

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

May 1, 2000
Copyright (C) 2000. All rights reserved.

David Dittrich
University of Washington
<dittrich@cac.washington.edu>

George Weaver
Pennsylvania State University
<weaver@gabriel.nso.psu.edu>

Sven Dietrich
NASA Goddard Space Flight Center
<spock@netsec.gsfc.nasa.gov>

Neil Long
Oxford University
<neil.long@computing-services.oxford.ac.uk>

Introduction

The following is an analysis of "mstream", a distributed denial of service (DDoS) attack tool, based on the source code of "stream2.c", a classic point-to-point DoS attack tool [12].

There are currently seven major code bases of recognized DDoS tools witnessed in use "in the wild" for executing DDoS attacks:

- trino0 [03]
- Tribe Flood Network (TFN) [04]
- Tribe Flood Network 2000 (tfn2k) [06]
- stacheldraht/stacheldrahtV4 [05]
- stacheldraht v2.666
- shaft [07]
- mstream

mstream is more primitive than any of the other DDoS tools. Examination of reverse engineered and recovered C source code reveals the program to be in early development stages, with numerous bugs and an incomplete feature set compared with any of the other listed tools. The effectiveness of the stream/stream2 attack itself, however, means that it will still be disruptive to the victim (and agent) networks even with an attack network consisting of only a hand full of agents.

(The source code for mstream was anonymously posted to the vuln-dev@securityfocus.com and BUGTRAQ email lists on April 29, 2000. [20] As a result, this analysis is being released in a bit less polished and complete form that originally intended, but will hopefully allow incident response teams and vendors to save some time in developing

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

their responses. Any errors we didn't catch are likely a result of this rush.)

The reader is advised that modification of the source code can and would change any of the details of this analysis, such as prompts, passwords, commands, TCP/UDP port numbers, or supported attack methods, signatures, and features. In fact, the values of communication ports have already been seen to differ between published and "in the wild" code.

[Note also that throughout this analysis, actual nicks, site names, and IP addresses have been sanitized.]

The handler/agent terminology used in this analysis was developed at the CERT Distributed System Intruder Tools workshop held in November 1999, and will be used in this analysis. It is highly recommended that the CERT workshop report [01] be read first, as well as background and supporting information on other DDoS tools [08]:

http://www.cert.org/reports/dsit_workshop.pdf
<http://staff.washington.edu/dittrich/misc/ddos/>

An mstream agent was discovered in late April 2000 on a compromised Linux system at a major university. This system was identified to be flooding packets using forged source addresses, targeted at over a dozen IP addresses.

Since RFC 2267 style egress filtering [13] was employed on this network, and all 32 bits of the source addresses were forged, only an extremely small number of attack packets (matching the internal network blocks) could have managed to leave the agent's network. The traffic did, however, cause the router (which served 18 subnets) to become non-responsive. This means that sites that do egress filtering may still suffer from these attacks themselves, even if the intended "victim" receives fewer packets than the attacker(s) intended. The lesson here is that there is no "quick fix" to DDoS in the form of simple technical filtering solutions. [The router manufacturer was briefed and is currently working on identifying the cause and fixes.]

Another side-effect of taking egress filtering down one level to that of internal subnets is that rejected packets will not make it past the router doing the filtering, so the effects of bandwidth consumption or router disruption will not be felt above the level of the router doing the filtering (or being saturated). This means a border router based IDS (e.g., net flows), or one outside your borders on a DMZ or upstream ISP's network, will not identify the attempted attacks. Unless you are monitoring the routers themselves, only user complaints would tip you off to an attack originating from your network. It also means that packet level analysis is made more difficult, as it must be done *in front* of the router doing the filtering in order to capture all packets. This puts an added burden on network engineers or incident responders to do packet level dumping, on site in a router closet, in order to know with a high degree of reliability what is going on.

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

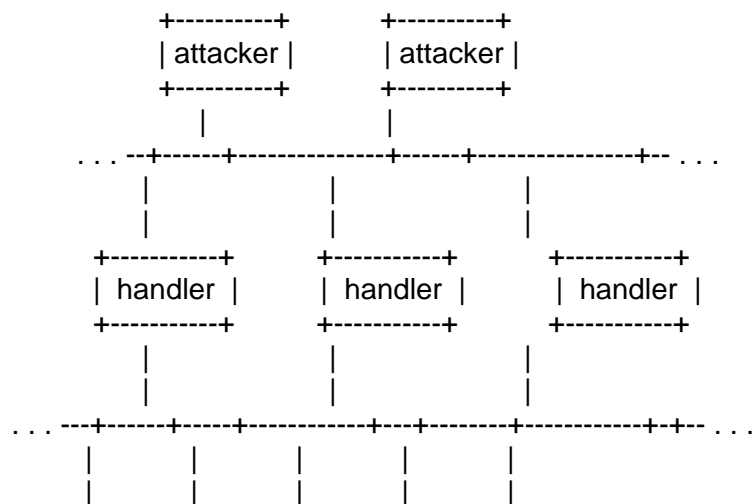
A major problem in forensic analysis of these attacks -- whether successful or blocked by egress filters -- is having policies and procedures in place to facilitate packet level logging. At least one site that was a victim of early February 2000 DoS attacks on eCommerce sites was not prepared for this kind of analysis, and to this day is not aware of what attack tool was used against them. They only knew their routers were crashing and relied on their upstream provider to determine how to block packets to restore throughput. This results in a major roadblock to investigation by law enforcement, who have been criticized heavily by some for their slow response (it is a non-trivial problem to perform an investigation of a DoS or DDoS attack with nothing but router or browser response behavioral observations to go on.) A description of the details of the attack on Yahoo! was published on Packet Storm Security's web site [11], but not many other details were available. (This was also pointed out by Anonymous [20] in his/her post.)

Unlike the high volume mass intrusions witnessed with trinoo, TFN, and stacheldraht in 1999, this particular mstream network appears to be in the very early stages of code development and to have been set up by hand (on both handler and agent systems) with a slightly modified Linux rootkit version 4 [10] to conceal the presence of the intruder's activity. For a description of intrusion and concealment on an agent system, see Appendix D. For a description of intrusion and concealment on a handler system, see Appendix E.

The network: client(s)-->handler(s)-->agent(s)-->victim(s)

The mstream network, like trinoo and shaft, is made up of one or more handlers ("master.c") and a large set of agents ("server.c"). Attacker to handler communication is at present unencrypted over TCP, with handler <-> agent communication unencrypted over UDP.

An mstream network would look like this:



THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

```
+-----+ +-----+ +-----+ +-----+ +-----+
| agent | | agent | | agent | | agent | | agent |
+-----+ +-----+ +-----+ +-----+ +-----+
```

Communication

Attacker to handler(s): 6723/tcp (in published source)
15104/tcp ("in the wild")
12754/tcp (in recovered source)
Agent to Handler(s): 9325/udp (in published source)
6838/udp ("in the wild")
Handler to agent(s): 7983/udp (in published source)
10498/udp ("in the wild")

Remote control of the mstream handler is accomplished via a TCP connection to port 6723/tcp (or 15104/tcp, or 12754/tcp, or...).

The handler expects commands to be contained entirely in the data payload of a single TCP packet, not broken up character by character in a stream. This means that "telnet" cannot be used to control a handler, but instead some other client program must be used to buffer the command line before sending (e.g., a special command shell or port redirector, netcat [19], etc. -- no special client was included in the source posted on Security Focus [20]).

The traffic over this connection is not encrypted (although it has been shown in stacheldraht that adding a Blowfish block cipher is not difficult.) Command lines are space separated argument lists.

Like trinoo, communication between the handler(s) and agent(s) is accomplished using UDP datagrams. Agent commands are slash ("/") separated argument lists, with some multi-item arguments being colon (":") separated lists.

[It is believed that the difference between attacker <-> handler communication port numbers between "lsof" output on a system running an active handler and that in the recovered source code is due to version differences. That is, the recovered source may have been an older version than the source used to compile the handler actually running at the time. Both ports are identified here, although they are trivial to change to something else (and were found to be different in the published source [20]).]

The examined code limits the maximum number of connected attackers to 3. This may be a protective measure, or possibly a redundant access measure in case one or more systems used as intermediaries by the attacker are discovered or otherwise taken out of service.

After connecting, the user must supply the proper password (default is "N7%diApf!" in the recovered code, and "sex" in the published code

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

[20]). If the proper password is not given, all currently connected users are notified of the attempt and the connection is dropped. If the proper password is given, all currently connected users are informed of the new session and the user is presented with a "> " prompt.

Handler Commands

The handler commands consist of up to 3 space-delimited fields.

If a connected attacker does not enter a command within 420 seconds, the connection is terminated. (It is assumed "420" was not chosen simply because 7 minutes is an ideal timeout value. ;)

The handler command set is:

help

Prints the following:

Available commands:

stream	stream attack !
servers	Prints all known servers.
ping	ping all servers.
who	tells you the ips of the people logged in
mstream	lets you stream more than one ip at a time

servers

List all currently known agents.

who

Shows the currently connected users.

ping

Identify remaining active agents. Sends the command "ping" to all known agents and reports to the connected users as each "pong" reply is received.

stream <hostname> <seconds>

Begin an attack against a single host, for the specified duration. The handler resolves the hostname to an IP address and sends the command "mstream/arg1:arg1/arg2" to all agents, where "arg1" is the resolved hosts' IP address twice with a colon between (this simplifies argument parsing in the agent) and "arg2" is the duration in seconds.

mstream <ip1:ip2:ip3:...> <seconds>

Begin an attack against multiple IP addresses, for the specified duration. The handler sends the command "mstream/arg1/arg2" to all agents, where "arg1" is the list of colon separated IP addresses, and "arg2" is the duration in seconds. Also for simplicity, in this command, there is no host name resolution (i.e., you MUST specify all targets by a properly formed colon

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

separated list of IP addresses)

quit

Terminates the attacker's connection to the handler.

