

Forum:[Hack Forums](#)

Thread: [Limit CPU Usage in your application](#)

GoldyHacker - January 12, 2016, 11:06 am

I am trying for a very long time now to try and limit the CPU Usage of my program in a percentage which doesn't make it noticeable in order not to be easily found.

Could anyone please give me any code sample ?

Ninja - January 12, 2016, 11:21 am

How about you tell us what kind of application it is or show some code so we can see if you can do it more efficient? Would be a better solution than actually limiting the CPU usage.

Assuming it is a virus or a RAT, just limit what it does, dont send information from the victim PC to your PC or vice versa immediately but rather make a queue system. Make sure you do threading efficiently or don't thread at all. Or even inject your process into another process.

AnonCensored - January 12, 2016, 11:58 am

You could use [Thread.Sleep](#) In between you're code to slow it down then that sorta lower's the cpu usage also I believe running you're code in a timer also lower's it correct If im wrong

LordCoder - January 12, 2016, 2:24 pm

The way to do that would be limiting the CPU cores per action, or simply avoid doing the same task multiple times. For example, put some Timers (crappy way) to manage the times between both actions.

SL x TnT - January 12, 2016, 2:42 pm

Make it more efficient before limiting actions. Once you do that, you can determine how often you want certain methods to run.

If you want a more specific answer, you'll have to post your code.

Diamondo - January 12, 2016, 2:45 pm

Well, make your app multi threaded and use thread.sleep method if the processor pass your choose percent and so on...

Odus - January 12, 2016, 5:16 pm

(Jan 12, 2016 03:42 PM)SL x TnT Wrote: [\[CITE\]](#) Make it more efficient before limiting actions.

^

That's one of the biggest reasons why we care about clean, efficient code.

iLogic™ - January 13, 2016, 8:47 am

Write clean and efficient code. Also you can write your app multithreaded (perform more tasks parallel)

stxd4n0n - January 13, 2016, 9:17 am

Well, as everyone else said, you should do it more efficient, rather than limiting CPU usage..

But, if you are using Threads, you can try and set ThreadPriority, see more here:

msdn.microsoft.com/en-us/library/system.threading.thread.priority.aspx

He B TeMy - January 13, 2016, 10:03 am

You can't do what you asked literally, you should know this since you seem to be a dev.

GoldyHacker - January 17, 2016, 3:12 pm

(Jan 13, 2016 11:03 AM)He B TeMy Wrote: [\[CITE\]](#) You can't do what you asked literally, you should know this since you seem to be a dev.

To be true I'm just an hobbyist programmer who has little knowledge of VB..I have run a profiler on my program and I located that most the most CPU usage goes here :

```
For Each fdir As String In tempfileArrayList
    Select Case True
        Case fdir.Contains("Program Files")
        Case fdir.Contains("Temporary Internet Files")
        Case fdir.Contains("AppData")
        Case fdir.Contains("Default")
        Case fdir.Contains("Application Data")
        Case Else
            Variables.UpFilesArray.Add(fdir)
    End Select
Next
tempfileArrayList.Clear()
```

I understand the reason why all the CPU goes here... tempfileArrayList is an arraylist which contains the paths of some filtered files I want to search on the victims computer...The CPU checks each single path for each of those cases.I know it's inefficient code and I would like to hear ideas on how to make this piece of code more efficient

Edit: Also I would like to say sorry for not replying quickly and thanks for helping me guys :)

SL x TnT - January 17, 2016, 3:18 pm

(Jan 17, 2016 04:12 PM)GoldyHacker Wrote: [\[CITE\]](#) To be true I'm just an hobbyist programmer who has little knowledge of VB..I have run a profiler on my program and I located that most the most CPU usage goes here :

```
For Each fdir As String In tempfileArrayList
    Select Case True
        Case fdir.Contains("Program Files")
        Case fdir.Contains("Temporary Internet Files")
        Case fdir.Contains("AppData")
        Case fdir.Contains("Default")
        Case fdir.Contains("Application Data")
        Case Else
            Variables.UpFilesArray.Add(fdir)
    End Select
Next
tempfileArrayList.Clear()
```

I understand the reason why all the CPU goes here... tempfileArrayList is an arraylist which contains the paths of some filtered files I want to search on the victims computer...The CPU checks each single path for each of those cases.I know it's inefficient code and I would like to hear ideas on how to make this piece of code more efficient

Edit: Also I would like to say sorry for not replying quickly and thanks for helping me guys :)

Example data of what's inside tempfileArrayList?
Where are you getting the data for tempfileArrayList?
How large is tempfileArrayList?
How many items are in UpFilesArray when it's finished?
How long does it run on your computer?
What's your goal?

