    ch = ch & *(ebx - 90);
    asm("cmpsb");
    asm("adc al,0xbc");
    (save)ds;
    esi = -1109602338;
    if(ah = ah & *edx) {
        goto L0042a0ff;
    }
    dl = dl | *LBDDCCFD0;
    edx = 1593342494;
    (save)ebp;
    asm("int1");
    al & 32;
    (restore)ds;
    gs = *(ecx + -663855310);
    asm("insd");
    asm("cmpsb");
    (restore)esi;
    eax = L00006e72();
    asm("sti");
    asm("Unknown opcode 0xd9");
    asm("out dx,eax");
    asm("pusha");
    (restore)esi;
    edx = -1984274760;
    asm("xchg eax,esp");
    asm("xchg eax,edi");
    al = al ^ ch;
    asm("adc dh,[%ds:ebx+0x5e]");

/* address  size  */
/* 0x0042a000       0 */ /* unknown */ void     __entry_point__;
/* 0x0042a1a8       0 */ /* unknown */ void     __imp__LoadLibraryA;
/* 0x0042a1ac       0 */ /* unknown */ void     __imp__GetProcAddress;
```

SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

B

```
Disassembly of File: a:\explorer32.exe

T.DateStamp = 00000000: Thu Jan 01 -8:00:00 1970
Code Offset = 00000000, Code Size = 00007C00
Data Offset = 00000000, Data Size = 00000000

Number of Objects = 0003 (dec), Imagebase = 00400000h

   Object01:      t   RVA: 00001000 Offset: 00000000 Size: 00000000
Flags: C00000E0
   Object02:          RVA: 00025000 Offset: 00000400 Size: 00004848
Flags: C00000E0
   Object03:      a   RVA: 0002A000 Offset: 00000200 Size: 00000200
Flags: C00000E0

++++++++++++++++++ RESOURCE INFORMATION ++++++++++++++++++

        There are no Resources in This Application.

++++++++++++++++++ IMPORTED FUNCTIONS ++++++++++++++++++

Number of Imported Modules =    1 (decimal)

   Import Module 001: KERNEL32.dll

++++++++++++++++++ IMPORT MODULE DETAILS +++++++++++++++++
   Import Module 001: KERNEL32.dll

Addr:0002A1C0 hint(0000) Name: LoadLibraryA
Addr:0002A1CE hint(0000) Name: GetProcAddress
++++++++++++++++++ EXPORTED FUNCTIONS ++++++++++++++++++
Number of Exported Functions =    0 (decimal)
++++++++++++++++++ Possible Strings Inside Code Block
++++++++++++++++++
++++++++++++++++++ ASSEMBLY CODE LISTING ++++++++++++++++++
//********************* Start of Code in Object CODE **************
Program Entry Point = 0042A000 (a:\explorer32.exe File Offset:00000000)


. . . .

//******************** Program Entry Point ********
=========
:0042A000 E802000000              call 0042A007
                    (StringData)"^"
:0042A005 FE7F5E
a little disassemble error near :0042A005