Agent Commands

The handler communicates to agents using string based commands in the data portion of UDP packets. These commands are not encrypted (although this, too, can easily be changed.)

There are only three agent commands currently. Commands are either a simple string, or a slash ("/") separated command and argument list.

ping

Replies to IP address that sent this packet with "pong".

stream/IP/seconds

Starts streaming at IP address for specified duration in seconds.

mstream/IP1[:IP2[:IPN]]/seconds

Starts streaming at all of the colon separated list of IP addresses for specified duration in seconds.

Even though the agent "in the wild" had two options for accepting DDoS commands, namely "stream" and "mstream" as in the published source, only the mstream command is used in the handler to agent protocol. A simple "stream 192.168.0.100 10" command to the handler sends the powerful command "mstream/192.168.0.100:192.168.0.100/10" to the agent, when in fact a simple "stream/192.168.0.100/10" should be generated. It is not clear why this was done, but it does look like simple algorithms are used for command parsing, so this might just indicate a "quick and dirty" development process.

Password protection

The handler is password protected, to prevent trivial takeover of the network handler. The password is not encrypted, just a string that is compared against the data payload of the initial packet as-is.

It should be explicitly noted here again that this program has a feature not found in other DDoS tools, which informs all connected users of access, successful or not, to the handler(s) by competing parties (black hat or white hat). Thus it does not matter that you can identify the password string in the binary, since you can't use it without detection (and can't simply hijack the TCP session, either, because of the command buffering described in the Communication section.)

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

There is no password protection of handler <-> agent communication, but that isn't surprising. As was seen with trinoo, a password in clear text is not much of a defense and is trivially attacked by sniffing network traffic.

Fingerprints

As mentioned above, command strings between the handler(s) and agent(s) is visible in packet flows.

Visible strings in the agent (in two truncated columns to save space) are:

```
-----  
ELF                mstream  
/lib/ld-linux.so.2 ping  
GNU                pong  
__gmon_start__    fork  
libc.so.6         init.c  
random            ...  
getpid            server.c  
perror            strchr@@GLIBC_2.0  
getuid            packet  
malloc            getpid@@GLIBC_2.0  
recvfrom          _DYNAMIC  
socket            _etext  
bind              __register_frame_info@@GLIBC_2.0  
inet_addr         recvfrom@@GLIBC_2.0  
__deregister_frame_info _fp_hw  
setsockopt        perror@@GLIBC_2.0  
rand              fork@@GLIBC_2.0  
strncmp           sock  
strncpy           cksum  
sendto            random@@GLIBC_2.0  
strtok            _init  
fork              malloc@@GLIBC_2.0  
memset            getpid@@GLIBC_2.0  
srand             sendto@@GLIBC_2.0  
getppid           __deregister_frame_info@@GLIBC_2.0  
time              setsockopt@@GLIBC_2.0  
htons             time@@GLIBC_2.0  
exit              _start  
atoi             forkbg  
_IO_stdin_used   strlen@@GLIBC_2.0  
__libc_start_main stream  
strlen            strncmp@@GLIBC_2.0  
strchr            inet_addr@@GLIBC_2.0  
__register_frame_info __bss_start  
free              main  
GLIBC_2.0         __libc_start_main@@GLIBC_2.0  
PTRh              data_start
```

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

```
QVh0          bind@@GLIBC_2.0
Ph%           getuid@@GLIBC_2.0
PhG           _fini
WVS           s_in
[^_           srand@@GLIBC_2.0
WVS           nlstr
j(j)          exit@@GLIBC_2.0
j h           atoi@@GLIBC_2.0
j(h)          _edata
j h           in_cksum
j(h)          _GLOBAL_OFFSET_TABLE_
[^_           free@@GLIBC_2.0
131.247.208.191 _end
129.79.20.202 htons@@GLIBC_2.0
socket        send2master
bind          memset@@GLIBC_2.0
setsockopt    strncpy@@GLIBC_2.0
newserver     _IO_stdin_used
stream        strtok@@GLIBC_2.0
__data_start __gmon_start__
socket@@GLIBC_2.0 rand@@GLIBC_2.0
```

Visible strings in the handler are:

```
% strings -n 3 master          Available commands:
socket                         stream
bind                           stream attack !
listen                          servers
setsockopt                       Prints all known servers.
fcntl                            ping
You're too idle !                ping all servers.
Connection from %s              who
newserver                       tells you the ips of the people log
New server on %s.               mstream
pong                             lets you stream more than one ip at
Got pong number %d from %s      who
%s has disconnected (not auth'd): % Currently Online:
Invalid password from %s.       Socket number %d
Password accepted for connection fr [%s]
Lost connection to %s: %s      ping
stream                          Pinging all servers.
Usage: stream <hostname> <seconds> mstream
Unable to resolve %s.           Usage: mstream <ip1:ip2:ip3:...> <s
stream/%s/%s                     MStreaming %s for %s seconds.
Streaming %s for %s seconds.     mstream/%s/%s
quit                              fork
%s has disconnected.              Forked into background, pid %d
servers                          Caught SIGHUP, ignoring.
Server file doesn't exist, creating Caught SIGINT, ignoring.
The following ips are known servers Segmentation Violation, Exiting cle
help                              Caught unknown signal, This should
```

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

commands

The agent (named "rpc.wall" on this system -- this same name was used for the handler as well) is seen with "lsdf" as follows:

COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME
rpc.wall 588 root cwd DIR 3,2 1024 2 /
rpc.wall 588 root rtd DIR 3,2 1024 2 /
rpc.wall 588 root txt REG 3,3 17016 15765 /usr/bin/rpc.wall
rpc.wall 588 root mem REG 3,2 342206 30771 /lib/ld-2.1.1.so
rpc.wall 588 root mem REG 3,2 4016683 30789 /lib/libc-2.1.1.so
rpc.wall 588 root 0u CHR 5,1 4952 /dev/console
rpc.wall 588 root 1w FIFO 0,0 646 pipe
rpc.wall 588 root 2w FIFO 0,0 647 pipe
rpc.wall 588 root 3u IPv4 656 UDP *:10498
rpc.wall 588 root 4u IPv4 657 UDP *:1044
rpc.wall 588 root 5u IPv4 658 UDP *:1045
rpc.wall 588 root 6u raw 30219 00000000:00FF->00000000:0000 st=07
rpc.wall 588 root 7r FIFO 0,0 648 pipe
rpc.wall 588 root 8u raw 30241 00000000:00FF->00000000:0000 st=07
rpc.wall 588 root 9u CHR 5,1 4952 /dev/console
rpc.wall 588 root 10u IPv4 30244 UDP *:1051
rpc.wall 588 root 11u raw 30245 00000000:00FF->00000000:0000 st=07
rpc.wall 588 root 21w FIFO 0,0 648 pipe

Bugs in the source code for both handler and agent result in an increasing number of raw sockets and UDP sockets in the agent (three each are were witnessed on this agent), and an increasing number of open file handles and UDP sockets in the handler (hundreds were shown by Andrew Korty). [This is no doubt an indication that mstream is in the early development stages, so this signature should not be counted on to identify a handler or agent running on a system.]

When an agent first starts up, it sends a "newserver" command to the list of default handlers compiled into it, as seen here with tcpdump:

00:04:38.530000 192.168.0.20.1081 > 192.168.0.100.6838: udp 9
0x0000 4500 0025 ef75 0000 4011 098a c0a8 0014 E..%.u..@.....
0x0010 c0a8 0064 0439 1ab6 0011 2b63 6e65 7773 ...d.9....+cnews
0x0020 6572 7665 7200 0000 0000 0000 0000 erver.....

If a rootkit is in place (as it was on both handler and agent systems), you cannot trust the standard operating system commands to show you the running handler or agent, or their network connections. The main lesson to be learned from rootkits is that a large percentage of Unix system administrators will NOT be skilled enough to get around rootkits. This means a couple things.

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

First, and fundamentally, intruders will tend to have an even greater advantage over unskilled system administrators. It is becoming ever more important that systems administrators -- Unix, NT, whatever -- have training as a primary task, not a luxury or burden to be avoided. The moment of a security incident is NOT the proper time to catch up on months of missed learning, and causes undue pressure to take shortcuts to get the system back up quickly (usually making things MUCH worse).

Second, incident response and forensic investigation may be made more difficult, if not impossible, as the simple "solution" that the unskilled Unix administrator will take is to give up and just re-install the operating system. This ill-advised choice of action destroys any evidence that may exist on the system and sets the system up for a subsequent intrusion because the same security precautions they did not take before the incident will usually not be taken this time either. All too often, this action is taken without first seeking advice and before incident response teams are able to notify the administrator and assist them in taking the correct steps. All system administrators are urged to take the time now to prepare to deal with rootkits [10].

As mentioned above, if an agent receives a UDP packet on port 10498/udp, containing the string "ping" as its data payload (and if it is not actively streaming at the time), it will reply to the sending system with a UDP packet on port 6838/udp with the string "pong" as its data payload. (The trailing zeroes are just a tcpdump artifact. The payload is clearly 4 bytes.)

```
-----  
00:05:16.457239 192.168.0.100.65364 > 192.168.0.20.10498: udp 5  
0x0000 4500 0021 f412 0000 4011 04f1 c0a8 0064 E!.....@.....d  
0x0010 c0a8 0014 ff54 2902 000d 6ce3 7069 6e67 .....T)...l.ping  
0x0020 0a
```

```
00:05:16.458214 192.168.0.20.1083 > 192.168.0.100.6838: udp 4  
0x0000 4500 0020 ef8c 0000 4011 0978 c0a8 0014 E.....@..x....  
0x0010 c0a8 0064 043b 1ab6 000c 8045 706f 6e67 ...d.;.....Epong  
0x0020 0000 0000 0000 0000 0000 0000 0000 .....
```

This sequence allows signature matching with programs like "ngrep" [14], "snort" [18] (see Appendix B for snort rules), or scanning for idle agents using "rid" [15] (see Appendix C for a RID template).