GoldyHacker - January 17, 2016, 3:33 pm

(Jan 17, 2016 04:18 PM)SL x TnT Wrote: [\[CITE\]](#) Example data of what's inside tempfileArrayList?
How large is tempfileArrayList?
How many items are in UpFilesArray when it's finished?
How long does it run on your computer?
What's your goal?

Thanks for the quick reply

Example of the contents of tempfileArrayList :

D:\Note 3\Screenshot\IMG_edited\2015-05-17-19-54-11.png
D:\Note 3\Screenshot\IMG_edited\2015-05-31-01-01-04.png
D:\Note 3\Screenshot\IMG_edited\2015-06-01-21-27-35.png
D:\Note 3\Screenshot\IMG_edited\2015-06-02-10-17-18.png
D:\Note 3\Screenshot\IMG_edited\2015-06-02-18-59-27.png
D:\Note 3\Screenshot\IMG_edited\2015-06-03-12-33-36.png
D:\Note 3\Screenshot\IMG_edited\2015-06-03-19-56-00.png
D:\Note 3\Screenshot\IMG_edited\2015-06-03-19-56-40.png
D:\Note 3\Screenshot\IMG_edited\2015-06-03-21-15-46.png
D:\Note 3\Screenshot\IMG_edited\2015-06-04-22-00-22.png
D:\Note 3\Screenshot\IMG_edited\2015-06-05-20-30-04.png
D:\Note 3\Screenshot\IMG_edited\2015-06-07-14-24-41.png
D:\Note 3\Screenshot\IMG_edited\2015-06-07-22-34-49.png
D:\Note 3\Screenshot\IMG_edited\2015-06-08-07-27-11.png
D:\Note 3\Screenshot\IMG_edited\2015-06-08-08-52-52.png
D:\Note 3\Screenshot\IMG_edited\2015-06-08-14-42-23.png
D:\Note 3\Screenshot\IMG_edited\2015-06-08-17-26-51.png
D:\Note 3\Screenshot\IMG_edited\2015-06-08-21-18-38.png
D:\Note 3\Screenshot\IMG_edited\2015-06-09-08-49-19.png
D:\Note 3\Screenshot\IMG_edited\2015-06-09-12-50-49.png
D:\Note 3\Screenshot\IMG_edited\2015-06-09-13-02-15.png

It depends on the computer I would say maximum 10,000 entries

1/10 of the tempfilearraylist the cases remove all the unwanted files or at least the majority of them

My PC is a hexa-core i7-5820k I would say briefly 10 seconds but I don't expect my potential victim to have that kind of a pc

All these files are being uploaded to an FTP Server

SL x TnT - January 17, 2016, 4:45 pm

(Jan 17, 2016 04:33 PM)GoldyHacker Wrote: [\[CITE\]](#) It depends on the computer I would say maximum 10,000 entries

1/10 of the tempfilearraylist the cases remove all the unwanted files or at least the majority of them

My PC is a hexa-core i7-5820k I would say briefly 10 seconds but I don't expect my potential victim to have that kind of a pc

All these files are being uploaded to an FTP Server
Are you sure?

Running 1m I'm getting ~700ms using a worse case equivalent of your code. Best case gives me ~50ms.
Proper multithreading gives me 300ms. Bad practices gives me 200ms.

That's running with the debugger.

Without debugger:

No threading = 450ms
Proper threading = 260ms
Improper threading = 130ms

You must be doing something that I'm missing. How are you adding to the list (it is a List and not an ArrayList, right?)

GoldyHacker - January 18, 2016, 1:26 am

(Jan 17, 2016 05:45 PM)SL x TnT Wrote: [\[CITE\]](#) Are you sure?

Running 1m I'm getting ~700ms using a worse case equivalent of your code. Best case gives me ~50ms.
Proper multithreading gives me 300ms. Bad practices gives me 200ms.

That's running with the debugger.

Without debugger:

No threading = 450ms
Proper threading = 260ms
Improper threading = 130ms

You must be doing something that I'm missing. How are you adding to the list (it is a List and not an ArrayList, right?)