:0042A008 EB
:0042A009 012F                    add dword[edi], ebp
:0042A00B 684BEB1CE0              push E01CEB4B
:0042A010 E802000000              call 0042A017
:0042A015
a little disassemble error near :0042A015
```

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
:0042A015 D6
:0042A016 EB5F                    jmp 0042A077
:0042A018 5A                      pop edx
:0042A019 EB02                    jmp 0042A01D
:0042A01B CD20                    int 20
:0042A01D BB35B52520              mov ebx, 2025B535
:0042A022 E802000000              call 0042A029
:0042A027 0301                    add eax, dword[ecx]
:0042A029 5F                      pop edi
:0042A02A 03DA                    add ebx, edx
:0042A02C 13FE                    adc edi, esi
:0042A02E 8D05F4000000            lea eax, dword[000000F4]
:0042A034 23FB                    and edi, ebx
:0042A036 E801000000              call 0042A03C
:0042A03B 385E8A                  cmp byte[esi-76], bl
:0042A03E 0B83C76381CF            or eax, dword[ebx+CF8163C7]
:0042A044 F8                      clc
:0042A045 0000                    add byte[eax], al
:0042A047 0081F1FE1FE2            add byte[ecx+E21FFEF1], al
:0042A04D 47                      inc edi
:0042A04E EB01                    jmp 0042A051
:0042A050 FF83E97FEB01            inc dword[ebx+01EB7FE9]
:0042A056 872A                    xchg dword[edx], ebp
:0042A058 C8EB02CD                enter 02EB, -33
:0042A05C 2081F1E24732            and byte[ecx+3247E2F1], al
:0042A062 3CC1                    cmp al, -3F
:0042A064 F60332                  test byte[ebx], 32
:0042A067 C80BF988                enter F90B, -78
:0042A06B 0B4643                  or eax, dword[esi+43]
:0042A06E 2BFA                    sub edi, edx
:0042A070 48                      dec eax
:0042A071 EB02                    jmp 0042A075
:0042A073 C3                      ret
:0042A074 DC83E80075BC            fadd 64real[ebx+BC7500E8]
:0042A07A C1CE0E                  ror esi, 0E
:0042A07D EB01                    jmp 0042A080
:0042A07F B7DE                    mov bh, -22
:0042A081 CD7C                    int 7C
:0042A083 3D7FDE8775              cmp eax, 7587DE7F
:0042A088 47                      inc edi
:0042A089 8DE1                    lea esp, ecx
:0042A08B 8D3F                    lea edi, dword[edi]
:0042A08D 53                      push ebx
:0042A08E 97                      xchg eax, edi
:0042A08F 46                      inc esi
:0042A090 AF                      scasd
:0042A091 93                      xchg eax, ebx
:0042A092 9F                      lahf
:0042A093 9D                      popfd
:0042A094 9F                      lahf
:0042A095 63B55868F379            arpl word[ebp+79F36858], si
:0042A09B 2B81C3B227F7            sub eax, dword[ecx+F727B2C3]
:0042A0A1 AA                      stosb
:0042A0A2 C2192E                  ret 2E19
:0042A0A5 83DE79                  sbb esi, 00000079
:0042A0A8 AB                      stosd
```

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
:0042A0A9 5A                        pop edx
:0042A0AA E79E                      out port[-62], eax
:0042A0AC D4DE                      aam
:0042A0AE 8BB96685AADD              mov edi, dword[ecx+DDAA8566]
:0042A0B4 E279                      loop 0042A12F
:0042A0B6 4B                        dec ebx
:0042A0B7 96                        xchg eax, esi
:0042A0B8 44                        inc esp
:0042A0B9 DA6DAC                    fisubr dword[ebp-54]
:0042A0BC DF851E895B8B              fild 16int[ebp+8B5B891E]
:0042A0C2 3F                        aas
:0042A0C3 AE                        scasb
:0042A0C4 2498                      and al, -68
:0042A0C6 5E                        pop esi
:0042A0C7 EF                        out port[dx], eax
:0042A0C8 2E41                      inc ecx
:0042A0CA 37                        aaa
:0042A0CB CE                        into
:0042A0CC 8F95976E4EA8
a little disassemble error near :0042A0CC
:0042A0D2 8F
:0042A0D3 3567656765                xor eax, 65676567
:0042A0D8 4E                        dec esi
:0042A0D9 95                        xchg eax, ebp
:0042A0DA 133C57                    adc edi, dword[edi+2*edx]
:0042A0DD E9B4862C52                jmp 526F2796
:0042A0E2 830CB744                  or dword[edi+4*esi], 00000044
:0042A0E6 678D33                    lea esi, word[bp+di]
:0042A0E9 75AD                      jne 0042A098
:0042A0EB 8D8F95A095E0              lea ecx, dword[edi+E095A095]
:0042A0F1 9D                        popfd
:0042A0F2 9F                        lahf
:0042A0F3 EE                        out port[dx], al
:0042A0F4 99                        cdq
:0042A0F5 E5A3                      in eax, port[-5D]
:0042A0F7 681667EAA5                push A5EA6716
:0042A0FC 2226                      and ah, byte[esi]
:0042A0FE 7534                      jne 0042A134
:0042A100 3408                      xor al, 08
:0042A102 F2
a little disassemble error near :0042A102
:0042A103 B8
:0042A104 EE                        out port[dx], al
:0042A105 8439                      test byte[ecx], bh
:0042A107 0E                        push cs
:0042A108 9A729F9EEB3FAA            call far 9F72:AA3FEB9E
:0042A10F 0B66D4                    or esp, dword[esi-2C]
:0042A112 5C                        pop esp
:0042A113 A24B896B76                mov byte[766B894B], al
:0042A118 AC                        lodsb
:0042A119 DE3B                      fidivr 16int[ebx]
:0042A11B 896B63                    mov dword[ebx+63], ebp
:0042A11E EBB6                      jmp 0042A0D6
:0042A120 42                        inc edx
:0042A121 42                        inc edx
:0042A122 6E                        outs port[dx], byte
```

© SANS Institute 2000 - 2002 Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

