## 使用LordPE和Import REC脱壳



<u>虾仁炖猪心</u> <u>野 2020-04-18 17:35:46 发布</u> 分类专栏:<u>加壳脱壳</u> 版权声明:本文为博主原创文章,遵循<u>CC 4.0 BY-SA</u>版权协议,转载请附上原文出处链接和本声明。 本文链接:<u>https://blog.csdn.net/weixin\_43575859/article/details/105598559</u> 版权



加壳脱壳专栏收录该内容

3 篇文章 1 订阅 订阅专栏 本博客用到的资源链接放在博客末尾。

**LordPE**是一款能够抓取内存映像的用来脱壳的软件,它能够把一个进程的内存数据给Dump下来,然后保存为 文件。

光了解一个软件的用处用起来如果出问题还是会比较头疼,所以这里简单说一下LordPE软件的原理。

该软件其实也是调用了系统的API。它首先调用CreateToolhelp32Snapshot函数来获取一个系统中的进程的快照。然后再用Module32Next函数来循环获取每个进程的信息,进程的信息会被填到一个MODULEENTRY32结构里面,信息包括了该进程的映像基址,进程的映像大小,进程的句柄,进程的完整路径。当然还有一些其他的信息,不过这些我们不用管。

得到了进程的句柄,以及映像基址和大小之后,软件就可以对它进行操作了。软件会根据这个数据读取出进程 的内存数据,然后将数据保存到磁盘文件。

那么它为什么能脱壳呢?其实用它进行脱壳的过程,一般是我们先用其他软件将加壳程序调试到OEP处停止, 这个时候程序在内存中已经被壳给解压缩或者解密了,这个时候的进程的内存数据就是没加壳时候的数据。然 后我们将这些数据Dump下来保存为文件。得到的就是没加壳的原程序文件了。其实就是先让壳对程序自行解压 或解密,然后我们将解压或解密后的程序数据Dump下来保存为文件。

)[LordPE 豪华版] by yoda				_	
路径	PID	镜像基址		^	PE 编辑器
🗞 [系统]	00002E70	00000000	00000000		调试努由由能
🔏 [系统]	00002B44	00000000	00000000		
🔏 [系统]	00001A04	00000000	0000000		重建 PE
🔏 [系统]	00001AA8	00000000	00000000		合并
🗞 [系统]	000009A0	00000000	00000000		林大田久祝
🖧 [系统]	00002978	00000000	00000000		转仔服分替
🖧 [系统]	00002E24	00000000	00000000		选项
●️\$[系统]	00002224	00000000	00000000		
🖧 [系统]	00001708	00000000	00000000		
🚾 d:\一个文件夹\wps\wps office\11.1	00002298	00C70000	00113000		
🙆 d:\一个文件夹\wps\wps office\11.1	0000323C	00610000	000C8000		
🖧 [系统]	000011C8	00000000	00000000		
🖧 [系统]	0000196C	00000000	0000000		
🖧 [系统]	00002E04	00000000	00000000		
🖧 [系统]	0000120C	00000000	00000000		
A. [系始]	00003258	0000000	0000000	<u> </u>	
路径		镜像大小		<u> </u>	
△d:\一个文件夹\wps\wps office\11.1	00610000	000C8000			
🚳 c:\windows\system32\ntdll. dll	777 <b>E</b> 0000	0019A000			
🚳 c:\windows\system32\kernel32. dll	76D20000	000E0000			
🚳 c:\windows\system32\kernelbase. dll 👘	75900000	001FE000			
🚳 c:\windows\system32\apphelp. dll	71DF0000	0009F000			
🚳 c:\windows\system32\aclayers. dll	715D0000	00284000			
🛒 c:\windows\system32\msvcrt. dll	76C60000	000BF000			
🛒 c:\windows\system32\user32. dll	77300000	00197000			
🛒 c:\windows\system32\win32u. dll	76B60000	00017000			
🔄 c:\windows\system32\gdi32. dll	772D0000	00021000			
o:\windows\system32\gdi32full. dll	76100000	0015A000			
📉 c:\windows\system32\msvcp_win. dll	76E00000	0007C000			
🛒 c:\windows\system32\ucrtbase. dll	74FE0000	0011F000			
🛒 c:\windows\system32\shell32. dll	765E0000	0057A000			关于
🔄 c:\windows\system32\cfgmgr32. dll	75BC0000	0003B000			退出
a luindour leveton 32 Lebaara dll	75750000	00084000		in nativel	Mar Mar

