How to count number of ESTABLISHED state TCP connections to a Windows server

Author : admin

Even if you have the background of a Linux system administrator, sooner or later you will have have to deal with some Windows hosts, thus i'll blog in this article shortly on how the established TCP if it happens *you will have to administarte a Windows hosts or help a windows sysadmin noobie* :)

In Linux it is pretty easy to check the number of established conenctions, because of the wonderful command **wc** (word count). with a simple command like:

```
$ netstat -etna |wc -l
```

Then you will get the number of **active TCP connections** to the machine and based on that you can get an idea on how busy the server is.

But what if you have to deal with lets say a **Microsoft Windows 2012 /2019 / 2020 or 2022 Server**, assuming you logged in as Administrator and you see the machine is quite loaded and runs multiple

Native Windows Administrator common services such as **IIS / Active directory Failover Clustering, Proxy server etc.**
How can you identify the established number of connections via a simple command in cmd.exe?

1.Count ESTABLISHED TCP connections from Windows Command Line

Here is the answer, simply use netstat native windows command and combine it with find, like that and use the /i (ignores the case of characters when searching the string) /c (count lines containing the string) options

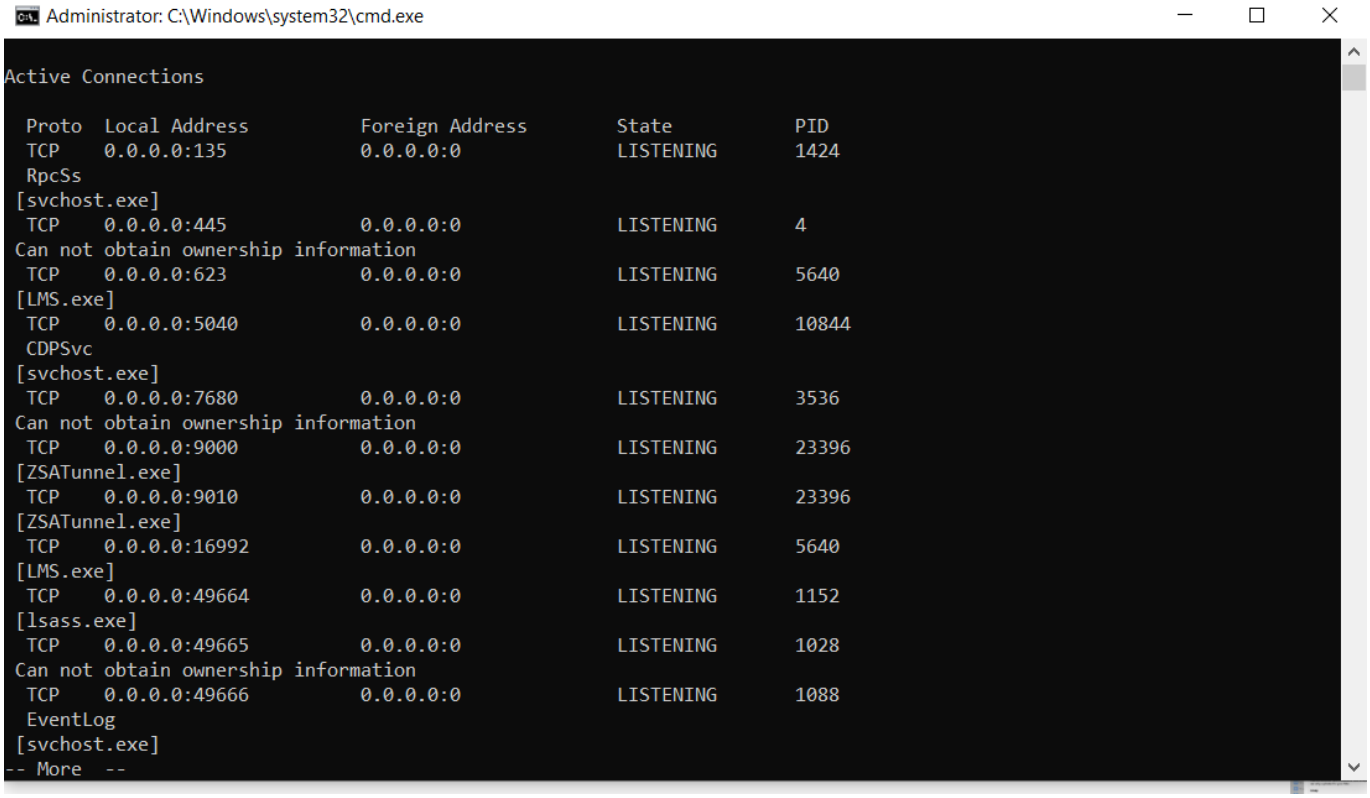
```
C:\Windows\system32>netstat -p TCP -n|  find /i "ESTABLISHED" /c
1268
```

Voila, here are number of established connections, only 1268 that is relatively low.
However if you manage Windows servers, and you get some kind of hang ups as part of the monitoring, it is a good idea to setup a script based on this simple command for at least **Windows Task Scheduler** (the equivallent of Linux's **crond** service) to log for Peaks in Established connections to see whether Server crashes are not related to High Rise in established connections.
*Even better if company uses **Zabbix / Nagios, OpenNMS** or other old legacy monitoring stuff like* Joschyd *even as of today 2024 used in some big of the **TOP IT companies** such as SAP (they were still using it about 4 years ago for their **SAP HANA Cloud)**, you can set the script to run and do a Monitoring template or Alerting rules to draw you graphs and Trigger Alerts if your connections hits a peak, then you at least might know your **Window**s server is under a "**Hackers**" **Denial of Service attack** or there is something happening on the network, like **Cisco Network Infrastructure Switch** flappings or whatever.*