```
:0042A123 2AA64E5B8D52          sub ah, byte[esi+528D5B4E]
:0042A129 59                    pop ecx
```

References

Lessig, Lawrence. Peer-to-Peer Harnessing the Power of Disruptive Technologies. Copyright 2001 O'Reilly& Associates, Inc. First Edition March 2001. Pages 448. Edited by Andy Oram

Sans Institute. Incident Handling Step-by-Step and Computer Crime Investigation. Sans Orlando 2002. Sans 2001

Sans Institute. Computer and Network Hacker Exploits Parts 1,2, and 3. Sans Orlando 2002. Sans 2001

Northcutt, Stephen; Zeltser, Lenny; Winters, Scott; Frederick, Karen Kent; Ritchey, Ronald W. Inside Network Perimeter Security. June 2002.

Reynolds, J./Postel, J. http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1700.html. October 1994.

Northcutt, Stephen/Kessler, Gary/Pomeranz, Hal. TCP/IP for Firewalls and Intrusion Detection. Sans 2001 Baltimore, Maryland. May 2001.

Good, Nathaniel S. and Krekelberg, Aaron. Usability and privacy: a study of Kazaa P2P file-sharing. Not dated.

Staniford, Stuart., Paxon, Vern., and Weaver, Nicholas. How to Own the Internet in Your Spare Time. URL: http://www.usenix.org/events/sec02/staniford.html.

McWilliams, Brian. Download sites Hacked, Source Code Backdoored. Security Focus. http://lwn.net/Articles/1486/. Jun 3 2002 4:37PM.

McKean, Chris. Peer-to-Peer Security and Intel's Peer-to-Peer Trusted Library. August 20, 2001. URL: http://rr.sans.org/threats/peer.php. (Requires Username and Password)

Truelove, Kelly and Chasin, Andrew. O'Reilly Network: Morpheus Out of the Underworld. July 2, 2001. URL: http://www.openp2p.com/pub/a/p2p/2001/07/02/morpheus.html

Port Numbers. URL: http://www.iana.org/assignments/port-numbers. Last updated August 15, 2002.

Minar, Nelson. Distributed Systems Topologies: Part 1 and 2

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

URL: http://www.openp2p.com/pub/a/p2p/2001/12/14/topologies_one.html. Dated December 14, 2001
URL: http://www.openp2p.com/pub/a/p2p/2002/01/08/p2p_topologies_pt2.html. Dated January 8, 2002

No author mentioned. Our Technology. http://www.grokster.com/technology.html. Not dated.

Weager, Nicholas. Reflections on Brilliant Digital: Single Points of Internet Ownership. URL: http://www.cs.berkeley.edu/~nweaver/0wn2.html. Not dated.

Heiser, Jay. "Combating Nonviral Malware, Tojans, sniffers and spyware, oh my!" Information Security Magazine, May 2002 edition.

Liu, Yana. W32.Kwbot.Worm. URL: http://www.sarc.com/avcenter/venc/data/pf/w32.kwbot.worm.html. Last updated on June 19, 2002.

Dragu, Bogdan. Backdoor.K0wbot.1.3.B. URL: http://www.bitdefender.com/virusi/virusi_descrieri.php?virus_id=87. Not dated.

Sophos Anti-Virus. W32/KWBopt-A. URL: http://www.sophos.com/virusinfo/analyses/w32kwbota.html. Not dated.

McAfee Virus Profile. W32/Kwbot.worm. URL: http://vil.mcafee.com/dispVirus.asp?virus_k=99555. July 8, 2002

Worley, Becky. KaZaA Worm Targets Users. URL: http://www.techtv.com/news/print/0,23102,3385557,00.html. Posted June 6, 2002.

Middleton, James. Kowbot worm targets Kazaa Network. URL: http://www.vnunet.com/News/1133129. January 7, 2002

No Author. Second Virus Targets KaZaZ Users. URL: http://resnet.albany.edu/news. Not dated.

Leyden, John. GamesSpy and KaZaA infected by viruses. URL: http://www.theregister.co.uk/content/56/25945.html. Posted June 26, 2002.

Landesman, Mary. KaZaA Worm, Benjamin a ploy for profit. http://anitvirus.about.com/library/weekly/aa052002a.htm

Singer, Michael. 'Benjamin' Worm plagues KaAaA. http://siliconvalley.internet.com/news/article.php/3531_1141841. May 20, 2002.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Lemos, Robert. KaZaA worm adds sour note to file swaps. http://news.com.com/2102-1001-918132.html. May 20, 2002.

© SANS Institute 2000 - 2002, Author retains full rights.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.