The ngrep command string to detect these packets would be:

```
# ngrep "p[oi]ng" udp port 6838 or udp port 10498
```

(You will need to tailor this command, or add the other ports listed above from the published code, to decrease the chance of false negatives.)

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

Attack packets have a fixed size of 40 bytes per packet, which may be on purpose to evade large packet triggers that exist on some IDSs.

The stream2.c attack floods the victim with TCP ACK packets, using forged source addresses generated by random() (i.e., any or all of the four octets will occasionally be zero) and incrementing source port and sequence numbers, as seen in this code snippet:

```
-----  
...  
  for(i=0; ; ++i) {  
    cksum.pseudo.saddr = packet.ip.ip_src.s_addr = random();  
    ++packet.ip.ip_id;  
    ++packet.tcp.th_sport;  
    ++packet.tcp.th_seq;  
  
    if (!dstport)  
      s_in.sin_port = packet.tcp.th_dport = rand();  
    ...  
  }  
-----
```

During an attack, the following packet signature would be observed (as seen by tcpdump using a recovered agent binary):

```
-----  
01:39:24.701083 192.168.0.2.65527 > 192.168.0.20.10498: [bad udp cksum 3100!]  
udp 24 (ttl 64, id 886)  
0x0000 4500 0034 0376 0000 4011 f5dc c0a8 0002 E..4.v..@.....  
0x0010 c0a8 0014 fff7 2902 0020 556c 7374 7265 .....)...Ulstre  
0x0020 616d 2f31 3932 2e31 3638 2e30 2e31 3030 am/192.168.0.100  
0x0030 2f31 300a /10.  
  
01:40:10.132724 192.168.0.2.65526 > 192.168.0.20.10498: [bad udp cksum 3100!]  
udp 24 (ttl 64, id 930)  
0x0000 4500 0034 03a2 0000 4011 f5b0 c0a8 0002 E..4....@.....  
0x0010 c0a8 0014 fff6 2902 0020 556d 7374 7265 .....)...Umstre  
0x0020 616d 2f31 3932 2e31 3638 2e30 2e31 3030 am/192.168.0.100  
0x0030 2f31 300a /10.  
  
01:41:23.674796 192.168.0.2.65525 > 192.168.0.20.10498: [bad udp cksum 4a00!]  
udp 49 (ttl 64, id 1031)  
0x0000 4500 004d 0407 0000 4011 f532 c0a8 0002 E..M....@..2....  
0x0010 c0a8 0014 fff5 2902 0039 a9b4 6d73 7472 .....)..9..mstr  
0x0020 6561 6d2f 3139 322e 3136 382e 302e 313a eam/192.168.0.1:  
0x0030 3139 322e 3136 382e 302e 3130 303a 3139 192.168.0.100:19  
0x0040 322e 3136 382e 302e 322f 3130 0a 2.168.0.2/10.  
  
01:41:23.675771 arp who-has 192.168.0.1 tell 192.168.0.20  
0x0000 0001 0800 0604 0001 0010 5a99 6544 c0a8 .....Z.eD..  
0x0010 0014 0000 0000 0000 c0a8 0001 0000 0000 .....  
0x0020 0000 0000 0000 0000 0000 0000 0000 .....  
  
01:41:23.675772 arp who-has 192.168.0.100 tell 192.168.0.20
```

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

0x0000 0001 0800 0604 0001 0010 5a99 6544 c0a8Z.eD..
0x0010 0014 0000 0000 0000 c0a8 0064 0000 0000d....
0x0020 0000 0000 0000 0000 0000 0000 0000

01:41:23.675773 77.172.43.85.38444 > 192.168.0.2.26296: . [tcp sum ok]
ack 0 win 16384 [tos 0x8] (ttl 255, id 50237)
0x0000 4508 0028 c43d 0000 ff06 bdde 4dac 2b55 E..(=.....M.+U
0x0010 c0a8 0002 962c 66b8 ea97 d237 0000 0000,f....7....
0x0020 5010 4000 7c74 0000 0000 0000 0000 P.@.|t.....

01:41:23.675774 88.148.222.45.39212 > 192.168.0.2.10342: . [tcp sum ok]
ack 0 win 16384 [tos 0x8] (ttl 255, id 51005)
0x0000 4508 0028 c73d 0000 ff06 fd1d 5894 de2d E..(=.....X.-
0x0010 c0a8 0002 992c 2866 ed97 d237 0000 0000,f...7....
0x0020 5010 4000 f705 0000 0000 0000 0000 P.@.....

01:41:23.675775 0.18.219.113.39980 > 192.168.0.2.41622: . [tcp sum ok]
ack 0 win 16384 [tos 0x8] (ttl 255, id 51773)
0x0000 4508 0028 ca3d 0000 ff06 555c 0012 db71 E..(=....U\...q
0x0010 c0a8 0002 9c2c a296 f097 d237 0000 0000,.....7....
0x0020 5010 4000 d213 0000 0000 0000 0000 P.@.....

01:41:23.675776 121.161.140.109.40748 > 192.168.0.2.16749: . [tcp sum ok]
ack 0 win 16384 [tos 0x8] (ttl 255, id 52541)
0x0000 4508 0028 cd3d 0000 ff06 27d1 79a1 8c6d E..(=.....'y..m
0x0010 c0a8 0002 9f2c 416d f397 d237 0000 0000,Am...7....
0x0020 5010 4000 02b2 0000 0000 0000 0000 P.@.....

01:41:23.675777 79.238.213.72.41516 > 192.168.0.2.46276: . [tcp sum ok]
ack 0 win 16384 [tos 0x8] (ttl 255, id 53309)
0x0000 4508 0028 d03d 0000 ff06 05a9 4fee d548 E..(=.....O..H
0x0010 c0a8 0002 a22c b4c4 f697 d237 0000 0000,.....7....
0x0020 5010 4000 6a32 0000 0000 0000 0000 P.@.j2.....

01:41:23.675778 104.24.203.64.42284 > 192.168.0.2.61623: . [tcp sum ok]
ack 0 win 16384 [tos 0x8] (ttl 255, id 54077)
0x0000 4508 0028 d33d 0000 ff06 f486 6818 cb40 E..(=.....h..@
0x0010 c0a8 0002 a52c f0b7 f997 d237 0000 0000,.....7....
0x0020 5010 4000 1a1d 0000 0000 0000 0000 P.@.....

01:41:23.675779 37.60.73.50.43052 > 192.168.0.2.51311: . [tcp sum ok]
ack 0 win 16384 [tos 0x8] (ttl 255, id 54845)
0x0000 4508 0028 d63d 0000 ff06 b671 253c 4932 E..(=.....q%<I2
0x0010 c0a8 0002 a82c c86f fc97 d237 0000 0000,o...7....
0x0020 5010 4000 0150 0000 0000 0000 0000 P.@..P.....

01:41:23.675780 142.14.73.40.43820 > 192.168.0.2.8979: . [tcp sum ok]
ack 0 win 16384 [tos 0x8] (ttl 255, id 55613)
0x0000 4508 0028 d93d 0000 ff06 4aa9 8e0e 4928 E..(=.....J...I(
0x0010 c0a8 0002 ab2c 2313 ff97 d237 0000 0000,#...7....
0x0020 5010 4000 37e4 0000 0000 0000 0000 P.@.7.....

01:41:23.676748 144.19.212.69.44588 > 192.168.0.2.51668: . [tcp sum ok]

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

```
ack 0 win 16384 [tos 0x8] (ttl 255, id 56381)
0x0000 4508 0028 dc3d 0000 ff06 ba86 9013 d445 E..(=.....E
0x0010 c0a8 0002 ae2c c9d4 0298 d237 0000 0000 .....,.....7....
0x0020 5010 4000 fdff 0000 0000 0000 0000 P.@.....
```

```
01:41:23.676749 155.176.45.2.45356 > 192.168.0.2.32793: . [tcp sum ok]
ack 0 win 16384 [tos 0x8] (ttl 255, id 57149)
0x0000 4508 0028 df3d 0000 ff06 532d 9bb0 2d02 E..(=....S-...-
0x0010 c0a8 0002 b12c 8019 0598 d237 0000 0000 .....,.....7....
0x0020 5010 4000 dd61 0000 0000 0000 0000 P.@..a.....
```

```
01:41:23.676750 10.98.211.13.46124 > 192.168.0.2.1995: . [tcp sum ok]
ack 0 win 16384 [tos 0x8] (ttl 255, id 57917)
0x0000 4508 0028 e23d 0000 ff06 3b70 0a62 d30d E..(=.....;p.b..
0x0010 c0a8 0002 b42c 07cb 0898 d237 0000 0000 .....,.....7....
0x0020 5010 4000 3af3 0000 0000 0000 0000 P.@:.....
```

```
01:41:23.676751 214.235.187.89.46892 > 192.168.0.2.14172: . [tcp sum ok]
ack 0 win 16384 [tos 0x8] (ttl 255, id 58685)
0x0000 4508 0028 e53d 0000 ff06 839a d6eb bb59 E..(=.....Y
0x0010 c0a8 0002 b72c 375c 0b98 d237 0000 0000 .....,7\...7....
0x0020 5010 4000 508c 0000 0000 0000 0000 P.@.P.....
```

```
01:41:23.676752 90.193.127.8.47660 > 192.168.0.2.64812: . [tcp sum ok]
ack 0 win 16384 [tos 0x8] (ttl 255, id 59453)
0x0000 4508 0028 e83d 0000 ff06 3916 5ac1 7f08 E..(=....9.Z...
0x0010 c0a8 0002 ba2c fd2c 0e98 d237 0000 0000 .....,.....7....
0x0020 5010 4000 3d37 0000 0000 0000 0000 P.@.=7.....
```

```
01:41:23.676753 160.176.42.60.48428 > 192.168.0.2.17432: . [tcp sum ok]
ack 0 win 16384 [tos 0x8] (ttl 255, id 60221)
0x0000 4508 0028 eb3d 0000 ff06 44f3 a0b0 2a3c E..(=....D... * <
0x0010 c0a8 0002 bd2c 4418 1198 d237 0000 0000 .....,D....7....
0x0020 5010 4000 ff28 0000 0000 0000 0000 P.@..(.....
```

