2017
绿盟科技
恶意样本分析手册

绿盟科技安全能力中心(SAC)

[文件封装篇]

(第二版)

关于绿盟科技

北京神州绿盟信息安全科技股份有限公司（简称绿盟科技）成立于 2000 年 4 月，总部位于北京。在国内外设有 30 多个分支机构，为政府、运营商、金融、能源、互联网以及教育、医疗等行业用户，提供具有核心竞争力的安全产品及解决方案，帮助客户实现业务的安全顺畅运行。

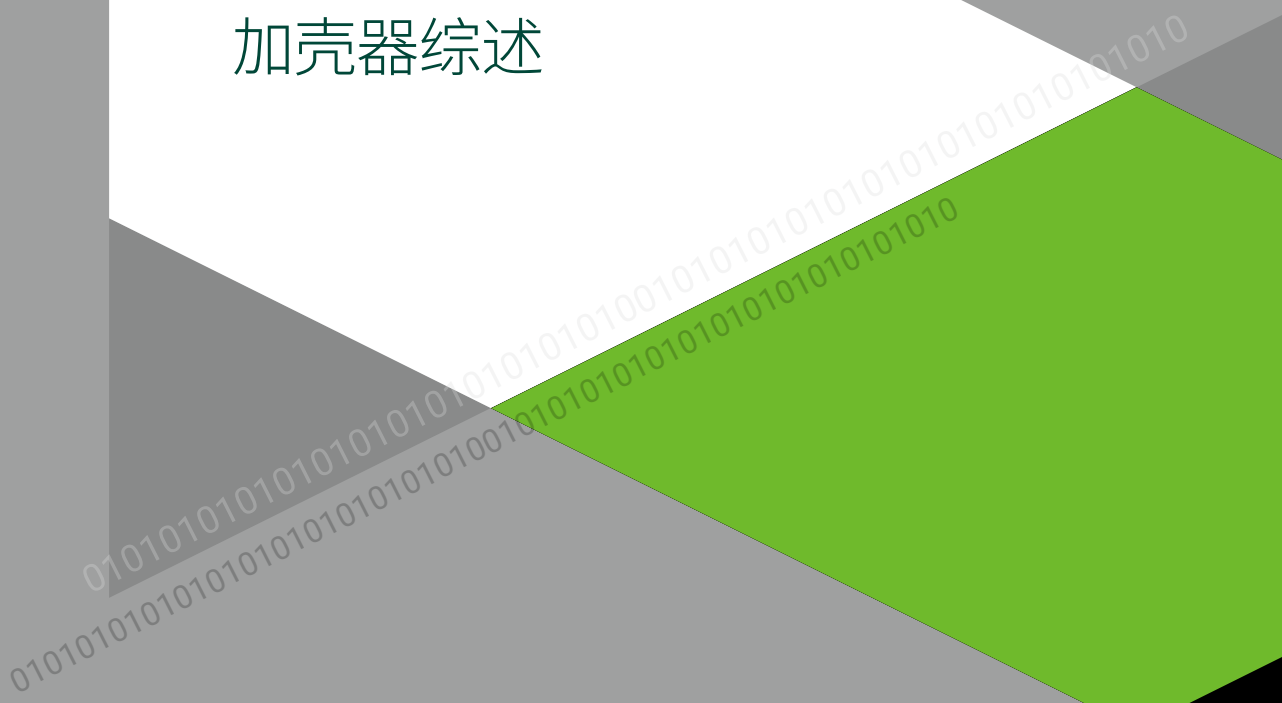
基于多年的安全攻防研究，绿盟科技在网络及终端安全、互联网基础安全、合规及安全管理等领域，为客户提供入侵检测 / 防护、抗拒绝服务攻击、远程安全评估以及 Web 安全防护等产品以及专业安全服务。

北京神州绿盟信息安全科技股份有限公司于 2014 年 1 月 29 日起在深圳证券交易所创业板上市交易。

股票简称：绿盟科技　股票代码：300369

目录

加壳器综述



加壳器属于一种封装工具，它可以对反病毒软件，复杂的恶意代码分析过程隐藏恶意代码的存在，另外，能缩小恶意代码可执行文件的大小，因此它们在恶意代码编写者中很受欢迎。大部分加壳器都是免费且易于使用的。基础静态分析技术对加壳后的程序毫无办法，要想进行静态分析，必须先将加壳的恶意代码脱壳，这就给恶意样本分析增加了很大的难度。

加壳可执行文件的两个主要目的是缩小程序的大小，阻碍对加壳程序的探测和分析。虽然加壳程序的种类繁多，但它们都遵循相似的模式：将一个可执行文件转换创建一个新的可执行文件，被转换的可执行文件在这个新的可执行文件中作为数据存储，另外新的可执行文件还包括一个供操作系统调用的脱壳存根 (stub)。

在讲解加壳后的文件结构前，先要了解一下加壳器的工作原理。

剖析加壳

所有的加壳器都是将一个可执行文件作为输入，输出一个新的可执行文件。被加壳的可执行文件经过压缩，加密或者其他转换，目的是使他们难以被识别，难以被逆向分析。

多数加壳器用压缩算法压缩可执行文件，通过加密原始可执行文件并且实施一些反逆向技术实现，如对抗反汇编、反调试和反虚拟机等。加壳器既可以打包整个可执行文件，包括所有的数据与资源节，也可以仅打包代码节和数据节。

要保持原程序的功能，加壳程序需要存储程序中的导入函数表信息。这些信息可以用任何格式存储。

脱壳存根

未加壳的可执行程序由操作系统加载，被加壳程序中的脱壳存根也是由操作系统加载，然后它负责加载原始程序。可执行程序的入口点指向脱壳存根，而不是原始代码。原始程序通常存储在加壳程序的一个或多个附加的节中。

我们可以查看脱壳存根，理解脱壳存根的不同部分是脱壳的基本规则。因为脱壳存根不负责执行程序的主体功能，所以它通常很小。另外，它的功能也很简单，只是脱壳原始程序。一般脱壳存根会执行以下三步操作：

1. 将原始程序脱壳到内存中
2. 解析原始可执行文件的所有导入函数
3. 将可执行程序转移到原始的程序入口点（OEP）

加载可执行程序

当加载一个可执行文件时，加载器会首先读取硬盘上可执行文件的 PE 头部信息，然后根据 PE 头部信息为可执行文件的各个节分配内存。然后，加载器将这些节复制到分配的内存空间中。

加壳后的可执行文件会组成 PE 头部，让加载器为它的节分配空间，它的节要么来自原始程序，要么是脱壳存根创建的节。脱壳存根会脱壳每个节的代码，并将它们复制到分配的内存空间中。具体使用哪种脱壳方法随加壳者的目的而定，一般情况下它们会存放在存根中。

解析导入函数表

未加壳的 PE 文件中有一个节段告诉加载器需要导入哪些函数，同时还有一个节存储了需要导入的函数名字与地址。Windows 加载器读取导入信息，确定需要导入哪些函数，然后填入导入函数的地址。

Windows 加载器不能读取被加壳可执行程序的导入函数表。对于加壳过的可执行文件，脱壳存根负责解析导入函数表，具体方法取决于使用的壳。

脱壳存根最常使用的方法是仅导入 LoadLibrary 和 GetProcAddress 两个函数。脱壳存根在脱壳出原始可执行文件之后，才能读取可执行文件的导入函数信息。为了将每个 DLL 加载到内存，脱壳存根将调用 LoadLibrary 函数导入每个函数库。然后使用 GetProcAddress 获取每个函数的内存地址。

另外一种方法是保持原始导入函数表的完整，让 Windows 加载器能够加载所有 DLL 和导入函数。这是最简单的方法，然而静态分析加壳程序就可以发现所有原始导入表，所以这种方法缺乏隐蔽性，此外，导入函数以明文存储在可执行文件中，因此这种方法的压缩性也不理想。

