Can a TCP/IP Stack be killed programmatically?

Asked 16 years, 3 months ago Modified 15 years, 11 months ago

Viewed 2k times

3

Our server application is listening on a port, and after a period of time it no longer accepts incoming connections. (And while I'd love to solve this issue, it's not what I'm asking about here;)

The strange this is that when our app stops accepting connections on port 44044, so does IIS (on port 8080). Killing our app fixes everything - IIS starts responding again.

So the question is, can an application mess up the entire TCP/IP stack? Or perhaps, how can an application do that?

Senseless detail: Our app is written in C#, under .Net 2.0, on XP/SP2.

Clarification: IIS is not "refusing" the attempted connections. It is never seeing them. Clients are getting a "server did not respond in a timely manner" message (using the .Net TCP Client.)

c# tcp

Share

Improve this question

Follow

edited Sep 25, 2008 at 16:26

asked Sep 25, 2008 at 13:46

Gene

286 ● 1 ● 8

What about the opposite workaround: If you kill IIS, does your app start accepting connections and respond with traffic?
– benc Jan 7, 2009 at 7:41

7 Answers

Sorted by: Highest score (default)

5

You may well be starving the stack. It is pretty easy to drain in a high open/close transactions per second environment e.g. webserver serving lots of unpooled requests.

This is exhacerbated by the default TIME-WAIT delay - the amount of time that a socket has to be closed before being recycled defaults to 90s (if I remember right)

There are a bunch of registry keys that can be tweaked - suggest at least the following keys are created/edited

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\T

TcpTimedWaitDelay = 30
MaxUserPort = 65534
```

```
MaxHashTableSize = 65536
MaxFreeTcbs = 16000
```

Plenty of docs on MSDN & Technet about the function of these keys.

Share Improve this answer Follow

answered Sep 25, 2008 at 17:03

community wiki
stephbu

This would have helped us avoid the issue, for a while. We would have eventually ground to a halt though, even with this fix. Thanks for the help. – Gene Oct 9, 2008 at 18:34

4

You haven't maxed out the available port handles have you ?

`netstat -a`

I saw something similar when an app was opening and closing ports (but not actually closing them correctly).

Share Improve this answer Follow

answered Sep 25, 2008 at 13:50

RichS
3,147 ● 31 ● 27

Good suggestion. We've been using netstat and TCPView (technet.microsoft.com/en-us/sysinternals/bb897437.aspx) to diagnose, but nothing jumps out at us. – Gene Sep 25, 2008 at 15:27

netstat might not be the best tool because it does not provide much information about the creation of connections. – benc Jan 7, 2009 at 7:42

1

Use netstat -a to see the active connections when this happens. Perhaps, your server app is not closing/disposing of 'closed' connections.

Share Improve this answer Follow

answered Sep 25, 2008 at 13:52

leppie
117k • 18 • 200 • 300

1

Good suggestions from everyone, thanks for your help.

So here's what was going on: It turns out that we had several services competing for the same port, and most of the time the "proper" service would get the port. Occasionally a second service would grab the port away, and the first service would try to open a different port. From that time on, the services would keep grabbing new ports every time they serviced a request (since they weren't using their preferred ports) and eventually we would exhaust all available ports.

Of course, the actual question was: "Can an application mess up the entire TCP/IP stack?", and the answer to that question is: Yes. One way to do it is to listen on a whole bunch of ports.

Share Improve this answer

Follow

answered Oct 9, 2008 at 18:41

Gene

286 ● 1 ● 8

So you had an infinite listener problem? That should show up in netstat -a, unless the large number of listeners was causing a stack-level problem and never actually opening the listener. – benc Jan 7, 2009 at 7:44

0

I guess the port number comment from RichS is correct.

Other than that, the TCP/IP stack is just a module in your operating system and, as such, can have bugs that might allow an application to kill it. It wouldn't be the first driver to be killed by a program.

(A tip to the hat towards Andrew Tanenbaum for insisting that operating systems should be modular instead of monolithic.)

Share Improve this answer

Follow

answered Sep 25, 2008 at 13:54

xmjx

1,153 ● 1 ● 7 ● 18

0

I've been in a couple of similar situations myself. A good troubleshooting step is to attempt a connection from the affected machine to good known destination that isn't at that moment experiencing any connectivity issues. If the connection attempt fails, you are very likely to get more interesting details in the error message/code. For

example, it could say that there aren't enough handles, or memory.

Share Improve this answer

Follow

answered Sep 25, 2008 at 22:20

Alexander

9,380 ● 2 ● 28 ● 23

0

From a support and sys admin standpoint, I have only seen this on the rarest of occasions (more than once), but it certainly can happen.

When you are diagnosing the problem, you should carefully eliminate the possible causes, rather than blindly rebooting the system at the first sign of trouble. I only say this because many customers I work with are tempted to do that.

Share Improve this answer

Follow

answered Jan 7, 2009 at 7:45

benc

2,011 ● 5 ● 35 ● 42