

Forum:[Hack Forums](#)

Thread: [check if process is already running](#)

BapeZ - September 29, 2018, 10:54 am

How can I do that? it needs to check it by the exact path, not by the name. Thanks for your help.

oxidebuffalo - September 29, 2018, 1:09 pm

U can use NtQueryInformationProcess w/ the ProcessImageFileName class. This will let you get the file name for any process you have sufficient access to and not just your own. See msdn for more infos.

```
Code Declare Function NtQueryInformationProcess Lib "ntdll.dll" ( _  
    ProcessHandle As IntPtr, _  
    ProcessInformationClass As Integer, _  
    ProcessInformation As IntPtr, _  
    ProcessInformationLength As Integer, _  
    ReturnLength As IntPtr _  
    ) As Integer
```

If you're trying to do this for your own process, then you can access the PEB and use the process parameter information ProcessParameters (_RTL_USER_PROCESS_PARAMETERS*) field.

```
Code typedef struct _RTL_USER_PROCESS_PARAMETERS {  
    ULONG MaximumLength;  
    ULONG Length;  
    ULONG Flags;  
    ULONG DebugFlags;  
    PVOID ConsoleHandle;  
    ULONG ConsoleFlags;  
    PVOID StandardInput;  
    PVOID StandardOutput;  
    PVOID StandardError;  
    CURDIR CurrentDirectory;  
    UNICODE_STRING DllPath;  
    UNICODE_STRING ImagePathName;  
    UNICODE_STRING CommandLine;  
    PVOID Environment;  
    ULONG StartingX;  
    ULONG StartingY;  
    ULONG CountX;  
    ULONG CountY;  
    ULONG CountCharsX;  
    ULONG CountCharsY;  
    ULONG FillAttribute;  
    ULONG WindowFlags;  
    ULONG ShowWindowFlags;  
    UNICODE_STRING WindowTitle;  
    UNICODE_STRING DesktopInfo;  
    UNICODE_STRING ShellInfo;  
    UNICODE_STRING RuntimeData;
```

```
RTL_DRIVE_LETTER_CURDIR CurrentDirectores[32];
ULONG EnvironmentSize;
} RTL_USER_PROCESS_PARAMETERS, *PRTL_USER_PROCESS_PARAMETERS;
```

The structure syntax is in C and not VB.NET. You'll have to convert it yourself appropriately.

The ImagePathName field of the structure will have a Buffer field since its a `_UNICODE_STRING` and that will contain the proper file name of the image for the process.

If you really need to do it from .NET APIs then you can but you'll learn something new if you do it manually like above. Which will season your dev skills.

Washi - October 1, 2018, 2:54 am

(Sep 29, 2018 01:09 PM)oxidebuffalo Wrote: [\[CITE\]](#) -snip-
I mean you could do that, but you can also just go the .NET route and obtain the path using [ProcessModule.FileName](#) on the mainmodule ([Process.MainModule](#)) of the processes with the same name as the current process. Sure you would learn something new if you do it manually, but other than that, not sure what benefits it really has.

PHP Code
<?php
Imports System.Reflection

Module Module1

```
Private Function IsAlreadyRunning() As Boolean
    Dim path = Assembly.GetExecutingAssembly().Location
    Dim processes = Process.GetProcessesByName(IO.Path.GetFileNameWithoutExtension(path))
))
    Dim currentProcess = Process.GetCurrentProcess()

    Return processes.Any(Function(p) currentProcess.Id <> p.Id AndAlso p.MainModule.FileName
= path)
End Function

Sub Main()
    If IsAlreadyRunning() Then
        Console.WriteLine("Already running!")
    Else
        Console.WriteLine("Not running!")
    End If

    Console.ReadKey()
End Sub

End Module
```

He B TeMy - October 1, 2018, 12:24 pm

(Oct 1, 2018 02:54 AM)Washi Wrote: [\[CITE\]](#) I mean you could do that, but you can also just go the .NET route and obtain the path using [ProcessModule.FileName](#) on the mainmodule ([Process.MainModule](#)) of the processes with the same name as the current process. Sure you would learn something new if you do it manually, but other than that, not sure what benefits it really has.

```
PHP Code
<?php
Imports System.Reflection
```

```
Module Module1
```

```
Private Function IsAlreadyRunning() As Boolean
    Dim path = Assembly.GetExecutingAssembly().Location
    Dim processes = Process.GetProcessesByName(IO.Path.GetFileNameWithoutExtension(path))
)
    Dim currentProcess = Process.GetCurrentProcess()

    Return processes.Any(Function(p) currentProcess.Id <> p.Id AndAlso p.MainModule.FileName
= path)
End Function

Sub Main()
    If IsAlreadyRunning() Then
        Console.WriteLine("Already running!")
    Else
        Console.WriteLine("Not running!")
    End If

    Console.ReadKey()
End Sub
```

```
End Module
```

You will get access denied if you try to access 64-bit processes with a 32-bit app when accessing Process.MainModule and there's no way around it on .net , even if you run a 64-bit app you will get access denied for PID 4 (System) and PID 0 (idle) , the best way is probably querying the winapi directly or WMI, <https://stackoverflow.com/a/3654195>

Washi - October 1, 2018, 12:44 pm

(Oct 1, 2018 12:24 PM)ff20 Wrote: [\[CITE\]](#) You will get access denied if you try to access 64-bit processes with a 32-bit app when accessing Process.MainModule and there's no way around it on .net , even if you run a 64-bit app you will get access denied for PID 4 (System) and PID 0 (idle) , the best way is probably querying the winapi directly or WMI, <https://stackoverflow.com/a/3654195>
I guess that is true, but then again, you are checking for processes that are running your application. If your application is 32 bit, then you will be opening 32 bit processes as well, and 64 bit processes otherwise. It would only fail when you share a process name with another application.

I guess better safe than sorry though. You could maybe wrap the retrieval of the file path in a try-catch statement. An exception can only be caught when the platforms mismatch, which already gives away the fact that the process you're trying to open is not the same application.

oxidebuffalo - October 1, 2018, 1:09 pm

(Oct 1, 2018 12:24 PM)ff20 Wrote: [\[CITE\]](#) even if you run a 64-bit app you will get access denied for PID 4 (System) and PID 0 (idle) You're always going to get STATUS_ACCESS_DENIED (NTSTATUS error code) returned from NtOpenProcess whilst trying to open a handle to a process like the "System" process.

The "System" process (PID of 4 reserved for it nowadays) is for ntoskrnl.exe.