第三种方法是为原始导入表中的每个 DLL 保留一个函数。分析时，这种方法只能查看每个导入库中的一个函数，因此它比第二种方法的隐蔽性更高，但分析时仍然能够看到原始可执行文件所有的导入库。因为导入库不需要被脱壳存根加载，所以这种加壳方法比第一种方法简单，但脱壳存根仍需要解析大部分导入函数。

最后一种方法是不导入任何函数。加载程序在不用函数的前提下，自己从库中查找所有需要的函数，或者加壳程序首先找到 LoadLibrary 函数和 GetProcAddress 函数，然后用它们定位其他的库。这种方法的好处是加壳程序不导入任何函数，因此具有很高的隐蔽性，然而，为了使用这种方法，脱壳存根必须很复杂。

尾部跳转

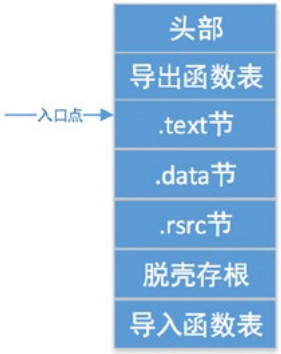
一旦脱壳存根完成脱壳，它就必须转到 OEP 运行。转到 OEP 的指令通常被叫做尾部跳转指令。jump 指令是最简单且最流行的转移运行指令。它非常普通，所以多数恶意的加壳程序试图使用 ret 或者 call 指令来隐藏这种行为。有时，恶意代码会使用操作系统转移控制的函数来掩盖尾部跳转，例如使用函数 NtContinue 或者 ZwContinue。

图示脱壳过程

图一：加壳前的可执行文件

图二：加壳后的可执行代码

图三：脱壳后加载到内存的程序



图四：完全脱壳后的程序

图一：原可执行文件。它的头部和各节都是可见的，而且代码开始点被设置为指向 OEP。

图二：存放在硬盘上的一个加壳后的可执行文件。它的可见部分有新的头部，脱壳存根及加过壳的原始代码。

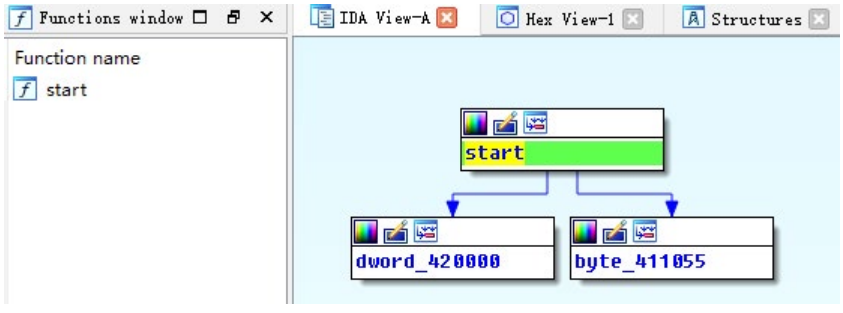
图三：载入内存中的一个加壳的可执行文件。此时脱壳存根已经脱出了原始代码，原来可执行文件的 .text 和 .data 节都可见。但可执行文件的入口点仍然指向脱壳存根，这种情况下，导入函数表一般无效。

图四：一个完全脱壳的可执行文件。导入表已经被重构，入口点也被设置为指向 OEP。

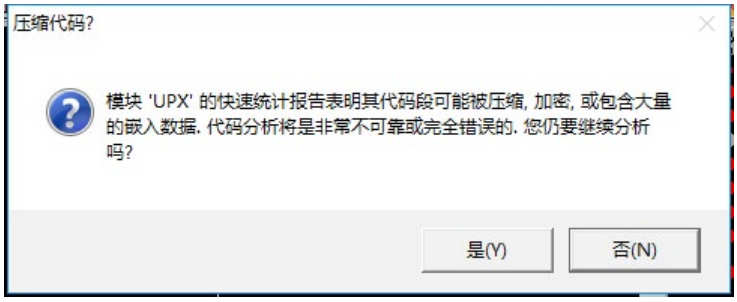
加壳程序的标识

下面几条总结了常见的恶意代码是否加壳的特征：

- 程序中导入函数很少，导入函数仅有 LoadLibrary 和 GetProcAddress 时，应该引起特别注意。
- 当使用 IDA Pro 打开程序时，通过自动分析，只有少量代码被识别。



- 当时用 OllyDbg 打开程序时，会有程序可能被加壳的警告。



- 程序的节名中包含某款加壳器的标识。

Name	Virtual Size	Virtual Addr...	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
UPX0	0001F000	00001000	00000000	00000400	00000000	00000000	0000	0000	E0000080
UPX1	00003000	00020000	00002C00	00000400	00000000	00000000	0000	0000	E0000040
.rsrc	00001000	00023000	00000400	00003000	00000000	00000000	0000	0000	C0000040

- 程序拥有不正常的节大小，例如 .text 节的原始数据大小为 0，但虚拟大小非零。

加壳后的文件特征

压缩壳

UPX

恶意代码最常使用的加壳器是 Ultimate Packer for eXecutables（UPX）。UPX 的特点是开源、免费且易于使用，同时还支持多种平台。UPX 用于压缩可执行文件，它为性能而不是安全所设计。UPX 之所以流行，是因为它有很高的压缩速度，较小的空间占用，而且压缩例程需要的内存也很少。

UPX 的工作原理其实是这样的：首先将程序压缩。所谓的压缩包括两方面，一方面在程序的开头或者其他合适的地方插入一段代码，另一方面是将程序的其他地方做压缩。压缩也可以叫做加密，因为压缩后的程序比较难看懂，主要是和原来的代码有很大的不同。最大的表现也就是他的主要作用就是程序本身变小了。变小之后的程序在传输方面有很大的优势。其次就是在程序执行时，实时的对程序解压缩。解压缩功能是在第一步时插入的代码完成的功能。联起来就是：UPX 可以完成代码的压缩和实时解压执行。且不会影响程序的执行效率。

