# CERT Incident Note IN-2000-05: mstream Distributed DoS

The CERT Coordination Center publishes incident notes to provide information about incidents to the Internet community.

## "mstream" Distributed Denial of Service Tool

Date: Tuesday, May 2, 2000

### Overview

In late April 2000, we began receiving reports of sites finding a new distributed denial of service (DDOS) tool that is being called "mstream". The purpose of the tool is to enable intruders to utilize multiple Internet connected systems to launch packet flooding denial of service attacks against one or more target systems.

### Description

The "mstream" tool consists of a handler and an agent portion, much like previously known DDOS tools such as Trinoo. We have seen both the agent and the handler running as "rpc.wall" in binary form. The source code we have seen names the handler "master.c" and the agent "server.c".

The handler does not require administrative privileges and can function under a regular user login on a Unix system. The agent crafts forged packet headers and requires administrative (e.g., root) privileges to function.

The handler can be controlled remotely by one or more intruders using a password-protected interactive login to a running handler. Simple commands issued to the handler cause instructions to be sent to agents deployed on compromised systems. The communications between intruder and handler, and the handler and agents, are configurable at compile time and have varied significantly from incident to incident. The default protocol and destination socket numbers in source code recently released to the public are

```
intruder --------- 6723/tcp -> handler
handler  --------- 7983/udp -> agent
agent    --------- 9325/udp -> handler
```

It is important to note that any of these socket numbers can easily be altered to any value at compile-time by an intruder. For example, we have seen the handler compiled to listen for communications from the agent on UDP socket 6838 rather than 9325.

Agent binaries contain a list of handlers that are defined at compile-time by the intruder. The list of handlers is visible by running 'strings' against the agent binary. Here is an example of the output that has been edited to show easily identifiable items, including a sample list of mstream handlers.

```
192.168.1.2
192.168.3.4
192.168.5.6
Must be ran as root.
socket
bind
setsockopt
newserver
stream
mstream
ping
pong
fork
Forked into background, pid %d
```

When an agent is first executed, it will send a "newserver" message via UDP to all known handlers. Any handlers receiving the "newserver" message record the agent in a list of known agents. The IP address of the agent is written to a disk file using a simple ASCII rotation to obscure the IP address. The contents of the file can be recovered using the following command

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

IP addresses contained in this file may represent compromised hosts running mstream agents. The filename is configurable at compile-time by the intruder and we have seen various names used. Some examples we have seen are

- /usr/bin/...
- .sr [found in the directory containing the handler binary]
  The payload of a mstream network is a packet flooding denial of service attack using TCP packets with the ACK flag set. Other observed attributes of the payload packet headers include

  - random source IP address (all octets) for each packet
  - random source TCP socket number for the initial packet, then incrementing for each additional packet
  - random destination TCP socket number for each packet
  - IP header type-of-service (TOS) field set to "0x08" for each packet
  - IP header ID field random for initial packet, then incrementing for each additional packet
  - IP header time-to-live (TTL) field set to 255 for each packet
  - TCP header window size set to 16384 for each packet
  - TCP header sequence number random for initial packet, then incrementing for each additional packet
  - TCP header acknowledgment number set to 0 for each packet
  - no data in the data portion of the packet

The handler can be instructed to initiate an attack using the commands 'stream' or 'mstream'. However, in versions analyzed by the CERT/CC, the 'stream' command does not function as intended due to coding errors by the author. The apparent intent for 'stream' is to cause the handler to instruct all known agents to launch a TCP ACK flood against a single target IP address for a specified duration. Future versions of the tool may correctly implement this function. The 'mstream' command causes the handler to instruct all known agents to launch a TCP ACK flood against one or more target IP addresses.

Here is sample tcpdump output showing the attack pattern. In this example, handler.example.net is running the handler and agent.example.net is running the agent. The IP addresses 10.1.1.2 and 10.1.1.3 are the victims of the attack.