The whole program runs for about 10 secs but it takes 40% of the CPU usage. I would like the program to operate in the most unnoticeable way

SL x TnT - January 18, 2016, 2:52 am

(Jan 18, 2016 02:26 AM)GoldyHacker Wrote: [\[CITE\]](#) The whole program runs for about 10 secs but it takes 40% of the CPU usage. I would like the program to operate in the most unnoticeable way
Then there's something else causing the high CPU. On a 5820k, 1 thread only uses ~8% of the CPU.

GoldyHacker - January 18, 2016, 3:28 am

(Jan 18, 2016 03:52 AM)SL x TnT Wrote: [\[CITE\]](#) Then there's something else causing the high CPU.
On a 5820k, 1 thread only uses ~8% of the CPU.
I mean on a normal PC , my 5820k uses 12% max for the program...It's quite demanding for

mainstream.Would you like the report of the profiler to check for yourself ?

SL x TnT - January 18, 2016, 3:43 am

(Jan 18, 2016 04:28 AM)GoldyHacker Wrote: [\[CITE\]](#) I mean on a normal PC , my 5820k uses 12% max for the program...It's quite demanding for mainstream.Would you like the report of the profiler to check for yourself ?

Nah, never used the profiler before.

From what you've said, the program takes 10 seconds to fully run. What does it do in those 10 seconds?

My only guess it goes through a list of operations to perform. If it does, it should be possible to output a completion message after every operation is complete. Find the operation that runs slow and post the entire code for it.

GoldyHacker - January 18, 2016, 3:54 am

(Jan 18, 2016 04:43 AM)SL x TnT Wrote: [\[CITE\]](#) Nah, never used the profiler before.

From what you've said, the program takes 10 seconds to fully run. What does it do in those 10 seconds?

My only guess it goes through a list of operations to perform. If it does, it should be possible to output a completion message after every operation is complete. Find the operation that runs slow and post the entire code for it.

The profiler shows where the CPU Usage goes to :

dirs : Directories from the system drive

extension : searches all files in these directories with an extension filter

Try

For Each found In (IO.Directory.GetFiles(dirs, extension,
IO.SearchOption.AllDirectories)) ' 89.8% of CPU goes here

Dim File As String = found

Dim infoReader As System.IO.FileInfo

infoReader = My.Computer.FileSystem.GetFileInfo(File) ' 7.7% goes here

Dim FileByte As String = infoReader.Length

Dim FileMegaByte As String = FileByte / 1024 / 1024

If FileMegaByte <= 15 Then

tempfileArrayList.Add(found)

End If

Next

Catch ex As UnauthorizedAccessException

Catch exx As IO.IOException

End Try

Next

For Each fdir As String In tempfileArrayList

Select Case True

Case fdir.Contains("Program Files")

Case fdir.Contains("Temporary Internet Files")

```
Case fdir.Contains("AppData")
Case fdir.Contains("Default")
Case fdir.Contains("Application Data")
Case Else
    Variables.UpFilesArray.Add(fdir)
End Select
Next
```

SL x TnT - January 18, 2016, 4:26 am

I recommend not using any VB methods.

Code My.Computer.FileSystem.GetFileInfo(File)
Small tests.

Creating a new FileInfo object = 2 second increase.
Getting the file size = 10x increase.

Remove that portion and test how fast it goes.

Other things:
You're checking the file size of every file you find.
Then you're removing them.

Switch it around. You told me that it removes 90% of the files. Why should you waste your time checking those sizes?

Better solution. Don't even check those directories. That should help the most depending on how long Directory.GetFiles runs.

GoldyHacker - January 18, 2016, 4:33 am

(Jan 18, 2016 05:26 AM)SL x TnT Wrote: [\[CITE\]](#) I recommend not using any VB methods.

Code My.Computer.FileSystem.GetFileInfo(File)
Small tests.

Creating a new FileInfo object = 2 second increase.
Getting the file size = 10x increase.

Remove that portion and test how fast it goes.

Other things:
You're checking the file size of every file you find.
Then you're removing them.

Switch it around. You told me that it removes 90% of the files. Why should you waste your time checking those sizes?

Better solution. Don't even check those directories. That should help the most depending on how long Directory.GetFiles runs.

Better way to check for the filesize ?

You have absolute right on the point where the check is made, I'll change it. Will post results

Edit : Made the changes..CPU Usage on the intense point dropped from 97 to 93 percent and on the reader point from 7 to 4 percent going to make some real world tests too