An attack (only packets including "0" octets are shown) would similarly be seen with Cisco Net Flows like this:

```
-----
% grep "[\.]0[\.]" ddos-000415
Apr 15 04:12:08 tcp 82.0.151.5(29497) -> 192.168.10.5(27072), 1 packet
Apr 15 04:12:18 tcp 207.0.149.32(21893) -> 192.168.10.5(3913), 1 packet
Apr 15 04:12:33 tcp 0.147.151.82(10473) -> 10.4.152.237(2810), 1 packet
Apr 15 04:13:39 tcp 60.0.33.36(41079) -> 10.4.152.237(31754), 1 packet
Apr 15 04:14:03 tcp 103.140.148.0(4247) -> 10.4.152.237(29689), 1 packet
Apr 15 04:14:15 tcp 214.1.99.0(46714) -> 10.4.152.237(22524), 1 packet
Apr 15 04:15:11 tcp 10.148.60.0(12276) -> 192.168.10.5(31122), 1 packet
Apr 15 04:15:20 tcp 0.112.67.108(4550) -> 192.168.10.5(63787), 1 packet
Apr 15 04:15:33 tcp 13.0.16.2(39092) -> 10.4.152.237(57998), 1 packet
...
Apr 15 06:45:24 tcp 18.167.171.0(54104) -> 10.200.5.8(32779), 1 packet
Apr 15 06:45:52 tcp 0.23.15.38(45621) -> 10.200.5.8(20780), 1 packet
```

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

```
Apr 15 06:46:14 tcp 0.12.109.77(38670) -> 10.200.5.8(47776), 1 packet
Apr 15 07:19:12 tcp 199.120.0.72(64912) -> 10.4.152.237(45151), 1 packet
Apr 15 07:27:37 tcp 0.28.232.21(52533) -> 10.4.152.237(338), 1 packet
Apr 15 07:28:13 tcp 99.61.233.0(20951) -> 10.4.152.237(58427), 1 packet
Apr 15 07:31:23 tcp 195.0.3.111(17193) -> 10.4.152.237(14601), 1 packet
Apr 15 07:32:19 tcp 61.108.245.0(24309) -> 10.4.152.237(32809), 1 packet
```

It should also be noted that some of the forged source addresses are broadcast addresses, multicast addresses, or network addresses, which can have ramifications if packets are directed back to the flooding systems (see [12]).

Analysis of the de-compiled agent source code shows the stream2.c attack is altered slightly to randomize more header fields, with some static values that can be noted when analyzing network traffic:

```
packet.ip.ip_id = rand();
...
packet.tcp.th_win = htons(16384);
...
packet.tcp.th_seq = random();
...
packet.tcp.th_sport = rand();
packet.tcp.th_dport = rand();
...
while (time(0) <= endtime) {
  if (floodtype != 0) {
    i = 0;
    while (arg4[i] != NULL) { /* until list exhausted */
      if (strchr(arg4[i], '.') != NULL) { /* valid ip */
        packet.ip.ip_dst.s_addr = inet_addr(arg4[i]);
        cksum.pseudo.daddr = inet_addr(arg4[i]);
        s_sin.sin_addr.s_addr = inet_addr(arg4[i]);

        cksum.pseudo.saddr = packet.ip.ip_src.s_addr = random();
        packet.ip.ip_id++;
        packet.tcp.th_sport++;
        packet.tcp.th_seq++;

        s_in.sin_port = packet.tcp.th_dport = rand();
      }
    }
  }
}
```

Defenses

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

There are none. We are all doomed. Time to shop for property in Montana and stock up on non-perishable food items and semi-automatic weapons.

(Seriously, we didn't have time to finish this section. Any defenses already discussed for stream/stream2 [12] or other DDoS tools [08] would apply. And for God's sake, SECURE ALL THOSE SYSTEMS they are using to build these networks! ;)

Weaknesses

Control communication (attacker <-> handler and handler <-> agent) is not encrypted, and is thus subject to session hijacking and third-party control, respectively. Agents do not authenticate the source of commands, so you can easily use the "ping"/"pong" feature to detect agents that are not currently streaming.

In fact, "pong" is the only response sent back to the handler, and "newserver" is only sent to handler(s) when the program first starts. (The installation of the agent described in Appendix D causes it to be run after each boot. This would mean a set of "newserver" packets could be detected at this point, although it is not recommended that rebooting be used as the first step in verifying an agent installation. It is better to follow the more thorough forensic analysis techniques detailed in Appendices D and E to ensure as little evidence as possible is deleted or altered.)

The agent will segmentation fault upon receiving a badly formatted command, e.g. "stream foo bar". It is conceivable that such fault generation can disable agents, but they will most likely get restarted at the next reboot of the host.

Flooding an agent with requests will saturate its file handle limit and cause it to become unresponsive. Since no notification or acknowledgement is made to the handler, one can easily send >1024 commands to the agent. Most floods are short, so it may be possible to "squeeze in there".

The agent process is single-threaded, implying that it will not process incoming commands while in mid-flood. This renders any ideas about remotely crashing or halting a flooding agent unworkable.

The agent also will, in multi-flood mode, flood all hosts in the colon-delimited list equally for the same amount of time. This seems to either implicate packet loss due to congestion or require that multiple flooding machines with different flood durations were in play to explain evidence witnessed in argus flows at third party sites, showing the side-effects of multiple targets being hit at the same time with spoofed IPs (namely ICMP "Host Unreachable" packets and TCP RST packets).

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

One thing that has been witnessed is that when there are multiple targets which start at the same time there is a tailing-off with some targets being hit longer than others and the rate of spoofed packets also seems to decline with time - sort of dwindling. (We ran out of time to analyze third-party effects of flooding, but there will no doubt be followup on this topic on BUGTRAQ.)

The next logical evolutionary steps

It is not a stretch to assume that features from other published DDoS tools will make their way into tools like mstream. Briefly, some likely enhancements to this code include:

- 1) Adding source port filtering for connections to handler.
- 2) Adding authentication between handler and agent.
- 3) Packet size selection.
- 4) Flags for flood packets (ACK, RST, NUL, random, whatever).
- 5) Encryption for attacker<->handler traffic.
- 6) More obfuscation of embedded commands.

FIN

Credits

As principal author of this particular analysis, I guess I get to do the shouts, greetz, and mad props.

Since last fall, I've spent every major holiday analyzing a DDoS tool. Halloween (trinoo), Thanksgiving (TFN), Christmas (stacheldraht) and New Years (building/testing scanning tools for all three). Sven Dietrich (no, we are not related and are not the same person as some have mistaken! ;) took the lead on analyzing "shaft" [07] along with Neil Long, allowing me to actually relax a bit on President's and Valentine's days.

Then some DDoS attacks brought mstream out of the shadows, and over the Easter Holiday weekend George Weaver kicked butt hand-decompiling the agent, Sven Dietrich obtaining packet and system call traces on a test network, and Neil Long gathering third-party effects data to correlate with actual attacks, while I analyzed forensic data gathered from an agent system and tried to keep up with the rest. Its amazing what can get done when four people consume that much sweets and hard boiled eggs.

Credit also goes to Andrew Korty and investigators at Indiana University for their forensic analysis and data gathering. Site to site cooperation and coordination is key to incident response in DDoS attacks, and the more people who know about this and learn the techniques, the better off we'll all be.

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

I should also thank Anonymous, who posted the source code for mstream through Security Focus [20], acknowledging me by name for my earlier work. My only reservation is that the result of publishing the source (instead of sending it to me directly) was to alter all of our weekend plans to rush this analysis out (e.g., I was planning on doing a training hike all day Sunday.) Whoever you are, I should think you owe me at least a sixer of good microbrew for that! ;) (The upside is that this simplifies the problem of getting code into the hands of all vendors, incident response teams, and security software authors in a fair and above board way.)

--

Dave Dittrich <dittrich@cac.washington.edu>
University of Washington

Appendix A - References

- [00] TCP/IP Illustrated, Vol. I, II, and III. W. Richard Stevens and Gary R. Wright., Addison-Wesley
- [01] CERT Distributed System Intruder Tools Workshop report
http://www.cert.org/reports/dsit_workshop.pdf
- [02] CERT Advisory CA-99-17 Denial-of-Service Tools
<http://www.cert.org/advisories/CA-99-17-denial-of-service-tools.html>
- [03] The DoS Project's "trinoo" distributed denial of service attack tool, David Dittrich
<http://staff.washington.edu/dittrich/misc/trinoo.analysis>
- [04] The "Tribe Flood Network" distributed denial of service attack tool, David Dittrich
<http://staff.washington.edu/dittrich/misc/tfn.analysis>
- [05] The "stacheldraht" distributed denial of service attack tool, David Dittrich
<http://staff.washington.edu/dittrich/misc/stacheldraht.analysis>
- [06] TFN2K - An Analysis, Jason Barlow and Woody Thrower, Axent Security Team
http://packetstorm.securify.com/distributed/TFN2k_Analysis-1.3.txt
- [07] An analysis of the ``Shaft" distributed denial of service tool, Sven Dietrich, Neil Long, and David Dittrich
http://netsec.gsfc.nasa.gov/~spock/shaft_analysis.txt
- [08] Distributed Denial of Service (DDoS) Attack Tools, David Dittrich
<http://staff.washington.edu/dittrich/misc/ddos/>
- [09] Distributed denial of service attack tools at Packet Storm Security