UPX 和普通的压缩，解压不同点就算在于 UPX 是实时解压缩的。实时解压的原理可以使用一下图形表示：

1==>2==>3==>4==>5==>6

假设 1 是 UPX 插入的代码，2，3，4 是压缩后的代码。5，6 是随便的什么东西。

程序从 1 开始执行。而 1 的功能是将 2，3，4 解压缩为 7，8，9。7，8，9 就是 2，3，4 在压缩之前的形式。

1==>7==>8==>9==>5==>6

连起来就是：

最初代码的形式就应该是：7==>8==>9==>5==>6

用 UPX 压缩之后形式为：1==>2==>3==>4==>5==>6

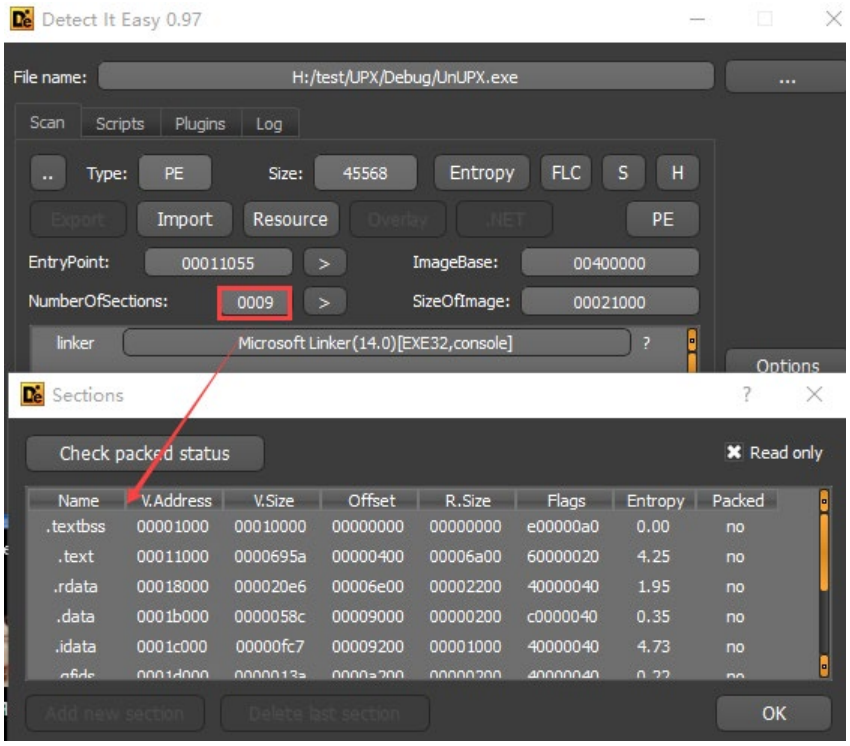
执行时的形式变为：1==>7==>8==>9==>5==>6

UPX 压缩可以大大的减少可执行文件的大小，如下图所示

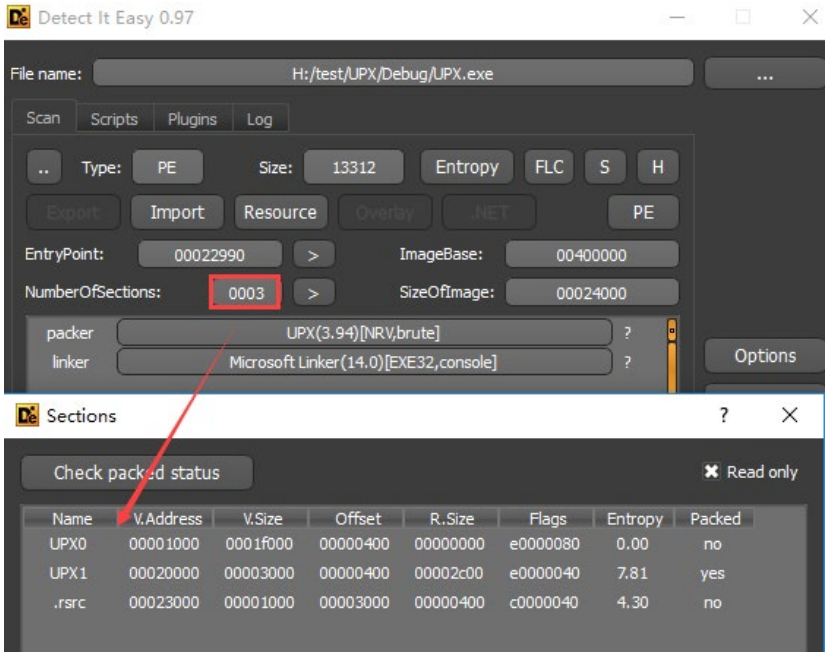
压缩前的大小

压缩后的大小

UPX 压缩同时会改变文件中的节信息，如下如所示，在压缩前，可执行文件有多个节，但是在压缩后，原始的节只剩下 .rsrc，不过添加了 UPX 增加的两个节。



压缩前的节信息



压缩后的节信息

ASPack

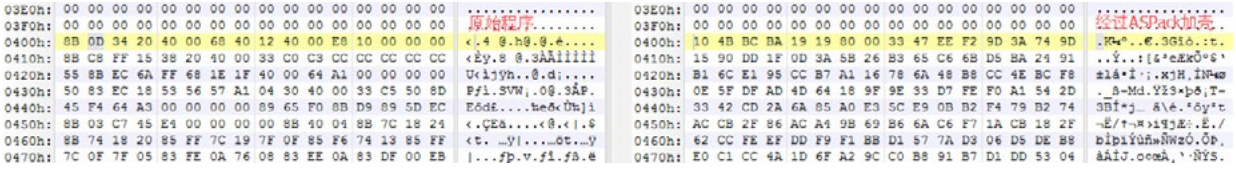
ASPack 的目的是为了安全，它采用一些技术让脱壳变得十分困难。ASPack 使用了自我修改代码，让设置端点和分析它变得困难。

在 ASPack 加壳的程序上设置端点，程序会立即终止，但是我们可以在栈地址上设置硬件断点完成对 ASPack 加壳程序的手动脱壳。另外，由于 ASPack 十分流行，因此很多自动脱壳程序都能对其进行脱壳。

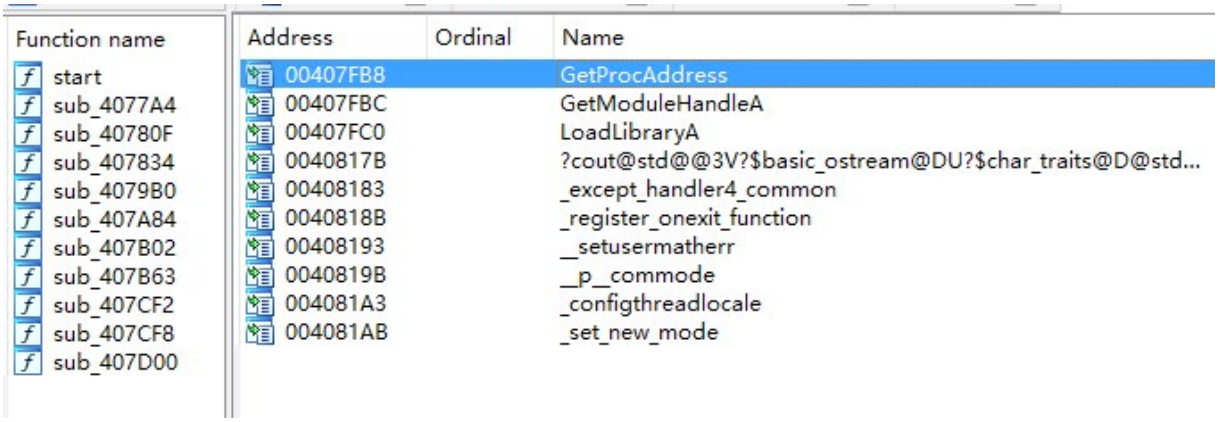
经过 ASPack 加壳的可执行文件会多出一个节 .aspack，而原始的文件中并没有这个节。

Name	Virtual Size	Virtual Addr...	Raw Size	Raw Address	Reloc Address
000002E0	000002E8	000002EC	000002F0	000002F4	000002F8
Byte[8]	Dword	Dword	Dword	Dword	Dword
.rdata	00001000	00002000	00000600	00000E00	00000000
.data	00001000	00003000	00000200	00001400	00000000
.gfids	00001000	00004000	00000200	00001600	00000000
.rsrc	00001000	00005000	00000200	00001800	00000000
.reloc	00001000	00006000	00000200	00001A00	00000000
.aspack	00002000	00007000	00001400	00001C00	00000000
.adata	00001000	00009000	00000000	00003000	00000000

