(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 101964040 A
(43)申请公布日 2011.02.02

(21)申请号 201010280090.6

(22)申请日 2010.09.10

(71)申请人 西安理工大学
地址 710048 陕西省西安市金花南路 5 号

(72)发明人 孙钦东 王倩 马哲

(74)专利代理机构 西安弘理专利事务所 61214
代理人 罗笛

(51)Int.Cl.
G06F 21/22(2006.01)

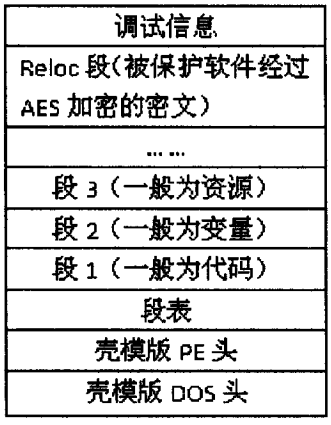
权利要求书 1 页 说明书 4 页 附图 1 页

(54)发明名称

一种基于 PE Loader 的软件加壳保护方法

(57)摘要

本发明的基于 PE Loader 的软件加壳保护方法，按照以下步骤实施：将壳模板映射到内存中，模拟 Windows 的 PE 加载器对壳模板进行基址重定位，计算出壳模板入口点所在区段的 RVA 和此区段的大小，用计算出的区段的 RVA 和区段的大小两个值作为参数，通过 SHA 算法计算出的散列值作为 AES 加密算法的密钥；将被保护软件映射到内存中并利用计算出的密钥，将被保护的软件用 AES 加密算法加密；将加密过的数据添加到壳模板的 Reloc 段中；处理将要保护软件中的特殊资源以及其附加数据提取，分别附加到壳模板文件的结尾处，并修改壳模板的 PE 结构中关于特殊资源的相应数据项；将修改好的壳模板从内存中转存到磁盘上，作为被保护后的软件即成。本发明方法显著增加了软件的安全性。



CN 101964040 A

1. 一种基于 PE Loader 的软件加壳保护方法，其特征在于：该方法按照以下步骤实施：

步骤 1、将壳模板映射到内存中，模拟 Windows 的 PE 加载器对壳模板进行基址重定位，计算出壳模板入口点所在区段的 RVA 和该区段的大小，用计算出的区段的 RVA 和该区段的大小值作为参数，通过 SHA 算法计算出的散列值作为 AES 加密算法的密钥；

步骤 2、将被保护软件映射到内存中并利用上步计算出的密钥，将被保护的软件用 AES 加密算法加密；

步骤 3、将加密过的数据添加到壳模板的 Reloc 段中；

步骤 4、处理将要保护软件中的特殊资源以及其附加数据提取，分别附加到壳模板文件的结尾处，并修改壳模板的 PE 结构中关于特殊资源的相应数据项；

步骤 5、将修改好的壳模板从内存中转存到磁盘上，作为被保护后的软件即成。

2. 根据权利要求 1 所述的基于 PE Loader 的软件加壳保护方法，其特征在于：所述步骤 4 中，要保护软件中的特殊资源以及其附加数据提取，选用特殊资源的新位置以及大小。

一种基于 PE Loader 的软件加壳保护方法

技术领域

[0001] 本发明属于计算机软件程序保护技术领域，针对软件被非法篡改和破解问题，具体涉及一种基于 PE Loader 的软件加壳保护方法。

背景技术

[0002] 软件保护是软件开发中一个不可忽视的环节，由于软件开发后要面对众多逆向分析人员的研究，要给发布的软件加一层保护壳几乎成了保护软件的一个必要步骤。现有的加壳保护技术一般是把可执行文件（这里主要指 Win32 平台的 PE 文件）的入口点 (Entry Point) 指向壳的 shellcode，并把原来的可执行文件压缩或者加密，然后把壳的 shellcode 作为目标程序一个新的段添加进去，运行时壳 shellcode 首先执行，在内存中解密或者解压，解密或者解压完成后跳到被保护程序的原始入口点 (OEP)，执行被保护程序。因此，在现有的传统壳保护下，攻击者很容易能找到被保护程序的原始入口点 (OEP)，这时壳已经把被保护的程序解压缩，解密，攻击者就可以通过内存中的明文数据，分析甚至还原目标程序，从而达到逆向的目的。

发明内容

[0003] 本发明的目的是提供一种基于 PE Loader 的软件加壳保护方法，解决现有技术中的传统加壳保护后的软件易于被攻击者破解及非法篡改的问题。

[0004] 本发明所采用的技术方案是，一种基于 PE Loader 的软件加壳保护方法，该方法按照以下步骤实施：

[0005] 步骤 1、将壳模板映射到内存中，模拟 Windows 的 PE 加载器对壳模板进行基址重定位，计算出壳模板入口点所在区段的 RVA 和该区段的大小，用计算出的区段的 RVA 和该区段的大小值作为参数，通过 SHA 算法计算出的散列值作为 AES 加密算法的密钥；

[0006] 步骤 2、将被保护软件映射到内存中并利用上步计算出的密钥，将被保护的软件用 AES 加密算法加密；

[0007] 步骤 3、将加密过的数据添加到壳模板的 Reloc 段中；

[0008] 步骤 4、处理将要保护软件中的特殊资源以及其附加数据提取，分别附加到壳模板文件的结尾处，并修改壳模板的 PE 结构中关于特殊资源的相应数据项；

[0009] 步骤 5、将修改好的壳模板从内存中转存到磁盘上，作为被保护后的软件即成。

[0010] 本发明的基于 PE Loader 的软件加壳保护方法，其特征还在于：所述步骤 4 中，要保护软件中的特殊资源以及其附加数据提取，选用特殊资源的新位置以及大小。