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

<http://packetstorm.securify.com/distributed/>

[10] "Root Kits" and hiding files/directories/processes after a break-in,
David Dittrich
<http://staff.washington.edu/dittrich/misc/faqs/rootkits.faq>

[11] Technical details of the attack on Yahoo!
<http://packetstorm.securify.com/distributed/yahoo.txt>

[12] BUGTRAQ threads on the stream.c DoS attack and its fallout
<http://staff.washington.edu/dittrich/misc/ddos/stream.txt>

[13] RFC 2267 -- Network Ingress Filtering: Defeating Denial of Service
Attacks which employ IP Source Address Spoofing, Paul Fergussen
and Daniel Senie
<ftp://ftp.isi.edu/in-notes/rfc2267.txt>

[14] ngrep
<http://www.packetfactory.net/ngrep/>

[15] rid
<http://theorygroup.com/Software/RID>

[16] Dan Farmer & Wietse Venema's class on computer forensic analysis
<http://www.fish.com/security/forensics.html>

[17] tcpdump
<ftp://ftp.ee.lbl.gov/tcpdump.tar.Z>

[18] snort
<http://www.clark.net/~roesch/security.html>

[19] netcat ("nc"), Hobbit
<http://packetstorm.securify.com/UNIX/netcat/nc110.tgz>

[20] Source code for mstream
<http://securityfocus.com/templates/archive.pike?list=82&date=2000-04-29&thread=200004291748.TAA13203@lobeda.jena.thur.de>

Appendix B - Example snort rules for detecting mstream

```
alert UDP any any -> any 6838 (msg: "IDS100/ddos-mstream-agent-to-handler"; content: "newserver";  
)  
alert UDP any any -> any 10498 (msg: "IDS101/ddos-mstream-handler-to-agent"; content: "stream/"; )  
alert UDP any any -> any 10498 (msg: "IDS102/ddos-mstream-handler-ping-to-agent" ; content:  
"ping";)  
alert UDP any any -> any 10498 (msg: "IDS103/ddos-mstream-agent-pong-to-handler" ; content:  
"pong";)  
alert TCP any any -> any 12754 (msg: "IDS109/ddos-mstream-client-to-handler"; flags: S;)  
alert TCP any 12754 -> any any (msg: "IDS110/ddos-mstream-handler-to-client"; content: ">"; flags:  
AP;)
```

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

alert TCP any any -> any 15104 (msg: "IDS111/ddos-mstream-client-to-handler"; flags: S;)
alert TCP any 15104 -> any any (msg: "IDS112/ddos-mstream-handler-to-client"; content: ">"; flags: AP;)

Appendix C - Rid templates for detecting mstream

```
start mstream-wild
  send udp dport=10498 data="ping"
  recv udp dport=6838 data="pong" nmatch=2
end mstream-wild
start mstream-published
  send udp dport=7983 data="ping"
  recv udp dport=9325 data="pong" nmatch=2
end mstream-published
```

Appendix D - Initial Intrusion and Concealment on Agent system

Examination of an agent system, and interviewing the owner, identified what looks like possibly two separate compromises. One sometime before March 31, 2000 (a password file entry for "inertia" existed until removed by the system owner on April 1), the transfer and installation of a rootkit (Irk4) and DDoS agent ("rpc.wall") on April 13 16:02 (all times, unless otherwise noted, are US/Pacific, or GMT-0700), and the marks of an ADM named attack and login on April 15 05:55. System logs had been deleted and/or scrubbed, so log evidence was not useful in determining what occurred on the system.

Some of these activities can still be identified using the "mactime" program, part of Dan Farmer and Wietse Venema's "Coroner's Toolkit" [16]. [The date "100" is, obviously, a Perl Y2K bug in the "mactime" program. This software was graciously provided by Dan Farmer, even though it is not publicly available at this time. It definitely shows how useful Unix forensic analysis tools can be.]

```
Apr 13 100 16:02:42 12060 .aa -rwxr-xr-x root/www root /bin/chown
12660 m.m -r-sr-xr-x root/www bin /bin/login
Apr 13 100 16:02:43 2048 mcmc drwxr-xr-x root/www root /bin
12660 cc -r-sr-xr-x root/www bin /bin/login
168748 .a. -rwxr-xr-x root/www root /usr/bin/as
64796 .a. -rwxr-xr-x root/www root /usr/bin/egcs
64796 .a. -rwxr-xr-x root/www root /usr/bin/gcc
64796 .a. -rwxr-xr-x root/www root /usr/bin/i386-redhat-linux-gcc
168496 .a. -rwxr-xr-x root/www root /usr/bin/ld
12656 m.c -rws--x--x root/www root /usr/bin/old
12656 m.c -r-xr-xr-x root/www bin /usr/bin/xstat
2315 .a. -rw-r--r-- root/www root /usr/include/_G_config.h
1313 .a. -rw-r--r-- root/www root /usr/include/alloca.h
```

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

```
4090 .a. -rw-r--r-- root/www root /usr/include/arpa/inet.h
3451 .a. -rw-r--r-- root/www root /usr/include/bits/byteswap.h
13327 .a. -rw-r--r-- root/www root /usr/include/bits/confname.h
168 .a. -rw-r--r-- root/www root /usr/include/bits/endian.h
2283 .a. -rw-r--r-- root/www root /usr/include/bits/errno.h
5107 .a. -rw-r--r-- root/www root /usr/include/bits/fcntl.h
4647 .a. -rw-r--r-- root/www root /usr/include/bits/in.h
3406 .a. -rw-r--r-- root/www root /usr/include/bits/posix_opt.h
2842 .a. -rw-r--r-- root/www root /usr/include/bits/select.h
4673 .a. -rw-r--r-- root/www root /usr/include/bits/sigset.h
1716 .a. -rw-r--r-- root/www root /usr/include/bits/sockaddr.h
9033 .a. -rw-r--r-- root/www root /usr/include/bits/socket.h
1297 .a. -rw-r--r-- root/www root /usr/include/bits/stdio_lim.h
2015 .a. -rw-r--r-- root/www root /usr/include/bits/time.h
4673 .a. -rw-r--r-- root/www root /usr/include/bits/types.h
1781 .a. -rw-r--r-- root/www root /usr/include/bits/uio.h
1798 .a. -rw-r--r-- root/www root /usr/include/errno.h
2481 .a. -rw-r--r-- root/www root /usr/include/fcntl.h
4579 .a. -rw-r--r-- root/www root /usr/include/features.h
9433 .a. -rw-r--r-- root/www root /usr/include/getopt.h
5861 .a. -rw-r--r-- root/www root /usr/include/gnu/stubs.h
973 .a. -rw-r--r-- root/www root /usr/include/libio.h
10291 .a. -rw-r--r-- root/www root /usr/include/netdb.h
17327 .a. -rw-r--r-- root/www root /usr/include/netinet/in.h
10779 .a. -rw-r--r-- root/www root /usr/include/netinet/in_sysm.h
1591 .a. -rw-r--r-- root/www root /usr/include/netinet/ip.h
9086 .a. -rw-r--r-- root/www root /usr/include/netinet/tcp.h
4855 .a. -rw-r--r-- root/www root /usr/include/rpc/netdb.h
2550 .a. -rw-r--r-- root/www root /usr/include/stdint.h
6467 .a. -rw-r--r-- root/www root /usr/include/stdio.h
20816 .a. -rw-r--r-- root/www root /usr/include/stdlib.h
27654 .a. -rw-r--r-- root/www root /usr/include/string.h
13245 .a. -rw-r--r-- root/www root /usr/include/strings.h
2104 .a. -rw-r--r-- root/www root /usr/include/sys/cdefs.h
4932 .a. -rw-r--r-- root/www root /usr/include/sys/select.h
3359 .a. -rw-r--r-- root/www root /usr/include/sys/socket.h
7996 .a. -rw-r--r-- root/www root /usr/include/sys/sysmacros.h
1577 .a. -rw-r--r-- root/www root /usr/include/sys/time.h
5337 .a. -rw-r--r-- root/www root /usr/include/sys/types.h
5299 .a. -rw-r--r-- root/www root /usr/include/sys/uio.h
1907 .a. -rw-r--r-- root/www root /usr/include/time.h
9314 .a. -rw-r--r-- root/www root /usr/include/unistd.h
36708 .a. -rw-r--r-- root/www root /usr/lib/crtn.o
874 .a. -rw-r--r-- root/www root /usr/lib/gcc-lib/i386-redhat-linux/egcs-2.91.66/cc1
1446620 .a. -rwxr-xr-x root/www root /usr/lib/gcc-lib/i386-redhat-linux/egcs-
2.91.66/collect2
46816 .a. -rwxr-xr-x root/www root /usr/lib/gcc-lib/i386-redhat-linux/egcs-
2.91.66/crtend.o
88444 .a. -rwxr-xr-x root/www root /usr/lib/gcc-lib/i386-redhat-linux/egcs-2.91.66/cpp
1424 .a. -rw-r--r-- root/www root /usr/lib/gcc-lib/i386-redhat-linux/egcs-
2.91.66/crtend.o
5794 .a. -rw-r--r-- root/www root /usr/lib/gcc-lib/i386-redhat-linux/egcs-
2.91.66/include/stdarg.h
```