*Perhaps an example script you can use if you decide to implement the little nestat established connection checks Monitoring in Zabbix is the one i've writen about in the previous article "*Calculate established connection from IP address with shell script and log to zabbix graphic*".*

2. Few Useful netstat options for the Windows system admin

```
C:\Windows\System32> netstat -bona
```



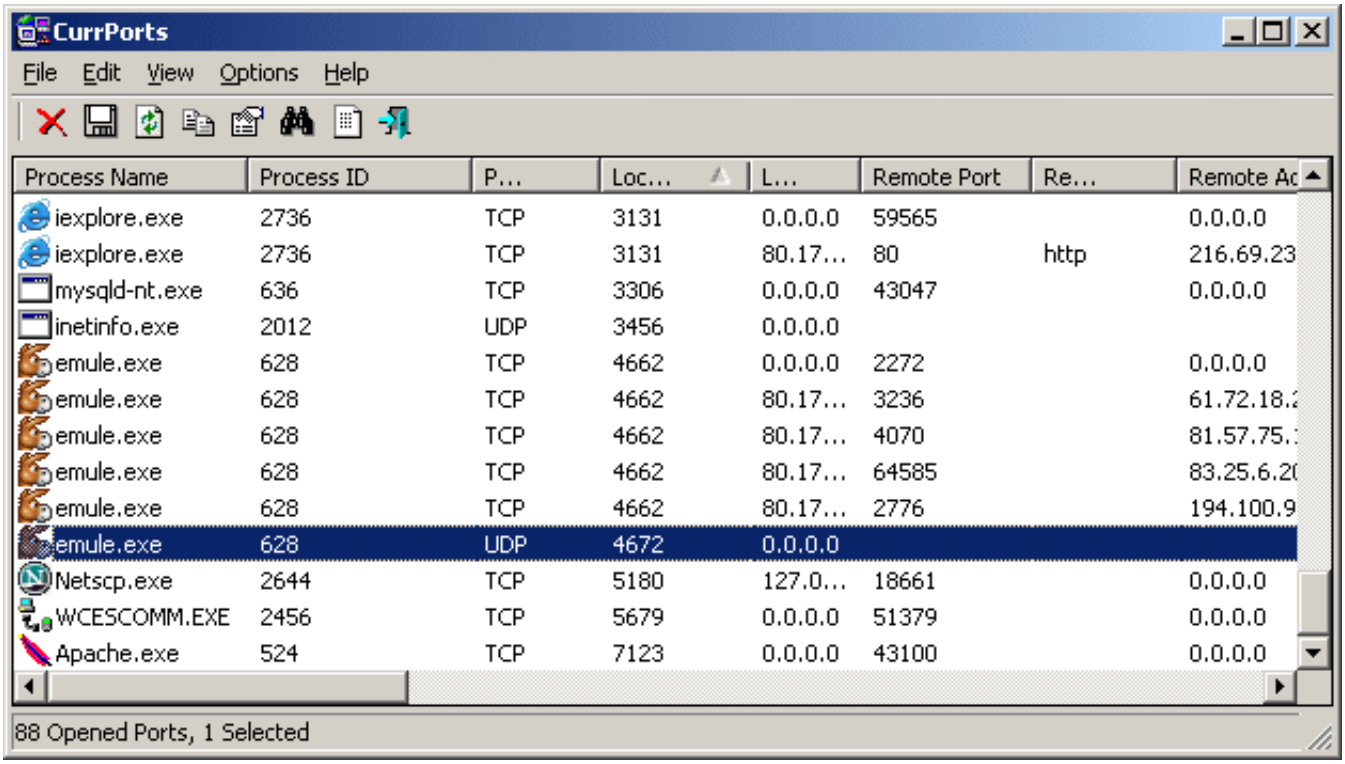
Cmd.exe will lists executable files, local and external IP addresses and ports, and the state in list form. You immediately see which programs have created connections or are listening so that you can find offenders quickly.

b - displays the executable involved in creating the connection.
o - displays the owning process ID.
n - displays address and port numbers.
a - displays all connections and listening ports.

As you can see in the screenshot, by using **netstat -bona** you get which process has binded to which local address and the Process ID PID of it, that is pretty useful in debugging stuff.

3. Use a Third Party GUI tool to debug more interactively connection issues

If you need to keep an eye in interactive mode, sometimes if there are issues CurrPorts tool **can be of a great help**



CurrPorts Tool own Description

CurrPorts is network monitoring software that displays the list of all currently opened **TCP/IP** and UDP ports on your local computer. For each port in the list, information about the process that opened the port is also displayed, including the process name, full path of the process, version information of the process (product name, file description, and so on), the time that the process was created, and the user that created it.
In addition, **CurrPorts allows you to close unwanted TCP connections, kill the process that opened the ports, and save the TCP/UDP ports information to HTML file** , **XML file**, or to tab-delimited text file.
CurrPorts also automatically mark with pink color suspicious **TCP/UDP** ports owned by unidentified applications (Applications without version information and icons).

Sum it up

What we learned is how to calculate number of established TCP connections from command line, useful for scripting, how you can use netstat to display the process ID and Process name that relates to a used Local / Remote TCP connections, and how eventually you can use this to connect it to some monitoring tool to periodically report High Peaks with TCP established connections (usually an indicator of servere

system issues).