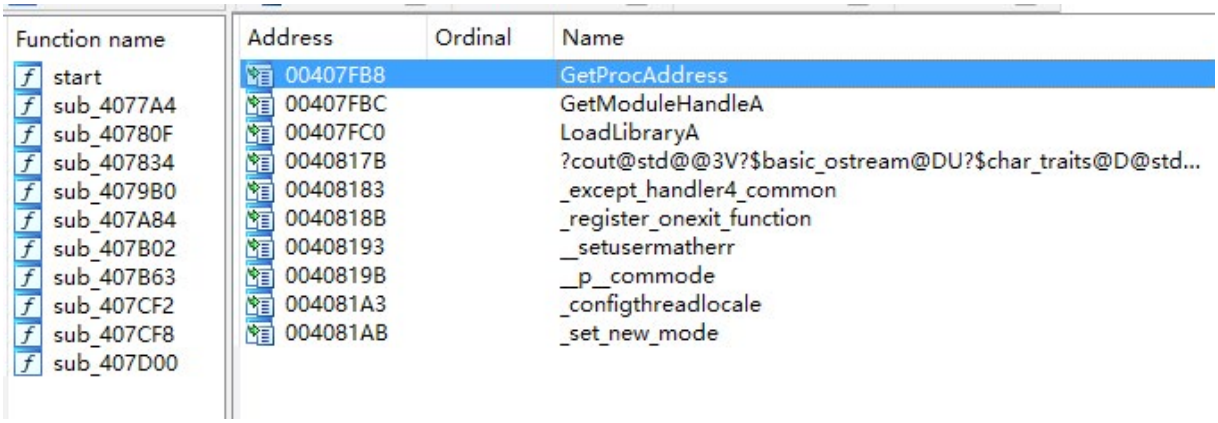
ASPack 会对程序的代码进行压缩



使用 IDA 加载可以发现，经过 ASPack 加壳的文件显示的函数和导入表中的函数比原始程序明显少了很多。



图：经过 ASPack 加壳的可执行文件

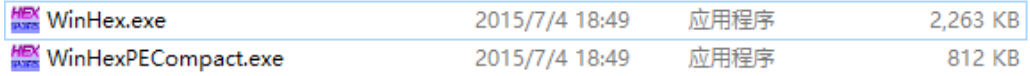


图：原始的可执行文件，部分截图

PECompact

PECompact 是一个能压缩可执行文件的工具，通过压缩代码、数据、相关资源使压缩能达到 100%，由于在运行时不需要恢复磁盘上压缩后的数据，所以与没有压缩的程序在运行时没有明显的速度差异，在某种程度上还有所改善。它同样也是一款能压缩可执行文件的工具（支持 EXE、DLL、SCR、OCX 等文件）。相比同类软件，PECompact 提供了多种压缩项目的选择，用户可以根据需要确定哪些内部资源需要压缩处理。同时，该软件还提供了加解密的插件接口功能。

经过 PECompact 压缩的程序在大小上有明显的变化



通过查看节信息也可以看出差异

Name	Virtual Size	Virtual Addr...	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word
CODE	001AE6CC	00001000	001AE800	00000400	00000000	00000000	0000
DATA	000094E0	001B0000	00009600	001AEC00	00000000	00000000	0000
BSS	0002D3ED	001BA000	00000000	001B8200	00000000	00000000	0000
.idata	00002E6A	001E8000	00003000	001B8200	00000000	00000000	0000
.tls	00000031	001EB000	00000000	001BB200	00000000	00000000	0000
.rdata	00000018	001EC000	00000200	001BB200	00000000	00000000	0000
.reloc	00016064	001ED000	00000000	00000000	00000000	00000000	0000
.rsrc	0007A800	00204000	0007A800	001BB400	00000000	00000000	0000

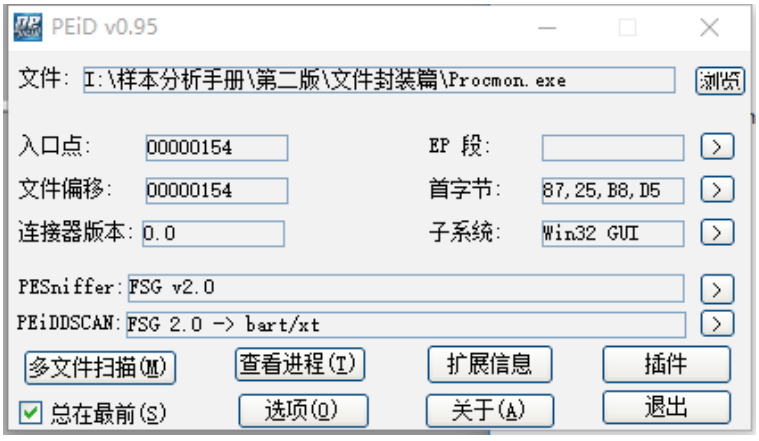
图：原始节信息

Name	Virtual Size	Virtual Addr...	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word
CODE	0027E000	00001000	000C4200	00000400	32434550	00004F7C	0000
.rsrc	00007000	0027F000	00006800	000C4600	00000000	00000000	0000
.reloc	00000200	00286000	00000200	000CAE00	00000000	00000000	0000

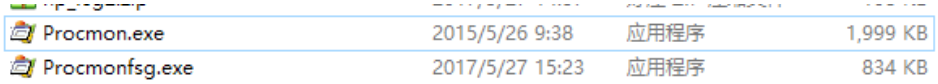
图：经过 PECompact 压缩的节信息

FSG

FSG 同上面三个一样，也是一种压缩壳，由于 fsg 壳比较常见，可以使用软件将其识别出来。



经过 fsg 压缩后，软件的大小明显变小了



节的数量也变少了

Name	Virtual Size	Virtual Addr...	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word
.text	0008AAFD	00001000	0008AC00	00000400	00000000	00000000	0000
.rdata	00026B94	0008C000	00026C00	0008B000	00000000	00000000	0000
.data	000096E4	000B3000	00001A00	000B1C00	00000000	00000000	0000
.rsrc	00135A34	000BD000	00135C00	000B3600	00000000	00000000	0000
.reloc	00008D90	001F3000	00008E00	001E9200	00000000	00000000	0000

图：原始的节信息

Name	Virtual Size	Virtual Addr...	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word
	001FC000	00001000	00000000	00000000	00000000	00000000	0000
	000D1000	001FD000	000D05F5	00000200	00000000	00000000	0000

图：经过 fsg 压缩的节信息

加密壳

Themida

Themida 是一个非常复杂的壳，拥有多种功能。其中大部分功能是反调试与反逆向分析，这保证了他是一个非常安全的壳，难以脱壳和分析。

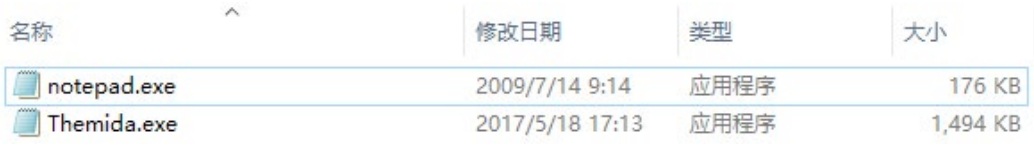
Themida 包含组织 VMware、调试器以及 Procmon 分析的功能。除此之外，Themida 还有一个内核模块，这让分析变得尤其困难。运行在内核中的代码显示很少，并且分析程序通常运行在用户空间中，因此分析会受到很多限制。

由于 Themida 拥有这么多功能，因此使用它加壳文件变得很笨重。除此之外，与大多数壳不同，Themida 代码会在原始程序运行后一直运行。

有一些专门为脱壳 Themida 文件而设计的自动脱壳工具，它们的成功与否，取决于程序加壳时使用的 Themida 版本和设置。

如果不能自动脱壳，则另一种较好的策略是使用 ProcDump 工具从内存中转储不在进行调试的进程。ProcDump 是微软提供的一个工具，用来转储一个 Windows 进程的内容。设计他的目的是与调试器一起工作，但是它本身不是调试器。ProcDump 的最大优点是在不停止进程或者调试进程的情况下，转储进程中的内存。这对转储那些使用了反调试机制的壳来说，非常有价值。甚至当不能调试一个可执行文件时，仍可以使用 ProcDump 工具转储正在运行可执行文件的脱壳内容。这个过程并不能完全恢复可执行文件，但那时他能让我们在代码上使用 Strings 工具并做一些分析。

如下图所示，可以看出文件的大小明显增加。



名称	修改日期	类型	大小
notepad.exe	2009/7/14 9:14	应用程序	176 KB
Themida.exe	2017/5/18 17:13	应用程序	1,494 KB

通过查看节的信息，发现此工具会对节进行修改，并且添加了节，从而造成文件大小增加。

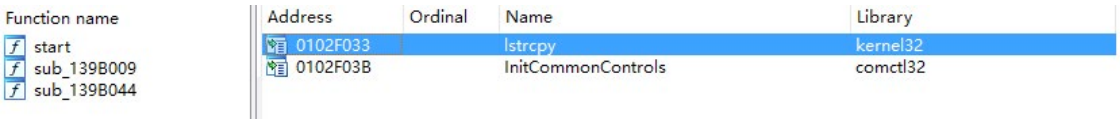
Name	Virtual Size	Virtual Addr...	Raw Size	Raw Address	Reloc Address
Byte[8]	Dword	Dword	Dword	Dword	Dword
.text	0000A68C	00001000	0000A800	00000400	00000000
.data	00002164	0000C000	00001000	0000AC00	00000000
.rsrc	0001F160	0000F000	0001F200	0000BC00	00000000
.reloc	00000E34	0002F000	00001000	0002AE00	00000000

notepad 的节信息

Name	Virtual Size	Virtual Addr...	Raw Size	Raw Address	Reloc Address
Byte[8]	Dword	Dword	Dword	Dword	Dword
	0000E000	00001000	00004C00	00001000	00000000
.rsrc	0001F160	0000F000	0001B400	00005C00	00000000
.idata	00001000	0002F000	00000200	00021000	00000000
	00217000	00030000	00000200	00021200	00000000
ahearhdy	00154000	00247000	00154000	00021400	00000000
thbropnc	00001000	0039B000	00000200	00175400	00000000