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

```
9834 .a. -rw-r--r-- root/www root /usr/lib/gcc-lib/i386-redhat-linux/egcs-
2.91.66/include/stddef.h
770000 .a. -rw-r--r-- root/www root /usr/lib/gcc-lib/i386-redhat-linux/egcs-
2.91.66/libgcc.a
1957 .a. -rw-r--r-- root/www root /usr/lib/gcc-lib/i386-redhat-linux/egcs-2.91.66/specs
178 .a. -rw-r--r-- root/www root /usr/lib/libc.so
69638 .a. -rw-r--r-- root/www root /usr/lib/libc_nonshared.a
6162 .a. -rw-r--r-- 1046 squid /usr/src/linux/include/asm-i386/errno.h
1492 .a. -rw-r--r-- 1046 squid /usr/src/linux/include/asm-i386/socket.h
277 .a. -rw-r--r-- 1046 squid /usr/src/linux/include/asm-i386/sockios.h
305 .a. -rw-r--r-- 1046 squid /usr/src/linux/include/linux/errno.h
Apr 13 100 16:02:44 702 mcmc -rwxr-xr-x root/www root /etc/rc.d/rc.local
1024 mcmc drwxr-xr-x root/www root /root/.ncftp
9 mcmc lrwxrwxrwx root/www root /root/.ncftp/history
9 mcmc lrwxrwxrwx root/www root /root/.ncftp/log
9 mcmc lrwxrwxrwx root/www root /root/.ncftp/trace
29696 m.c drwxr-xr-x root/www root /usr/bin
17016 m.c -rwxr-xr-x root/www root /usr/bin/rpc.wall
8460 .a. -rw-r--r-- root/www root /usr/lib/crt1.o
1124 .a. -rw-r--r-- root/www root /usr/lib/crti.o
1892 .a. -rw-r--r-- root/www root /usr/lib/gcc-lib/i386-redhat-linux/egcs-
2.91.66/crtbegin.o
...
Apr 15 100 05:55:09 1024 mcmc drwxr-xr-x root/www root /var/named
1024 mcmc drwxr-xr-x root/www root /var/named/ADMROCKS
Apr 15 100 05:56:19 20437 .a. -rwxr-xr-x root/www root /usr/sbin/tcpd
Apr 15 100 05:56:20 34 .aa -rw-r--r-- root/www root /usr/libexec/awk/addy.awk
35628 .a. -rwxr-xr-x root/www root /usr/sbin/in.telnetd
Apr 15 100 05:56:26 159576 .a. -rwxr-xr-x root/www root /usr/bin/pico
975 .a. -rw-r--r-- root/www root /usr/share/terminfo/v/vt200
975 .a. -rw-r--r-- root/www root /usr/share/terminfo/v/vt220
-----
```

>From this listing, the following observations can be made:

- o On April 13 at 16:02, /bin/login was created and /bin/chown run.
- o At the same time, the compile was run (gcc/egcs) and /bin/old and /bin/xstat created. Access times on the .h files listed show the program being compiled uses network libraries.
- o Next, the /etc/rc.d/rc.local file is modified to include the line "/usr/bin/rpc.wall" at the end, thus re-starting the agent on each reboot. ncftp logging files are modified (they were deleted and turned into links to /dev/null to disable logging of ncftp file transfers.)
- o The program /usr/bin/rpc.wall was modified, and C runtime libraries accessed, which implies it was run. (This cannot be confirmed because the rootkit prevented the program from being seen by the administrator, who also rebooted the system and restarted it, thus modifying the /usr/bin/rpc.wall access date.)

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

o On April 15 at 05:55, it appears that the ADM named buffer overrun exploit was used against this system, followed within the minute access to a tcpd wrapped service which invoked in.telnetd. The rootkit configuration file /usr/libexec/awk/addy.awk was also accessed. [It is not clear if this was a just a coincidental second (third?) intrusion attempt.]

o Six seconds later /usr/bin/pico was run and the vt200 terminfo definition was accessed. Since the trojan version of login contains the string "vt200", this confirms the backdoor that was installed two days earlier was used to gain root access.

While the /etc/passwd entry for the "inertia" account had been deleted by the administrator of the system, the /etc/shadow entry (original modification date not known) was not:

```
-----  
inertia:iUCNir1cd8pl2:::-----  
-----
```

The password, "hi", was cracked in a fraction of a second.

Strings in "/bin/login" show the classic trojan horse signs of a path to a non-standard program ("/usr/bin/xstat") and embedded terminal type ("vt200") that triggers a root shell:

```
-----  
. . .  
login  
/bin/sh  
/usr/bin/xstat  
TERM  
bcshjvmudzwxftejk  
vt200  
%s=%s  
init.c  
. . .  
-----
```

Strings in "/bin/ls" and "/bin/ps" show the names of the rootkit configuration files to be "/usr/libexec/awk/files.awk" and "/usr/libexec/awk/ps.awk" (respectively). Another configuration file is seen above, "/usr/libexec/awk/addy.awk".

The big tipoff that this is a standard Linux rootkit is the compiler inserted debugging information that includes the (edited) path to the source file, which had better not be a valid path on Red Hat's development system:

```
-----  
. . .  
ls.c  
/home/XXXXX/stuff/lrk4/fileutils-3.13/src/  
-----
```

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

```
gcc2_compiled.  
int:t1=r1;-2147483648;2147483647;  
char:t2=r2;0;127;
```

...

Since telnetd was not otherwise used on the system (instead SSH was used for remote login), the intruder placed an entry (date unknown) in /etc/inetd.conf to run in.telnetd as service "working":

```
...  
#finger stream tcp      nowait root    /usr/sbin/tcpd  in.fingerd  
#cfinger stream        tcp      nowait root    /usr/sbin/tcpd  in.cfingerd  
#systat stream tcp     nowait guest   /usr/sbin/tcpd  /bin/ps -auwx  
#netstat               stream tcp   nowait guest   /usr/sbin/tcpd  /bin/netstat    -f inet  
working                stream tcp   nowait root    /usr/sbin/tcpd  in.telnetd  
...
```

The following entry was added (date unknown) to the /etc/services file for this service:

```
working                1120/tcp      # Kerberos working daemon
```

This service port is shown in (edited) "lsof" output as listening:

```
inetd  353  root  4u  IPv4  375      UDP *:talk  
inetd  353  root  5u  IPv4  376      UDP *:ntalk  
inetd  353  root  6u  IPv4  377      TCP *:working (LISTEN)  
inetd  353  root  8u  IPv4  378      TCP *:time (LISTEN)  
inetd  353  root  10u IPv4  379      UDP *:time  
inetd  353  root  11u IPv4  380      TCP *:auth (LISTEN)  
inetd  353  root  12u IPv4  381      TCP *:linuxconf (LISTEN)
```

It was noted earlier that the trojan version of /bin/login allows access using a terminal type of "vt200" (as seen by strings in the binary), and that the mactime output shows the termcap entries for vt200 are accessed right after the backdoor in.telnetd was accessed. This places the last login to the system using this backdoor at approximately 05:56 on April 15 (per the system's clock). This is within the time window of one of the attacks logged by the network operations engineers (Apr 15 04:11:35 to 07:32:22).

Appendix E - Analysis of handler system from Indiana University

[Note - this, too, was edited prior to publication.]

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

On 20 April 2000 the IT Security Office at Indiana University was contacted by Dave Dittrich of the University of Washington (and subsequently by Penn State). A system had been identified attempting to use an agent similar to Trinoo to initiate denial of service attacks using spoofed source addresses. The agent binary contained an IP address for a host on one of IU's networks.

Our first step was to have our networks group block all traffic to and from this IP address and log any inbound attempts. The main intention was to prevent the machine from causing more trouble, but we also wanted to keep the intruder from closing up shop once our presence was known. We avoided rebooting the machine and making physical changes to the network (e.g., unplugging the LAN cable) in case the software running on the machine was able to detect such events and delete itself.

Once the system was isolated from the rest of the network, we ran the lsof program, the output of which can be found at the end of this report.

This output indicates a process rpc.wall listening on several UDP ports, including 6838, to which Mr. Dittrich had noted his agent process had been sending packets. The /usr/bin/... has been opened for reading hundreds of times. This file contained a list of IP addresses, enciphered by a simple ASCII rotation. These IP addresses seem to be those of agent servers controlled by this master. Indeed, the identified agent system was on the list. We found a similar file, /dev/grab/..., which we think might have been used by another master server.

An example line from one of these files might look like

```
ckd`chj`cc`jb<
```

The cipher merely adds 50 to the ASCII value of each character, so the resulting IP is

```
192.168.11.80
```

Such a file can be translated by passing it through the UNIX command

```
tr 'b-k' '0-9.' | sed 's/<$/'
```

Using this translation, we found 76 IP addresses on the system, presumably all DDoS slave agents.

Source code for the rpc.wall program was found by searching through the raw disk. The attacker had done a good job of deleting the original source file but failed to use the -pipe option to gcc. Preprocessed code was therefore written to a temporary file to be passed to the compiler proper. Though this temporary file was

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

deleted from the filesystem by the compiler, the data itself remained intact on the disk.

Analyzing the source code of this program, we found it can perform "stream" as well as "mstream" attacks. An mstream attack is apparently a simultaneous stream attack on multiple IPs. We believe this added functionality could result in larger-scale DDoS attacks than we have seen before.

The source code found is included at the end of this report.

At this point we felt we had most of the important evidence, so we used the dd program to copy the entire filesystem over the network to my notebook. From there we could mount the file readonly and examine it later. Here's what we have found so far:

ROOT KIT

A root kit called flukek.tgz was found in /bin. It contains replacements for, among many others, cron, in.timed, inetd, login, named, passwd, rshd, syslogd, tcpd. The following files were added to /bin, /sbin, /usr/bin, and /usr/sbin since the system was installed. Nothing appeared to have been added to /usr/libexec or anything under /usr/local. Times listed are inode change times. The file /usr/bin/old appears to be the old login binary.

```
-rw-r--r-- 1 root wheel 2387704 Dec 18 16:37 bin/flukek.tgz
-r-sr-xr-x 1 root daemon 12656 Apr 13 23:39 bin/login
-rw-r--r-- 1 root wheel 1401 Apr 24 12:11 usr/bin/...
-rwsr-xr-x 1 root wheel 20164 Mar 31 14:50 usr/bin/old
-rwxr-xr-x 1 root wheel 33610 Apr 13 23:37 usr/bin/rpc.wall
-rwxr-xr-x 1 root wheel 32727 Apr 5 21:56 usr/bin/xf5
-r-xr-xr-x 1 root daemon 20164 Mar 31 14:50 usr/bin/xstat
-rwxr-xr-x 1 root wheel 111500 Mar 31 20:09 usr/sbin/dnskeygen
-rwxr-xr-x 1 root wheel 266712 Mar 31 20:09 usr/sbin/irpd
-rwxr-xr-x 1 root wheel 528612 Mar 31 20:09 usr/sbin/named
-rwxr-xr-x 1 root wheel 7166 Mar 31 20:09 usr/sbin/named-bootconf
-rwxr-xr-x 1 root wheel 285076 Mar 31 20:09 usr/sbin/named-xfer
-rwxr-xr-x 1 root wheel 37056 Mar 31 20:09 usr/sbin/ndc
```

No libraries, kernel modules, or PAM modules appear to have been replaced.