这个是程序的界面,打开程序时要注意用管理员权限打开,不然可能用OD调试的进程在这里可能看不到。点击选项,然后调整设置成下面这样:



里面有个选项:从磁盘粘贴文件头。意思是Dump下来的数据里面的PE文件头是直接从磁盘原始文件中复制的。

当我们要Dump一个进程的内存数据的时候,右键目标进程,然后选择完整转存就可以了。

· · · · · · · ·				
🖧[系统]		00002E24	00000000	00000000
🔏 [系统]		00002224	00000000	0000000
🔏 [系统]		00001708	0000000	0000000
₩ d: \一个文件身	完整转存		070000	00113000
🙆 d: \一个文件扌	部分转存		610000	000C8000
💑[系统]			000000	00000000
🔧[系统]	区域转存		000000	00000000
🔧 [系统]	送报##方 <b>司</b>		000000	00000000
🔧 [系统]	処理物理が		000000	0000000
💐 [ୟkż]	优先权		> 000000	0000000
路径	007012		像大小	
🔽 d: \—个文件列	修正镜像大小		113000	
🚳 c:\windows\s			19A000	
🚳 c:\windows\s	载人到 PE 编辑器	(临时文件)	0E0000	
🚳 c:\windows\s	载入到 PE 编辑器…	(只读)	1FE000	
🚳 c:\windows\s			09F000	
🚳 c:\windows\s	终止进程		284000	
🚳 c:\windows\s	副新	E5	OBFOOO	
🚳 c:\windows\s	18004	and the test of the second	197000	
🗟 c:\windows\syste	m32\win32u. dll	76B60000	00017000	

但是,既然有这种Dump技术,那就肯定还有反Dump技术,因为LordPE软件是通过MODULEENTRY32结构 里面的数据来获取进程的信息的。所以,一些程序的壳会修改里面的modBaseSize和modBaseAddr字段,往 里面填入错误的值。而因为如果修改modBaseAddr的值,会让系统出现问题,所以一般修改的都 是modBaseSize字段。这样我们Dump下来的就是不完整的数据。

出现这种问题其实解决方法很简单,可以让软件直接到PE文件头里面的SizeOfImage字段读取映像的真正大小。做法就是右键目标几进程,然后选择修正镜像大小。然后再Dump就可以了。

加壳程序还可能用一种技术来防止用户Dump进程数据,就是在程序中使用VirtualProtect函数来设置PE文件头为不可读,这样我们用软件Dump进程的时候就会出现访问错误。在软件界面右键目标进程,然后点击区域转存,就出现了下图:

[转存区域]					x
地址	大小	保护	状态	类型	-
00000000	00010000	NOACCESS	FREE		
00010000	00010000	RW	COMMET	MAPPED	
00020000	00001000	RW	COMMET	PRIVATE	
00021000	0000F000	NOACCESS	FREE		
00030000	00004000	R	COMMET	MAPPED	
00034000	00004000		RESERVE	MAPPED	
00038000	00008000	NOACCESS	FREE		
00040000	0001B000	R	COMMET	MAPPED	
0005B000	00005000	NOACCESS	FREE		
00060000	00030000		RESERVE	PRIVATE	
00090000	00003000	RW GUARD	COMMET	PRIVATE	
00093000	0000D000	RW	COMMET	PRIVATE	
00040000	00079000		RECERVE	PRTVATE	<b>`</b>
•					
┌转存信息-					
地址:	00000000	大小: 000100	🔍/blog.csdn.nei	/weixin转存/	5859