经过 Themida 加密的节信息

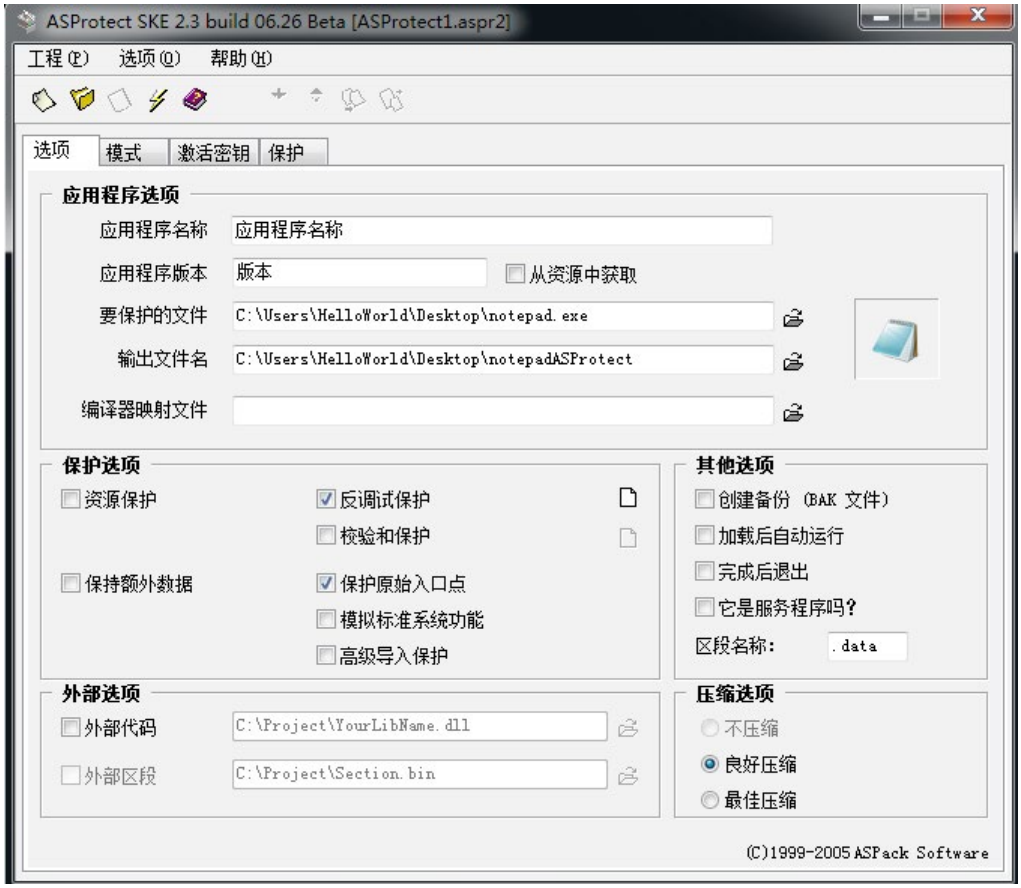
使用 IDA 加载加壳后的文件，只能看到很少解析出来的函数和导入函数



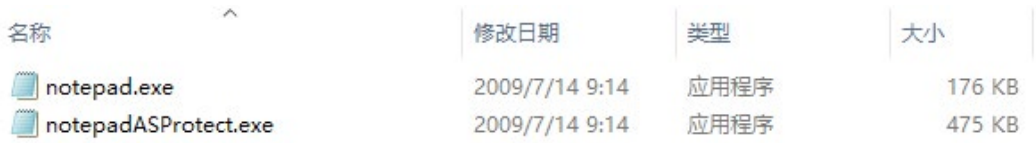
ASProtect

ASProtect 是一款非常强大的 Windows 32 位保护工具，它拥有压缩、加密、反跟踪代码、反 - 反汇编代码、CRC 校验和花指令等保护措施。它使用 Blowfish、Twofish、TEA 等强劲的加密算法，还用 RSA1024 作为注册密钥生成器。还通过 API 钩子（API hooks，包括 Import hooks（GPA hook）和 Export hooks）与加壳的程序进行通信。甚至用到了多态变形引擎（Polymorphic Engine），反 Apihook 代码（Anti-Apihook Code）和 BPE32 的多态变形引擎（BPE32 的 Polymorphic Engine）。并且 ASProtect 为软件开发人员提供 SDK，实现加密程序内外结合。

此款加壳工具的界面如下所示：



从图中可以看出来，通过这款工具可以为可执行文件添加不同的功能，同时，也将增加可执行文件的大小。



通过对比查看节的信息也可以看到，加壳后节的大小和数量也发生了变化。

Name	Virtual Size	Virtual Addr...	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...
000001D8	000001E0	000001E4	000001E8	000001EC	000001F0	000001F4	000001F8
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word
	0000B000	00001000	00005600	00000400	00000000	00000000	0000
	00003000	0000C000	00000200	00005A00	00000000	00000000	0000
.rsrc	00020000	0000F000	0001F200	00005C00	00000000	00000000	0000
	00001000	0002F000	00000E00	00024E00	00000000	00000000	0000
.data	00051000	00030000	00051000	00025C00	00000000	00000000	0000
.adata	00001000	00081000	00000000	00076C00	00000000	00000000	0000

加壳后的节信息

Name	Virtual Size	Virtual Addr...	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word
.text	0000A68C	00001000	0000A800	00000400	00000000	00000000	0000
.data	00002164	0000C000	00001000	0000AC00	00000000	00000000	0000
.rsrc	0001F160	0000F000	0001F200	0000BC00	00000000	00000000	0000
.reloc	00000E34	0002F000	00001000	0002AE00	00000000	00000000	0000

加壳前的节信息

VMProtect

VMProtect 是一款纯虚拟机保护软件。它是当前最强的虚拟机保护软件，经 VMProtect 处理过的代码，至今还没有人公开宣称能还原。虽然保护强度高，但是会影响程序速度，因此在一些对速度要求很高的场合就不适用了。

这款工具拥有很多选项，可以按照字节的目的进行修改。

▼ 文件	
内存保护	是
导入保护信息	是
资源保护	是
压缩输出的文件	是
输出文件	WinHex.vmp.exe
▼ 检测	
调试器	否
虚拟化工具	否
▼ 附加	
VM 分段	.vmp
使用 Taggant 系统	否
移除调试信息	是
移除重定位信息 (只对 EXE 文件有效)	否
水印	
▼ 消息	
检测到调试器	A debugger has been found running in your system. Please, unload it from memory and restart your program.
检测到虚拟机	Sorry, this application cannot run under a Virtual Machine.
文件受损	File corrupted! This program has been manipulated and maybe it's infected by a Virus or cracked. This file won't work anymore.
必需的序列号	This code requires valid serial number to run. Program will be terminated.
▼ 授权参数	
文件名	WinHex.exe.vmp
密钥对算法	无
激活服务器	

此次的重点不在于加壳，所以都选用默认的值，直接进行编译。然后 VMProtect 会生成一个 .vmp.exe 的文件。使用 ResourceHacker 来查看原始文件和经过加壳的文件，对比图如下所示：