ACCOUNTS ADDED

The /etc/passwd and /etc/shadow files were recently modified:

```
-rw-r--r-- 1 root wheel 849 Feb 17 00:57 /mnt/etc/passwd
-rw----- 1 root wheel 884 Feb 17 00:57 /mnt/etc/passwd-
-r----- 1 root wheel 794 Feb 17 00:57 /mnt/etc/shadow
-r----- 1 root wheel 658 Nov 15 10:07 /mnt/etc/shadow-
```

The backup of /etc/shadow made 15 November gives us a clue what

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

accounts were added:

```
www:MyjKA0KGHplq6:11004:0:99999:7::  
login1:MyjKA0KGHplq6:11004:0:99999:7::  
web:af47L/OTL7K6.:11004:0:99999:7::  
x::11004:0:99999:7::
```

I have not yet attempted to crack these passwords. I did try to log in as "x" but was denied access.

INETD SERVICE ADDED

Inspection of /etc/services and /etc/inetd.conf reveals a service called "a", running in.telnetd on TCP port 1111, has been added.

/etc/services:

```
a 1111/tcp
```

/etc/inetd.conf

```
a stream tcp nowait root /usr/sbin/tcpd in.telnetd
```

I haven't attempted to connect to it yet.

LOG FILES

The following three files were the only log files found to have any pertinent information.

./bash_history:

```
nslookup  
cd /bin  
w  
ps x  
ftp 192.168.0.1 21  
w
```

/var/log/secure:

```
Mar 29 18:39:18 herc in.ftpd[824]: connect from 10.156.97.157  
Mar 29 19:29:15 herc in.ftpd[876]: connect from 10.156.97.111  
Mar 29 19:49:58 herc in.ftpd[882]: connect from 10.156.97.111  
Mar 29 19:50:21 herc in.ftpd[887]: connect from 10.156.97.111  
Mar 31 14:58:14 herc in.telnetd[4224]: connect from 10.54.115.105  
Apr 3 23:54:02 herc in.telnetd[10403]: connect from 10.72.135.165  
Apr 4 05:44:34 herc in.telnetd[11235]: connect from 10.103.26.127  
Apr 4 08:28:28 herc in.ftpd[11397]: connect from 10.31.68.158  
Apr 4 11:36:16 herc in.ftpd[11565]: connect from 10.31.68.158  
Apr 7 05:33:32 herc in.telnetd[16737]: connect from 10.22.82.6  
Apr 7 07:32:19 herc in.telnetd[16849]: connect from 10.22.82.6  
Apr 7 07:33:01 herc in.telnetd[16851]: connect from 10.22.82.6
```

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

```
Apr 7 07:33:20 herc in.ftpd[16852]: connect from 10.22.82.6
Apr 7 07:34:11 herc in.ftpd[16855]: connect from 10.22.82.6
Apr 7 07:35:22 herc in.ftpd[16859]: connect from 10.22.82.6
Apr 7 07:37:02 herc in.rlogind[16860]: connect from 10.22.82.2
Apr 7 07:37:12 herc in.fingerd[16863]: connect from 10.22.82.2
Apr 7 07:37:18 herc in.rexecd[16866]: connect from 10.22.82.2
Apr 7 07:37:22 herc in.rshd[16867]: connect from 10.22.82.2
Apr 7 07:37:24 herc in.telnetd[16868]: connect from 10.22.82.2
Apr 7 07:37:30 herc in.ftpd[16870]: connect from 10.22.82.2
Apr 8 13:53:02 herc in.ftpd[19028]: connect from 10.247.49.53
Apr 10 23:00:05 herc in.ftpd[23304]: connect from 10.8.148.36
Apr 10 23:07:51 herc in.ftpd[23347]: connect from 10.8.148.36
Apr 13 06:50:02 herc in.telnetd[27895]: connect from 10.215.99.125
Apr 13 10:52:27 herc in.ftpd[28170]: connect from 10.114.238.145
Apr 13 10:55:50 herc in.ftpd[28171]: connect from 10.114.238.145
Apr 13 11:02:39 herc in.ftpd[28217]: connect from 10.114.238.145
Apr 16 16:29:47 herc in.ftpd[1734]: connect from 10.161.208.34
Apr 16 16:30:10 herc in.ftpd[1737]: connect from 10.161.208.34
Apr 23 18:59:36 herc in.telnetd[14746]: connect from 10.27.211.234
Apr 24 17:02:03 herc in.telnetd[16505]: connect from 10.79.16.203
```

/var/log/wtmp (reverse chronological order):

```
root pts/2 :0 Mon Apr 24 18:05 still logged in
root pts/0 :0 Mon Apr 24 17:24 still logged in
ftp ftp XXXXXX-XXXXXXXXX Thu Apr 13 10:02 - 10:02 (00:00)
ftp ftp XXXXXX-XXXXXXXXX Thu Apr 13 09:55 - 09:56 (00:00)
ftp ftp XXXXXXXX-X.XXXXXX Mon Apr 10 22:07 - 22:09 (00:01)
ftp ftp XXX.XXX.82.6 Fri Apr 7 06:34 - 06:35 (00:00)
ftp ftp XXX.XXX.82.6 Fri Apr 7 06:33 - 06:34 (00:00)
ftp ftp XXXXX.XX-XXXXXXXXX Tue Apr 4 10:36 - 10:36 (00:00)
ftp ftp XXXXXXXXX-XXXX.XX Wed Mar 29 19:50 - 19:50 (00:00)
ftp ftp XXXXXXXXX-XXXX.XX Wed Mar 29 19:29 - 19:29 (00:00)
reboot system boot Wed Mar 29 16:17 (26+20:09)
```

OUTPUT OF lsof [edited]

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE	NODE	NAME
...								
inetd	378	root	cwd	DIR	3,1 1024	2 /		
inetd	378	root	rtd	DIR	3,1 1024	2 /		
inetd	378	root	txt	REG	3,1 18020	143469	/usr/sbin/inetd	
inetd	378	root	mem	REG	3,1 342206	30722	/lib/ld-2.1.1.so	
inetd	378	root	mem	REG	3,1 4016683	30729	/lib/libc-2.1.1.so	
inetd	378	root	mem	REG	3,1 251436	30766	/lib/libnss_nisplus-2.1.1.so	
inetd	378	root	mem	REG	3,1 364235	30742	/lib/libnsl-2.1.1.so	
inetd	378	root	mem	REG	3,1 243964	30760	/lib/libnss_files-2.1.1.so	
inetd	378	root	0u	CHR	1,3	2071	/dev/null	
inetd	378	root	1u	CHR	1,3	2071	/dev/null	
inetd	378	root	2u	CHR	1,3	2071	/dev/null	
inetd	378	root	4u	inet	506		TCP *:ftp (LISTEN)	
inetd	378	root	5u	inet	507		TCP *:telnet (LISTEN)	

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

```

inetd 378 root 6u inet 508 TCP *:shell (LISTEN)
inetd 378 root 7r FIFO 0,0 499 pipe
inetd 378 root 8u inet 509 TCP *:login (LISTEN)
inetd 378 root 9u CHR 5,1 2113 /dev/console
inetd 378 root 10u inet 510 TCP *:exec (LISTEN)
inetd 378 root 11u inet 511 UDP *:talk
inetd 378 root 12u inet 512 UDP *:ntalk
inetd 378 root 13u inet 513 TCP *:finger (LISTEN)
inetd 378 root 14u inet 514 TCP *:auth (LISTEN)
inetd 378 root 15u inet 515 TCP *:linuxconf (LISTEN)
inetd 378 root 16u inet 516 TCP *:a (LISTEN)
inetd 378 root 21w FIFO 0,0 499 pipe
...
rpc.wall 29108 root cwd DIR 3,1 31744 59393 /usr/bin
rpc.wall 29108 root rtd DIR 3,1 1024 2 /
rpc.wall 29108 root txt REG 3,1 33610 61172 /usr/bin/rpc.wall
rpc.wall 29108 root mem REG 3,1 342206 30722 /lib/ld-2.1.1.so
rpc.wall 29108 root mem REG 3,1 65996 30758 /lib/libnss_dns-2.1.1.so
rpc.wall 29108 root mem REG 3,1 4016683 30729 /lib/libc-2.1.1.so
rpc.wall 29108 root mem REG 3,1 243964 30760 /lib/libnss_files-2.1.1.so
rpc.wall 29108 root mem REG 3,1 251436 30766 /lib/libnss_nisplus-2.1.1.so
rpc.wall 29108 root mem REG 3,1 364235 30742 /lib/libnsl-2.1.1.so
rpc.wall 29108 root mem REG 3,1 251787 30764 /lib/libnss_nis-2.1.1.so
rpc.wall 29108 root mem REG 3,1 164797 30770 /lib/libresolv-2.1.1.so
rpc.wall 29108 root 0r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 1u CHR 136,1 3 /dev/pts/1
rpc.wall 29108 root 2u CHR 136,1 3 /dev/pts/1
rpc.wall 29108 root 3u inet 36066 TCP *:15104 (LISTEN)
rpc.wall 29108 root 4u CHR 136,1 3 /dev/pts/1
rpc.wall 29108 root 5u inet 36067 UDP *:6838
rpc.wall 29108 root 6r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 7r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 8r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 9r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 10r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 11r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 12r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 13r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 14r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 15r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 16r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 17r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 18r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 19r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 20r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 21u inet 38990 UDP *:2000
rpc.wall 29108 root 22r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 23u inet 38992 UDP *:2001
rpc.wall 29108 root 24r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 25u inet 38996 UDP *:2002
rpc.wall 29108 root 26r REG 3,1 1401 198670 /usr/bin/...
rpc.wall 29108 root 27u inet 38997 UDP *:2003
rpc.wall 29108 root 28r REG 3,1 1401 198670 /usr/bin/...