- intruder sending 'mstream 10.1.1.2:10.1.1.3 5' command to handler
  11:58:43.530004 lo > intruder.example.com.1044 > handler.example.net.6723: P 769187158:769187187(29) ack 770575957 win 31072 ( DF) (ttl 64, id 54036)

- handler echoing commands back to intruder
  11:58:43.530301 lo > handler.example.net.6723 > intruder.example.com.1044: P 1:45(44) ack 29 win 31072 (DF) (ttl 64, id 54037)

- handler sending 'mstream/10.1.1.2:10.1.1.3/5' command to agent
  11:58:43.530648 lo > handler.example.net.1035 > agent.example.net.7983: udp 28 (ttl 64, id 54038)

- agent beginning to attack two victim hosts; each source IP address and destination socket number is random
  11:58:43.531109 eth0 > xxx.xxx.xxx.xxx.2458 > 10.1.1.2.51479: . 2110392958:2110392958(0) ack 0 win 16384 [tos 0x8] (ttl 255, id 12979)
  11:58:43.531116 eth0 > xxx.xxx.xxx.xxx.2714 > 10.1.1.3.29405: . 2127170174:2127170174(0) ack 0 win 16384 [tos 0x8] (ttl 255, id 13235)
  11:58:43.531136 eth0 > xxx.xxx.xxx.xxx.2970 > 10.1.1.2.29837: . 2143947390:2143947390(0) ack 0 win 16384 [tos 0x8] (ttl 255, id 13491)
  11:58:43.531186 eth0 > xxx.xxx.xxx.xxx.3226 > 10.1.1.3.10268: . 2160724606:2160724606(0) ack 0 win 16384 [tos 0x8] (ttl 255, id 13747)
  11:58:43.531192 eth0 > xxx.xxx.xxx.xxx.3482 > 10.1.1.2.16764: . 2177501822:2177501822(0) ack 0 win 16384 [tos 0x8] (ttl 255, id 14003)
  11:58:43.531211 eth0 > xxx.xxx.xxx.xxx.3738 > 10.1.1.3.34732: . 2194279038:2194279038(0) ack 0 win 16384 [tos 0x8] (ttl 255, id 14259)

Output of 'strings' run against the handler binary produces some easily recognizable output. Here is an example:

```
You're too idle !
Connection from %s
newserver
New server on %s.
pong
Got pong number %d from %s
%s has disconnected (not auth'd): %s
Invalid password from %s.
Password accepted for connection from %s.
Lost connection to %s: %s
stream
Usage: stream
Unable to resolve %s.
stream/%s/%s
Streaming %s for %s seconds.
quit
%s has disconnected.
servers
Server file doesn't exist, creating ;)
The following ips are known servers:
help
commands
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
Currently Online:
Socket number %d        [%s]
ping
Pinging all servers.
mstream
Usage: mstream
MStreaming %s for %s seconds.
mstream/%s/%s
fork
Forked into background, pid %d
Caught SIGHUP, ignoring.
Caught SIGINT, ignoring.
Segmentation Violation, Exiting cleanly..
Caught unknown signal, This should not happen.
__exit_dummy_decl
_send2server
_sendtoall
```

## Impact

Distributed denial of service (DDOS) tools in general are capable of producing high magnitude packet flooding denial of service attacks. At the time of this writing, the "mstream" tool is capable of producing a severe denial of service condition against one or more victim sites, including sites being used as hosts for portions of a "mstream" DDOS network. However, at this time, "mstream" does not contain any functionality that significantly adds to the overall threat posed by DDOS tools in general.

Based on differences observed during analysis, we believe the code for "mstream" to be under active testing and development. The functionality of the tool may diverge from the functionality described in this Incident Note as the tool evolves.

## Solutions

The CERT/CC has previously published several resources discussing distributed denial of service tools. These resources contain advice on handling distributed denial of service attacks and the associated tools.

CA-2000-01, Denial-of-Service Developments
CA-99-17, Denial-of-Service Tools
IN-99-07, Distributed Denial of Service Tools

For general information about distributed system intruder tools, please see the results of the CERT-sponsored DSIT workshop from November 2, 1999.

Results of the Distributed-Systems Intruder Tools Workshop

An independent analysis of "mstream" was produced and made available by David Dittrich - University of Washington, George Weaver - Pennsylvania State University, Sven Dietrich - NASA Goddard Space Flight Center, and Neil Long - Oxford University. It is available from

http://staff.washington.edu/dittrich/misc/mstream.analysis.txt

**Authors**: Kevin Houle, Chad Dougherty