通过查看资源文件的差异，可以发现很多资源文件都被 VMProtect 隐藏了起来。接下来再来看看节信息：

Name	Virtual Size	Virtual Addr...	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word
CODE	001AE6CC	00001000	001AE800	00000400	00000000	00000000	0000
DATA	000094E0	001B0000	00009600	001AEC00	00000000	00000000	0000
BSS	0002D3ED	001BA000	00000000	001B8200	00000000	00000000	0000
.idata	00002E6A	001E8000	00003000	001B8200	00000000	00000000	0000
.tls	00000031	001EB000	00000000	001BB200	00000000	00000000	0000
.rdata	00000018	001EC000	00000200	001BB200	00000000	00000000	0000
.reloc	00016064	001ED000	00000000	00000000	00000000	00000000	0000
.rsrc	0007A800	00204000	0007A800	001BB400	00000000	00000000	0000

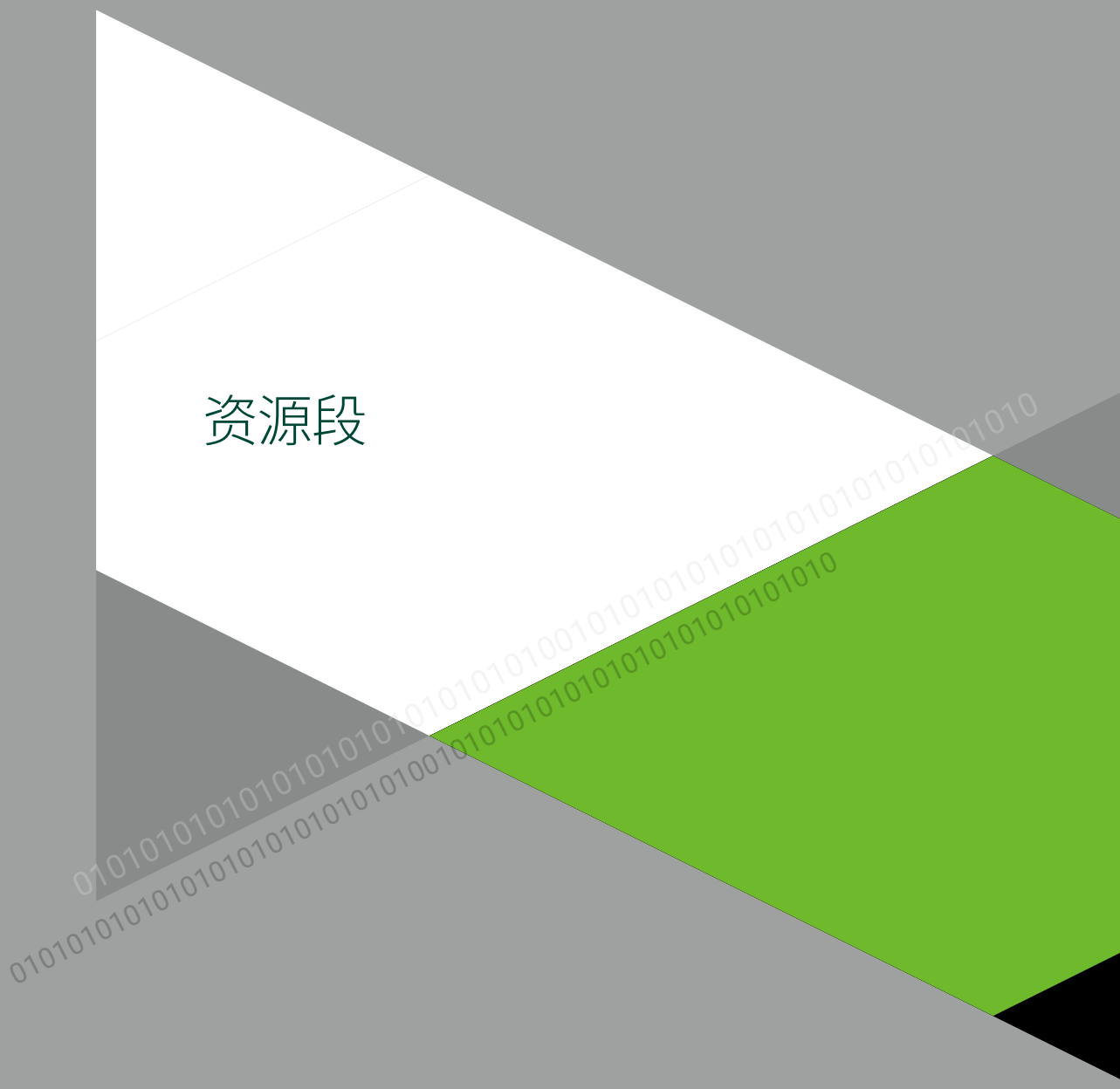
图：加壳前的节信息

Name	Virtual Size	Virtual Addr...	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word
CODE	001AE6CC	00001000	00000000	00000000	00000000	00000000	0000
DATA	000094E0	001B0000	00000000	00000000	00000000	00000000	0000
BSS	0002D3ED	001BA000	00000000	00000000	00000000	00000000	0000
.idata	00002E6A	001E8000	00000000	00000000	00000000	00000000	0000
.tls	00000031	001EB000	00000000	00000000	00000000	00000000	0000
.rdata	00000018	001EC000	00000000	00000000	00000000	00000000	0000
.reloc	00016064	001ED000	00000000	00000000	00000000	00000000	0000
.vmp0	00194EC4	00204000	00000000	00000000	00000000	00000000	0000
.vmp1	001DB430	00399000	001DB600	00000400	00000000	00000000	0000
.reloc	000002E0	00575000	00000400	001DBA00	00000000	00000000	0000
.rsrc	0000B983	00576000	0000BA00	001DBE00	00000000	00000000	0000