```

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

rpc.wall	29108	root	29u	inet	39001		UDP *:2004
rpc.wall	29108	root	30r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	31u	inet	39002		UDP *:2005
rpc.wall	29108	root	32r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	33u	inet	39003		UDP *:2006
rpc.wall	29108	root	34r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	35u	inet	39007		UDP *:2007
rpc.wall	29108	root	36r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	37u	inet	39011		UDP *:2008
rpc.wall	29108	root	38r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	39u	inet	39012		UDP *:2009
rpc.wall	29108	root	40r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	41u	inet	39016		UDP *:2010
rpc.wall	29108	root	42r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	43u	inet	39017		UDP *:2011
rpc.wall	29108	root	44r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	45u	inet	39018		UDP *:2012
rpc.wall	29108	root	46r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	47u	inet	39019		UDP *:2013
rpc.wall	29108	root	48r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	49u	inet	39020		UDP *:2014
rpc.wall	29108	root	50r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	51u	inet	39027		UDP *:2016
rpc.wall	29108	root	52r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	53u	inet	39028		UDP *:2017
rpc.wall	29108	root	54r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	55u	inet	39029		UDP *:2018
rpc.wall	29108	root	56r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	57u	inet	39033		UDP *:2019
rpc.wall	29108	root	58r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	59u	inet	39034		UDP *:2020
rpc.wall	29108	root	60r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	61u	inet	39038		UDP *:2021
rpc.wall	29108	root	62r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	63u	inet	39039		UDP *:2022
rpc.wall	29108	root	64r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	65r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	66u	inet	39101		UDP *:2024
rpc.wall	29108	root	67r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	68r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	69u	inet	39109		UDP *:2025
rpc.wall	29108	root	70r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	71u	inet	39129		UDP *:2026
rpc.wall	29108	root	72r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	73u	inet	39185		UDP *:2027
rpc.wall	29108	root	74r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	75u	inet	39186		UDP *:2028
rpc.wall	29108	root	76r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	77u	inet	39190		UDP *:2029
rpc.wall	29108	root	78r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	79u	inet	39195		UDP *:2030
rpc.wall	29108	root	80r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	81u	inet	39200		UDP *:2032

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

rpc.wall	29108	root	82r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	83u	inet	39204		UDP *:2033
rpc.wall	29108	root	84r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	85u	inet	39208		UDP *:2035
rpc.wall	29108	root	86r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	87u	inet	39215		UDP *:2036
rpc.wall	29108	root	88r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	89u	inet	39216		UDP *:2037
rpc.wall	29108	root	90r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	91u	inet	39223		UDP *:2038
rpc.wall	29108	root	92r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	93r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	94r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	95r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	96r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	97r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	98r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	99r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	100r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	101r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	102r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	103r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	104r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	105r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	106r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	107r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	108r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	109r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	110r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	111r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	112r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	113u	inet	39671		UDP *:2039
rpc.wall	29108	root	114r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	115r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	116r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	117r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	118r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	119r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	120r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	121r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	122r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	123r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	124r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	125r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	126r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	127r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	128r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	129r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	130r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	131r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	132r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	133r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	134r	REG	3,1	1401	198670 /usr/bin/...

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

rpc.wall	29108	root	135r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	136u	inet	41521		UDP *:2040
rpc.wall	29108	root	137r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	138r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	139r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	140r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	141u	inet	41532		UDP *:2041
rpc.wall	29108	root	142r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	143r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	144u	inet	41614		UDP *:2042
rpc.wall	29108	root	145r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	146u	inet	41615		UDP *:2045
rpc.wall	29108	root	147r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	148u	inet	41622		UDP *:2048
rpc.wall	29108	root	149r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	150u	inet	41681		UDP *:2051
rpc.wall	29108	root	151r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	152u	inet	41685		UDP *:2055
rpc.wall	29108	root	153r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	154u	inet	41690		UDP *:2056
rpc.wall	29108	root	155r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	156u	inet	41694		UDP *:2057
rpc.wall	29108	root	157r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	158u	inet	41696		UDP *:2058
rpc.wall	29108	root	159r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	160u	inet	41706		UDP *:2059
rpc.wall	29108	root	161r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	162u	inet	41707		UDP *:2060
rpc.wall	29108	root	163r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	164u	inet	41708		UDP *:2061
rpc.wall	29108	root	165r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	166u	inet	41715		UDP *:2062
rpc.wall	29108	root	167r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	168u	inet	41716		UDP *:2063
rpc.wall	29108	root	169r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	170r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	171r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	172r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	173r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	174r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	175r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	176r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	177r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	178r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	179r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	180r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	181r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	182r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	183r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	184r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	185r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	186r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	187r	REG	3,1	1401	198670 /usr/bin/...

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

rpc.wall	29108	root	241r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	242r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	243r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	244r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	245r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	246r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	247r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	248r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	249r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	250r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	251r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	252r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	253r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	254r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	255r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	256r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	257r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	258r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	259r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	260r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	261r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	262r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	263r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	264r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	265r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	266r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	267r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	268r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	269r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	270r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	271r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	272r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	273r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	274r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	275r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	276r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	277r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	278r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	279r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	280r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	281r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	282r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	283r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	284u	inet	53028			UDP *:2064
rpc.wall	29108	root	285r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	286u	inet	53029			UDP *:2065
rpc.wall	29108	root	287r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	288u	inet	53039			UDP *:2066
rpc.wall	29108	root	289r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	290u	inet	53040			UDP *:2067
rpc.wall	29108	root	291r	REG	3,1	1401	198670	/usr/bin/...
rpc.wall	29108	root	292u	inet	53048			UDP *:2068
rpc.wall	29108	root	293r	REG	3,1	1401	198670	/usr/bin/...

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

rpc.wall	29108	root	294u	inet	53052		UDP *:2069
rpc.wall	29108	root	295r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	296r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	297u	inet	53057		UDP *:2070
rpc.wall	29108	root	298r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	299u	inet	53284		UDP *:2071
rpc.wall	29108	root	300r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	301u	inet	53292		UDP *:2072
rpc.wall	29108	root	302r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	303r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	304u	inet	53607		UDP *:2073
rpc.wall	29108	root	305r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	306u	inet	53608		UDP *:2074
rpc.wall	29108	root	307r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	308u	inet	53671		UDP *:2075
rpc.wall	29108	root	309r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	310u	inet	53672		UDP *:2076
rpc.wall	29108	root	311r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	312u	inet	53681		UDP *:2077
rpc.wall	29108	root	313r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	314u	inet	53682		UDP *:2078
rpc.wall	29108	root	315r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	316u	inet	53688		UDP *:2079
rpc.wall	29108	root	317r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	318u	inet	53699		UDP *:2081
rpc.wall	29108	root	319r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	320u	inet	53707		UDP *:2082
rpc.wall	29108	root	321r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	322u	inet	53773		UDP *:2083
rpc.wall	29108	root	323r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	324u	inet	53774		UDP *:2084
rpc.wall	29108	root	325r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	326u	inet	53783		UDP *:2085
rpc.wall	29108	root	327r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	328u	inet	53784		UDP *:2086
rpc.wall	29108	root	329r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	330u	inet	53785		UDP *:2088
rpc.wall	29108	root	331r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	332u	inet	53794		UDP *:2089
rpc.wall	29108	root	333r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	334u	inet	53802		UDP *:2090
rpc.wall	29108	root	335r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	336u	inet	53810		UDP *:2091
rpc.wall	29108	root	337r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	338u	inet	53811		UDP *:2092
rpc.wall	29108	root	339r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	340r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	341u	inet	54862		UDP *:2093
rpc.wall	29108	root	342r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	343r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	344r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	345r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	346r	REG	3,1	1401	198670 /usr/bin/...

THE "MSTREAM" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

rpc.wall	29108	root	347r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	348u	inet	55930		UDP *:2094
rpc.wall	29108	root	349r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	350u	inet	56658		UDP *:2095
rpc.wall	29108	root	351r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	352r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	353r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	354u	inet	57339		UDP *:2096
rpc.wall	29108	root	355r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	356r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	357u	inet	57737		UDP *:2098
rpc.wall	29108	root	358r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	359u	inet	57876		UDP *:2099
rpc.wall	29108	root	360r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	361u	inet	57952		UDP *:2100
rpc.wall	29108	root	362r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	363r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	364r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	365u	inet	58274		UDP *:zephyr-clt
rpc.wall	29108	root	366r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	367u	inet	58855		UDP *:zephyr-hm
rpc.wall	29108	root	368r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	369u	inet	58860		UDP *:2105
rpc.wall	29108	root	370r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	371u	inet	58861		UDP *:2106
rpc.wall	29108	root	372r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	373u	inet	58873		UDP *:2107
rpc.wall	29108	root	374r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	375r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	376r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	377r	REG	3,1	1401	198670 /usr/bin/...
rpc.wall	29108	root	378r	REG	3,1	1401	198670 /usr/bin/...

SOURCE CODE FOR rpc.wall (mstream)

[Source code removed in light of publication. See [20]]

Andrew J. Korty, Lead Security Engineer
Information Technology Security Office
Office of the Vice President for Information Technology
Indiana University