这里只是举个例子,这个程序还没有进行脱壳调式。可以看到有些区域显示的是**NOACCESS**。如果加壳程序做了我们上面所说的处理的话,PE头也会出现这种情况。

这个时候我们可以打开OD,按下"Alt+M"或者点击蓝色小框框里面的M打开内存映像,然后在PE文件头那里右键,选择设置访问->所有访问。

ששנ				PP1V	00021004	RW	ĸw		
<u> 300</u>			数据块 于	Priv	00021004	RW	RW		
000	RebPE		PE 文/H VI	T	04 004 009	n	דו ייד.		
<u> 100</u>	RebPE	.text	代码	刷新			E		
<b>300</b>	RebPE	.rdata	数据	在CPU数	据窗口中春春		E		
<b>300</b>	RebPE	.data		ш			ΙE		
<b>300</b>	RebPE	.rsrc	资源	数据			E		
<u> 100</u>	RebPE	.pediy	SFX,辅	查找		Ctrl+B	ΙE		
<u> 100</u>							-t-		
<b>300</b>				在访问上谈	2置中断	F2	1		
<u> 100</u>									De
<b>300</b>	apphe lp		PE 文	设宜内仔训	5191断点(A)		E		
<u> 100</u>	apphe lp	.text	SFX, 件	设置内存属	写入断点(W)		ΙE		
<u> 1</u> 00	apphe lp	.data	数据	沿署话间			、 ·	<b>林正</b> 话问	
<u> 1</u> 00	apphe lp	.idata	输入表	双直初间			<u></u>		
<u> 100</u>	apphe lp	.rsrc	资源	复制到剪则	栃板		>	只读	
<u> 1</u> 00	apphe lp	.reloc						读取/写入	
<b>300</b>	Kerne 1Ba		PE 文	俳序按			<b>`</b>	++ /=	
<b>300</b>	Kerne 1Ba	.text	SFX, 件	界面选项			>	באאמ	
<b>300</b>	Kerne 1Ba	.data	数据	Imag	01001002	R	R	执行/读取	
<u> 1</u> 00	Kerne 1Ba	.idata	输入表	Imag	01001002	R	R	所有访问	
<u> 1</u> 00	Kerne 1Ba	.didat		Imag	01001002	R	Rus	111 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1	
000	Kerne 1Ba	.rsrc	资源	Imag	01001002	R/blog.c	RWE	iet/weixin_435758	
		_				_			

下面就试一下用LordPE进行脱壳,用到的加壳程序是书本《加密与解密》里面的例子: RebPE。

首先我们用OD打开目标程序:

地址	HEX 数据	反汇编	汪祥
00413000	60	PUSHAD	
00413001	E8 C2000000	CALL RebPE.004130C8	
00413006	2E:3001	XOR BYTE PTR CS:[ECX],AL	
00413009	0000	ADD BYTE PTR DS:[EAX],AL	
0041300B	0000	ADD BYTE PTR DS:[EAX],AL	
0041300D	0000	ADD BYTE PTR DS:[EAX],AL	
0041300F	0000	ADD BYTE PTR DS:[EAX],AL	
00413011	003E	ADD BYTE PTR DS:[ESI],BH	
00413013	3001	XOR BYTE PTR DS:[ECX],AL	
00413015	002E	ADD BYTE PTR DS:[ESI],CH	
00413017	3001	XOR BYTE PTR DS:[ECX],AL	
00413019	0000	ADD BYTE PTR DS:[EAX],AL	
0041301B	0000	ADD BYTE PTR DS:[EAX],AL	
0041301D	0000	ADD BYTE PTR DS:[EAX],AL	
0041301F	0000	ADD BYTE PTR DS:[EAX],AL	
00413021	0000	ADD BYTE PTR DS:[EAX],AL	
00413023	0000	ADD BYTE PTR DS:[EAX],AL	
00413025	0000	ADD BYTE PTR DS:[EAX],AL	
00413027	0000	ADD BYTE PTR DS:[EAX],AL	
00413029	0000	ADD BYTE PTR DS:[EAX],AL	
0041302B	0000	ADD BYTE PTR DS:[EAX],AL	
0041302D	0020	ADD BYTE PTR DS:[EAX],AH	https://blog.csdp.pet/weixin_43575859
0041302F	5F	POP EDI	https://biog.csun.net/weixin_400/0009

很明显是被加壳了。然后我们单步执行一下之后,在ESP寄存器中的内存地址上设置内存访问断点,然后运行 程序,跟踪到OEP的地方:

0040112F	90	NOP	
00401130	<b>r.</b> 55	PUSH EBP	
00401131	. 8BEC	MOU EBP, ESP	
00401133	. 6A FF	PUSH –Ø×1	
00401135	. 68 B8504000	PUSH RebPE.004050B8	
0040113A	. 68 FC1D4000	PUSH RebPE.00401DFC	SE 处理程序安装
0040113F	. 64:A1 0000000	MOV EAX,DWORD PTR FS:[0]	
00401145	. 50	PUSH EAX	
00401146	. 64:8925 00000	MOU DWORD PTR FS:[0],ESP	
0040114D	. 83EC 58	SUB ESP,0×58	
00401150	. 53	PUSH EBX	
00401151	. 56	PUSH ESI	
00401152	. 57	PUSH EDI	
00401153	. 8965 E8	MOU DWORD PTR SS:[EBP-0×18],ESP	
00401156	. FF15 28504000	CALL DWORD PTR DS:[0×405028]	kerne132.GetVersion
0040115C	. 33D2	XOR EDX,EDX	
0040115E	. 8AD4	MOU DL,AH	
00401160	. 8915 08854000	MOU DWORD PTR DS: [0×408508], EDX	https://blog.csdn.net/weixin_43575859
00401166	8868	MOUL FOY FOY	

这个时候进程在内存中已经脱壳完成了,现在我们就要用LordPE来Dump数据了。用管理员权限打开软件,然后右键目标进程,先选择修正镜像大小:



软件成功修正了镜像的大小。所以说这个程序也修改了MODULEENTRY32结构里面的值。

然后我们再选择完整转存,保存到磁盘文件。

最后我们点击运行一下:



居然出现了一个错误。这就很崩溃了。

但是办法总比困难多,一番查询就会知道,加壳程序会对程序的IAT做手脚,破坏原程序的输入表是加密外壳必须具备的功能。所以上面这个程序虽然我们把它的外壳去掉了,但是它的IAT已经被破坏了,所以无法调用系统 API。

一些外壳会改变原程序IAT里面的内容,加壳的时候,由于壳会自建一个输入表,并且让原来的PE文件的数据目 录里面输入表指针指向自建的输入表。这样PE文件装载器会对自建的输入表进行填写。而原来的输入表被外壳 变形后储存。程序运行的时候外壳将变形后的输入表进行扫描,然后重新填写所有函数的地址。

要注意:因为IAT填写完之后,输入表的其他部分就不重要了,程序依靠IAT里面的函数地址就能正常运行,并 且壳只是填了IAT里面的内容,对于输入表其他部分,可以说是已经破坏了,也就是说,加壳程序运行的时候, 原程序的输入表已经不在了,内存里面就只有一个IAT。

所以像我们上面那样Dump数据后,文件里面的没有输入表的,自然程序运行不了。所以我们要重建输入表。

还有一些外壳,填写到IAT里面的并不是真正的API地址。它填的可能是外壳的HOOK API地址,这样程序每次调用API就会跳转到外壳中运行,这样外壳又可以获得控制权。

这个时候我们就要用到Import REC(全称: Import REConstructor)这个软件了。

这个软件可以帮助我们重建程序的输入表,并且用起来很简单。

运行这个软件前我们要确保几个事情:

- 目标文件已经被像我们上面那样给Dump下来了。
- 目标文件正在运行
- 我们事先得知道目标程序的真正OEP或者IAT的偏移量和大小。

所以如果完成了上面的Dump的步骤之后,我们再将原来的没有Dump的程序用OD打开,然后跟踪到OEP处:

0040112F	90	NOP	
00401130	<b>1. 55</b>	PUSH EBP	
00401131	. 8BEC	MOU EBP, ESP	
00401133	. 6A FF	PUSH -0×1	
00401135	. 68 B8504000	PUSH RebPE.004050B8	
0040113A	. 68 FC1D4000	PUSH RebPE.00401DFC	SE 处理程序安装
0040113F	. 64:A1 000000	MOU EAX, DWORD PTR FS:[0]	

得到真正的OEP(箭头所指)。

然后我们打开软件,在下拉栏里面找到目标进程:

附加到一个活动进程 は、逆向脱売工具、importreconstructor 1.7e\importreconstructor 1.7e\importrec.美化版 exe (0000289 は、百度风盤下载、光盘\pediy book4光盘2\pediy book4\chap11\tools\lordpe\lpe-dlx\lordpe hh.exe (00 d.\百度风盤下载、光盘\pediy book4光盘2\pediy book4\chap16\16.4 單建输入表\16.4.4 importrec d.\百度风盤下载、光盘\pediy book4光盘2\pediy book4\chap16\16.4 單建输入表\16.4.4 importrec d.\百度风虚下载、光盘\pediy book4光盘2\pediy book4\chap16\16.4 單建输入表\16.4.4 importrec d.\百度风虚下载、光盘\pediy book4光盘2\pediy book4\chap16\16.4 單建输入表\16.4.4 importrec d.\百度风虚下载、光盘\pediy book4分法2\pykernel.exe (00003838) d.\窗行艺\iqiyi video\lstyle\7.4.108.1612\qyfragment.exe (0000366C) d.\gq\bin\tplatform.exe (00000694) d.\steanlibrary\steamapps\common\wallpaper\_engine\wallpaper32.exe (00003204) d.\—个文件来\wps\wps office\11.1.0.9584\office6\wps.exe (00003238) d.\—个文件来\wps\wps office\11.1.0.9584\office6\wps.exe (00003238) d.\—个文件来\wps\wps office\11.1.0.9584\office6\wps.exe (00003238) d.\—个文件来\wps\wps office\11.1.0.9584\office6\wps.exe (00003238)

然后在OEP框框那里填入正确的OEP的RVA,然后再点击IAT自动搜索,让软件自动搜索IAT的偏移量和大小:



如果OEP发挥作用,那么软件会弹出一个窗口:

	显示可能	
	自动服	<u>- 退出</u>
	清空导》	入表 清空日志
已发现一些信息!	× 0000	必需的 IAT 信息 D1130 IAT 白动機室
发现了可能是在原始 IAT 中的地址. 尝试 '获 (如果无效, 尝试 RVA: 00005000 大小:00001	取导入表'. 1000)	25000 大小 000000B8 
	<b>詩</b> 入	表信息(IID+ASCII+加载器)
_	0000	20000 大小 00000000
	備定	🗹 添加新的区段
: c:\windows\system32\ntdll. dll c:\windows\system32\kernel32. dll c:\windows\system32\kernelbase. dll c:\windows\system32\apphelp. dll c:\windows\system32\aser32. dll	https://blo <u>g.esdn.r</u>	载入树 保存树 (修正转储)

如果说我们没有正确的OEP,或者说软件没有找到IAT的偏移量,那么我们需要手动填入IAT的RVA和SIZE:

	必需的	IAT 信息
OEP	00001130	
RVA	00005000	大小 000000B8

完成上面步骤之后,我们点击获取导入表:

		IAT 信息
OEP	00001130	IAT 白动搜索
RVA	00005000	大小 00000088
		0三4.羌

日志框:

日志
模块已加载: c:\windows\system32\win32u.dll
積決已加载: c:\windows\system32\gdi32full.dll 積決已加载: c:\windows\system32\nsvcn win dll
權決已加載: c:/windows/system32/ucrtbase.dll 描述:Dint#: c:/windows/system32/ucrtbase.dll
原始 IAT RVA 发现于: 00005028 位于区段 RVA: 00005000 大小:00001000
TAL 英雄(MA). RVA:00005088 来自 模块:ntdll.dll 序号:029A 名称:RtlAllocateHeap RVA:00005090 来自 模块:ntdll.dll 序号:05C8 名称:RtlReAllocateHeap
20 (十进制:44) 色导入函数. 《研练规// 的 ( 小进制 : 144) // eixin_435758.

如果有读取不成功的,就会在下面这个框中显示有效:否。

找到的导入表函数
⊞-kernel32.dll FThunk:00005000 函数数:26(十进制:38)有效:是 册-user32.dll FThunk:0000509C 函数数:6(十进制:6)有效:是

这里我们的API都能够识别,如果有不能识别的,右键这个框的空白处,然后选择跟踪级别1,然后点击"显示无效的",如果还是有无效的,那么再右键空白处,选择跟踪级别2。

最后就到了修复程序的时候了,在下面这个地方的复选框中选中添加新节区:

这样软件会把新的输入表放在这个新的节区里面,这样会不可避免导致程序变大,我们也可以将输入表建在程 序空白处,这样我们就得自己填写RVA了。

最后点击修正转储,选择我们之前Dump下来的文件进行转储,软件会生成一个新的文件,OEP也会被自动修 正,这个文件就是最后修正了输入表的文件,点击运行:



完全没有问题。至此就脱壳完成了。

加壳程序: https://download.csdn.net/download/weixin\_43575859/12337598

LordPE和Import REC在吾爱破解的爱盘里面都能找到。