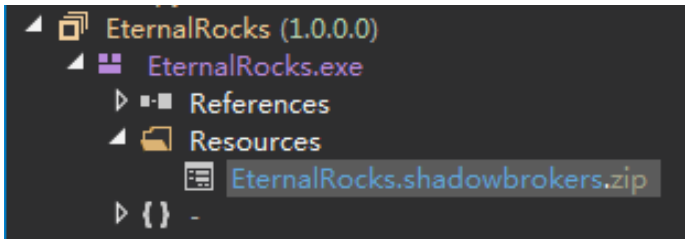
图：加壳后的节信息

通过对比节信息可以发现，经过 VMProtect 加壳后的应用程序多了两个节。

资源段

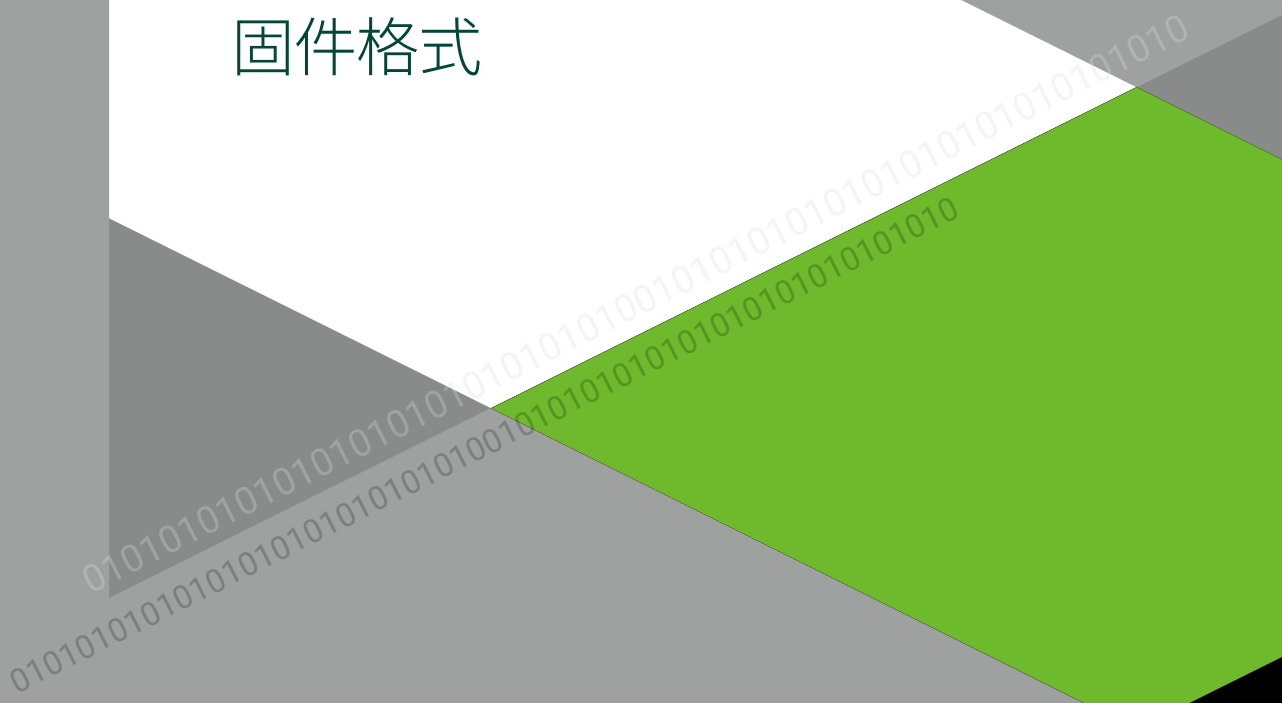


有些恶意样本会将它所需要的工具之类的文件隐藏在资源段中，然后在运行的时候将其释放出来。一种是申请内存，将所需要的工具释放到内存中，也就是所说的无文件运行，通过这种方法可以躲避杀毒软件的静态检测，比较安全。一种是直接将所需的工具释放到磁盘中，这种方法就没有上一种方法安全，由于将文件释放到了磁盘中，很容易被杀毒软件查杀。



如上图所示，样本名为 EternalRocks.exe，在它的资源段中包含了 shadowbrokers.zip，这个压缩包中包含了样本将要使用的漏洞利用程序。样本会在运行时，将这个压缩包释放到磁盘中并进行解压，然后执行文件夹中的漏洞利用程序进行攻击。

固件格式



对于路由器固件的格式和前面所说的 PE 文件格式大不相同，下面通过几个小工具对固件的解析结果来对其格式进行说明。

首先使用 file 命令来查看一下它的文件类型。

```
root@kali:~# file DIR890LA1_FW108b03.bin
DIR890LA1_FW108b03.bin: data
root@kali:~#
```

可以看出来它的文件类型为 data，接下来使用 hexdump 工具来查看文件中的每一个字节，使用 -C 命令可以设置 hexdump 输出为 hex+ASCII 的方式，便于阅读。通过这个工具输出的 txt 的文件很大，通过看输出文件的前一部分可以知道此固件是属于 dlink 的。

```
root@kali:~# hexdump -C DIR890LA1_FW108b03.bin >hex.txt
root@kali:~# cat hex.txt|more
00000000  5e a3 a4 17 00 00 00 28  00 00 00 00 73 69 67 6e  |^......(....sign|
00000010  61 74 75 72 65 3d 77 72  67 61 63 33 36 5f 64 6c  |ature=wrgac36_dl|
00000020  69 6e 6b 2e 32 30 31 33  67 75 69 5f 64 69 72 38  |ink.2013gui_dir8|
00000030  39 30 00 00 5e a3 a4 17  00 00 00 24 01 0e 30 20  |90..^......$..0 |
00000040  df bb 90 21 c4 2d 74 90  ab 32 45 1f 84 fb 3f f9  |...!.-t..2E...?.|
00000050  64 65 76 3d 2f 64 65 76  2f 6d 74 64 62 6c 6f 63  |dev=/dev/mtdbloc|
00000060  6b 2f 37 00 74 79 70 65  3d 66 69 72 6d 77 61 72  |k/7.type=firmwar|
00000070  65 00 00 00 5d 00 00 00  02 a0 d9 4a 00 00 00 00  |e...]......J....|
00000080  00 00 69 bc 00 2e 30 33  d8 56 a4 b3 90 93 ce 28  |..i...03.V.....(|
```

作为初始的信息收集，strings 可以说是最常用的工具之一，它可以显示文件中所有可打印的数据。从获得的 string 信息中可以看到它的类型为 firmware。

```
root@kali:~# strings DIR890LA1_FW108b03.bin|more
signature=wrgac36_dlink.2013gui_dir890
dev=/dev/mtdblock/7
type=firmware
,@NK
1P/Q
WKe6
```

binwalk 会分析进制文件中可能的固件头或者文件系统，然后输出识别出的每个部分以及对应的偏移量。从下图的输出结果中可知该固件采用的是 Squashfs 文件系统。

```
root@kali:~# binwalk DIR890LA1_FW108b03.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             DLOB firmware header, boot partition: "dev=/dev/mtdblock/7"
116           0x74            LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes, uncomp
ressed size: 4905376 bytes
1835124       0x1C0074        PackImg section delimiter tag, little endian size: 3207680 bytes; big endian si
ze: 15872000 bytes
1835156       0x1C0094        Squashfs filesystem, little endian, version 4.0, compression:lzma (non-standard
 type definition), size: 15869852 bytes,  2642 inodes, blocksize: 131072 bytes, created: Tue Jul  7 18:17:21
2015
```

封装工具

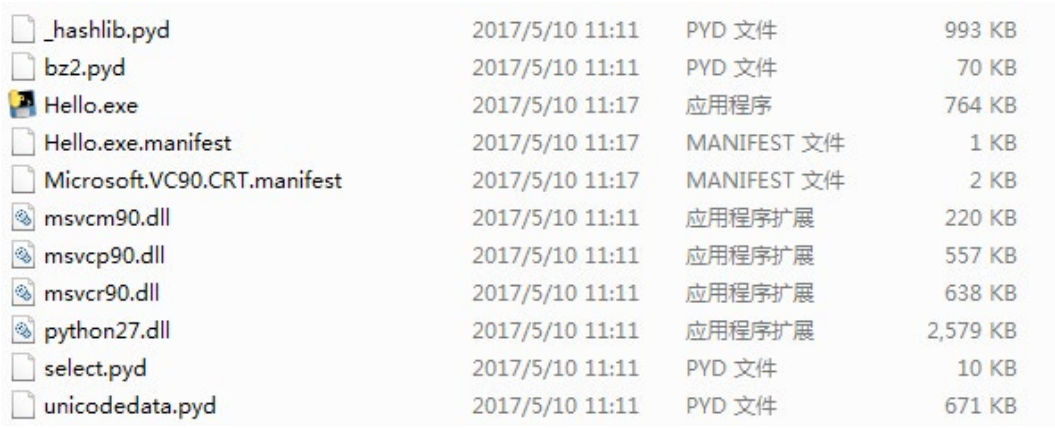
有些软件是使用 python，ruby 等语言写的，如果没有特定的环境的话就无法运行，为了便于部署，就出现了一些工具，将脚本源码编译成可执行文件，编译后的文件就可以脱离特定的环境运行了，在这里首先介绍三个打包程序，知道它的过程后，更有利于理解打包后的文件结构。

python 封装工具 -Pyinstaller

直接使用 Python 开发的软件时有许多不方便的地方，如需要安装特定的 Python 环境，需要安装依赖库。为了便于部署，需要将 Python 源代码编译成可执行文件，编译后的可执行文件就能脱离 python 环境运行了。可以通过 Pyinstaller 来对文件进行封装。

最简单的使用方式是运行 pyinstaller hello.py 来生成可执行文件，其中 hello.py 是需要编译成可执行文件的源代码，通过这种方式生成的可执行文件默认位于当前文件夹的 dist 目录下，dist 目录位于 pyinstaller 所在的 myscript 目录中。

通过这种方式同时还会生成 build 文件夹，这个文件夹存放的是编译过程中的生成的临时文件，可以删除。在 dist 文件夹下，除了有 exe 文件外，还有若干个其他文件，这些文件都是运行时必须的。



