عنوان: **عبارات باقاعده و نیاز به Timeout**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۲/۲۳ ۱۸:۴۱:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

یکبار سعی کنید مثال ساده زیر را اجرا کنید:

```
using System;
using System.Text.RegularExpressions;

namespace RegexLoop
{
    class Program
    {
        static void Main(string[] args)
        {
            var emailAddressRegex = new Regex(@"^[A-Za-z0-9]([_\.\-]?[A-Za-z0-9]+)*\@[A-Za-z0-9]([_\.\-
]?[A-Za-z0-9]+)*\.[A-Za-z0-9]([_\.\-]?[A-Za-z0-9]+)*$|^$");
            if (emailAddressRegex.IsMatch("an.infinite.loop.sample.just_for.test"))
            {
                Console.WriteLine("Matched!");
            }

            var input = "The quick brown fox jumps";
            var pattern = @"([a-z ]+)*!";
            if (Regex.IsMatch(input, pattern))
            {
                Console.WriteLine("Matched!");
            }
        }
    }
}
```

پس از اجرا، برنامه هنگ خواهد کرد یا به عبارتی برنامه در یک حلقه بی‌نهایت قرار می‌گیرد (در هر دو مثال؛ اطلاعات بیشتر و آنالیز کامل در اینجا). بنابراین نیاز به مکانیزمی امنیتی جهت محافظت در برابر این نوع ورودی‌ها وجود خواهد داشت؛ مثلا یک Timeout . اگر تا 2 ثانیه به جواب نرسیدیم، اجرای Regex متوقف شود. تا دات نت 4، چنین timeout ایی پیش بینی نشده؛ اما در دات نت 4 و نیم آرگومانی جهت تعریف حداکثر مدت زمان قابل قبول اجرای یک عبارت باقاعده در نظر گرفته شده است (^) و اگر در طی مدت زمان مشخص شده، کار انجام محاسبات به پایان نرسد، استثنای RegexMatchTimeoutException صادر خواهد شد.

خیلی هم خوب. به این ترتیب کسی نمی‌تونه با یک ورودی ویژه، CPU Usage سیستم رو تا مدت زمان نامحدودی به 100 درصد برساند و عملا استفاده از سیستم رو غیرممکن کنه.

اما تا قبل از دات نت 4 و نیم چکار باید کرد؟ روش کلی حل این مساله به این ترتیب است که باید اجرای Regex را به یک ترد دیگر منتقل کرد؛ اگر مدت اجرای عملیات، از زمان تعیین شده بیشتر گردید، آنگاه می‌شود ترد را Abort کرد و به عملیات خاتمه داد. روش پیاده سازی و نحوه استفاده از آن‌را در ادامه ملاحظه خواهید نمود:

```
using System;
using System.Text.RegularExpressions;
using System.Threading;

namespace RegexLoop
{
    public static class TimedRunner
    {
        public static R RunWithTimeout<R>(Func<R> proc, TimeSpan duration)
        {
            using (var waitHandle = new AutoResetEvent(false))
```

```
            {
                var ret = default(R);
                var thread = new Thread(() =>
                {
                    ret = proc();
                    waitHandle.Set();
                }) { IsBackground = true };
                thread.Start();

                bool timedOut = !waitHandle.WaitOne(duration, false);
                waitHandle.Close();

                if (timedOut)
                {
                    try
                    {
                        thread.Abort();
                    }
                    catch { }
                    return default(R);
                }
                return ret;
            }
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            var emailAddressRegex = new Regex(@"^[A-Za-z0-9]([_\.\-]?[A-Za-z0-9]+)*\@[A-Za-z0-9]([_\.\-
]?[A-Za-z0-9]+)*\.[A-Za-z0-9]([_\.\-]?[A-Za-z0-9]+)*$|^$");
            if (TimedRunner.RunWithTimeout(
                 () => emailAddressRegex.IsMatch("an.infinite.loop.sample.just_for.test"),
                 TimeSpan.FromSeconds(2)))
            {
                Console.WriteLine("Matched!");
            }

            var input = "The quick brown fox jumps";
            var pattern = @"([a-z ]+)*!";
            if (TimedRunner.RunWithTimeout(() => Regex.IsMatch(input, pattern),
TimeSpan.FromSeconds(2)))
            {
                Console.WriteLine("Matched!");
            }
        }
    }
}
```

اینبار به هر کدام از عبارات باقاعده 2 ثانیه زمان برای اتمام کار داده شده است. در غیراینصورت مقدار پیش فرض خروجی متد فراخوانی شده، بازگشت داده می‌شود که در اینجا false است.