这给我们发布软件带来了不便，如果希望编译出的 exe 文件不依赖其他文件，可以添加 -F 选项：

pyinstaller –F hello.py，这时候再看，文件夹里面只有一个 exe 文件。

下面对其常用的参数进行说明：

参数	说明
-w --windowed --noconsole	使用 windows 子系统执行，当程序启动的时候，不会打开命令行。
-d --debug	产生 debug 版本的可执行文件
-c --console --nowindowed	使用控制台子系统执行 (默认)(只对 Windows 有效)
--distpach DIR	设置产生的应用程序的存放位置，默认是 ./dist
--workpath WORKPATH	设置临时文件的存放目录，默认为 ./build
--specpath DIR	设置 spec 文件的存放目录，默认在当前文件夹下
-n NAME --name NAME	设置生成的应用程序和 spec 文件的名称
-p DIR --paths DIR	设置导入路径 (和使用 PYTHONPATH 效果相似). 可以用路径分割符 (Windows 使用分号 ,Linux 使用冒号) 分割 , 指定多个目录 . 也可以使用多个 -p 参数来设置多个导入路径
--key KEY	使用指定的 key 加密 python 代码
-i FILE.ico or FILE.exe or FILE.icns --icon FILE.ICO or FILE.exe or FILE.icns	指定生成的应用程序的图标
--uac-admin	指定生成的应用程序需要管理员权限运行

Ruby 封装工具 -exerb

目前将 Ruby 代码打包成 exe 可执行文件主要有 3 种方式：

1. rubyscript2exe，年久失修，打包出来的文件太大，不对源文件进行加密，运行时将源码释放到一个临时目录后执行；
2. exerb，已经支持 Ruby1.8.7 和 1.9，可以设置程序的版本信息，图标等，打包后的可执行文件可以用 UPX 压缩，功能很强大，执行时不释放源文件出来，对程序加密较好，但是对 waitr 这种需要调用 DLL 的 gem 支持不是很好，无法进行打包处理；
3. ocra，原理和 rubyscript2exe 差不多，可定制性不强，但是对 Ruby1.8.7 和 1.9 以及 waitr 都提供很好的支持，而且打包的时候会对文件进行压缩，打包后程序的大小可以接受，程序图标和版本信息暂时不能定制，但是默认图标比 rubyscript2exe 要好看，如果对源码保护要求不是很强，用 exerb 又无法成功打包的时候，可以采用这个。

这里以比较强大的 exerb 为例进行说明。先执行 mkexy hello.rb，会自动生成一个 hello.exy 的配置文件，打开此配置文件，加入下面的代码：

```
resource:
 icon:
  - width : 16
    height: 16
    color : 8
    file  : 19lou.ico
```

```
  - width : 32
    height: 32
    color : 8
    file  : 19lou.ico
  version:
    file_version_number   : 1.2.3.4
    product_version_number: 5.6.7.8
    comments              : Comments Field
    company_name          : Company Name Field
    legal_copyright       : Legal Copyright Field
    legal_trademarks      : Legal Trademarks Field
    file_version          : File Version Field
    product_version       : Product Version Field
    product_name          : Product Name Field
    file_description      : File Description Field
    internal_name         : Internal Name Field
    original_filename     : Original Filename Field
    private_build         : Private Build Field
    special_build         : Special Build Field
```

这些代码用来配置生成的 exe 文件的一些信息，如图标、版本等；

然后执行 exerb main.exy，生成最终的可执行文件，该可执行文件比较大，可以使用 UPX 进行压缩，压缩率可以达到 70% 以上。

ASP 源码封装工具 NetBox

NetBox 是一个使用脚本语言进行应用软件开发与发布的开发环境和运行平台，使用 NetBox，可以完全使用脚本语言 (比如 VBScript，Javascript) 创建出稳定高效的应用软件，并且可以平滑移植到从 Windows 98 到 Windows .NET Server 的全部操作系统上。适用范围对于 WEB 应用，可以迅速将已有的 iis+asp 的应用平滑移植到 NetBox 应用中，除极少数高级编程外，代码不需要任何修改，同时 NetBox 还提供大量扩展部件，使得 WEB 应用更加方便。由于 NetBox 可以将全部代码最终发布成为应用程序，保护了开发人员的利益和代码的完整性。同时，NetBox 还可以方便地编写更多的桌面应用、系统服务器应用、定制网络应用等等。

简单的形容就是把 ASP 文件打包成一个 EXE 文件，并且不需要在调试的机器上安装 IIS 即可正常调试。下面

通过一个简单的实例来认识 NetBox 的使用。

1. 第一步肯定是要下载安装 NetBox。此步略过
2. 创建一个空的目录在任意位置，再在其中创建一个名为“main.box”的文件，将下面的代码内容复制到文件中：

```
Dim httpd
Shell.Service.RunService "NBWeb", "NetBox Web Server", "NetBox Http Server
Sample"
'---------------------- Service Event ----------------------
Sub OnServiceStart()
Set httpd = CreateObject("NetBox.HttpServer")
If httpd.Create("", 80) = 0 Then
Set host = httpd.AddHost("", "wwwroot")
host.EnableScript = true
host.AddDefault "index.asp"
host.AddDefault "default.asp"
host.AddDefault "index.htm"
host.AddDefault "default.htm"
httpd.Start
else
Shell.Quit 0
end if
End Sub
Sub OnServiceStop()
httpd.Close
End Sub
Sub OnServicePause()
httpd.Stop
End Sub
Sub OnServiceResume()
httpd.Start
End Sub
```

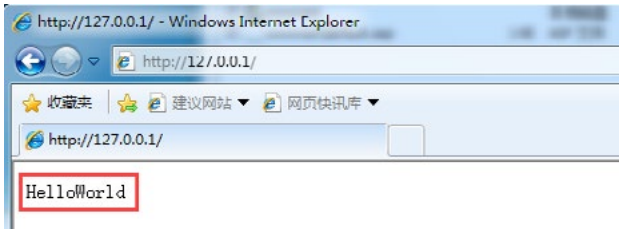
注：Set host = httpd.AddHost(" ", "wwwroot") 中的 wwwroot 就是放网站程序的目录。也就是一定要与第 3 步将建立的目录名称相同！

3. 复制 ASP 应用：上文我们提到网站主目录为 wwwroot 目录，则我们需要在第 2 步我们建立的目录下建立一个 wwwroot 目录，为了方便测试，我们编写一个简单的 asp 文件，编辑内容如下：只是在网页中输出 HelloWorld。

```
<%
Reponse.write "HelloWorld"
%>
```

4. 调试运行 asp 程序：双击我们第 2 步创建的 main.box 文件，这时候就出现了一个红色的 .b 的托盘图标

然后再在浏览器中输入 127.0.01，如果页面中出现了“HelloWorld"，就说明我们以上的配置非常成功。



5. 编译：我们可以将 ASP 程序打包成 .exe 程序，就需要执行 nbdw.exe （netbox deployment wizard 就是部署向导），然后点击“Select Folder”按钮，找到第二步建立的目录再点击“Brower”按钮，输入要生成的执行文件的目录和名称，选择好之后直接点击“Build”开始编译。

6. 找到生成的 exe 文件，双击运行，然后在浏览器中输入 127.0.0.1，同样可以看到网页中的 HelloWorld 字样。

《2017 绿盟科技恶意样本分析手册 - 文件封装篇》
由如下部门撰写

- 绿盟科技安全能力中心（SAC）

如需了解更多，请联系：

官方网站

技术博客

微信公众号

特别声明

为避免客户数据泄露，所有数据在进行分析前都已经匿名化处理，不会在中间环节出现泄露，任何与客户有关的具体信息，均不会出现在本报告中。

版权声明

THE EXPERT
BEHIND GIANTS
巨人背后的专家

多年以来，绿盟科技致力于安全攻防的研究，
为政府、运营商、金融、能源、互联网以及教育、医疗等行业用户，提供
具有核心竞争力的安全产品及解决方案，帮助客户实现业务的安全顺畅运行。
在这些巨人的背后，他们是备受信赖的专家。

www.nsfocus.com