[0011] 本发明的有益效果是，将被保护的程序感染到加密壳的软件保护架构，加密壳具有 PEloader 功能，将被保护软件在内存中加载并执行，这种加密软件保护壳的结构大大增加了攻击者在破解或者非法篡改软件的难度，甚至无法找到真正的程序的入口点，能有效的防止逆向分析人员对软件的反汇编破解及非法篡改，大大增加了软件的安全性。

附图说明

[0012] 图 1 是未加任何保护的 PE 文件操作窗口示意图；

[0013] 图 2 是将保护的 PE 文件加密放入壳模板的 Reloc 段后操作窗口示意图。

具体实施方式

[0014] 下面结合附图和具体实施方式对本发明进行详细说明。

[0015] 本发明的方法，将要进行加壳保护的软件先经过 AES 加密，将加密后的内容放入外层的壳模板中的 Reloc 段中，加密的密钥的获取是由根据壳模板中的代码段数据通过 SHA 算法动态获得的，若逆向分析人员调试或者修改了壳模板代码段数据，则造成程序解密失败，不能正常执行，从而有效的达到保护软件的目的。

[0016] 本发明方法按照以下步骤实施：

[0017] 步骤 1、将壳模板映射到内存中，模拟 Windows 的 PE 加载器对壳模板进行基址重定位，计算出壳模板入口点所在区段的 RVA(Relative Virtual Addresses) 和此区段的大小，用计算出的区段的 RVA 和区段的大小（两个）值作为参数，通过 SHA 算法计算出的散列值作为 AES 加密算法的密钥；

[0018] 步骤 2、将被保护软件映射到内存中并利用计算出的密钥，将被保护的软件用 AES 加密算法加密；

[0019] 步骤 3、将加密过的数据添加到壳模板的 Reloc 段中，如图 2 所示；

[0020] 步骤 4、处理将要保护软件中的特殊资源以及其附加数据提取，如特殊资源的新位置以及大小，分别附加到壳模板文件的结尾处，并修改壳模板的 PE 结构中关于特殊资源的相应数据项；

[0021] 步骤 5、将修改好的壳模板从内存中转存到磁盘上，作为被保护后的软件，从而完成软件的保护。

[0022] 对于上述处理后保护壳的执行实施步骤包括：1) 壳模板根据自己在内存中的入口点所在的段的 RVA 以及段的大小，由 SHA 算法计算出哈希值作为密钥，解密自己的 Reloc 段。2) 将解密后的明文（即被保护的软件）通 过壳模板的 PE Loarder 将其加载并执行。

[0023] 本发明的可行性以及有益性效果分析，通过对一个程序加保护壳，对比加保护壳前后运行结果进行正确性分析，实验结果表明，程序加壳前后运行正常。

[0024] 图 1 是未加任何保护的 PE 文件操作窗口示意图。常用的保护壳技术是在被保护软件的基础上写入外壳代码，这样外壳初始化的现场环境（各寄存器的值）于原程序的现场环境必须是相同的，所以在外壳程序和原始的被保护软件 OEP 处之间就会有一条明显的界限，这条界限使得壳和被保护程序无论从空间还是逻辑功能上都明显的分割开来，破解者往往会找到这条界限进行攻击。图 2 是将保护的 PE 文件加密放入壳模板的 Reloc 段后操作窗口示意图。参照图 2，将被保护程序感染到的壳模板中就会使这条界限不再存在，利用壳模板的 PE Loarder 功能来加载执行被保护的软件，这种方法更加隐蔽和安全，还可以在外壳程序和中间结构中再加入大量的花指令和反调试技术。通过本发明的软件保护方法，能使破解者的攻击消耗大量的时间，破解者的所做的工作是不经济的，则达到了有益的效果。

[0025] 以下通过对比普通保护壳和本方法保护壳来分析有益效果，即通过动态分析加了

ASPACK（以 ASPACK 壳为代表）的保护程序和经本发明方法保护程序来分析有益效果。

[0026] 首先用 OllyDbg 打开经过 ASPACK 保护过的程序，可以看到加载的被保护程序断在第一条指令处，也就是外壳程序的第一条指令，在 OllyDbg 的反汇编窗口中，可以看到前三条指令分别是：

[0027] pushad

[0028] call 0045700A

[0029] jmp 45A274F7

[0030] 分析这前三条指令，由 pushad 可以得出 ASPACK 外壳程序在保存当前环境，后两条 call 和 jmp 指令在执行相应的壳程序，因为普通的保护壳软件的编写都是遵守堆栈平衡原理，也就是说在外壳程序执行前首先要保存当前的环境（各寄存器的值，主要是 ESP 和 EBP 等重要寄存器的值），通常通过 pushad/pushfd 命令或者间接的通过其他方式保存当前环境，所以可以根据 popad/popfd 或加内存断点跟踪找到跳转到 OEP 的指令。根据堆栈平衡原理加内存断点跟踪到外壳的结尾处：

[0031] popad

[0032] jnz short 004573BA

[0033] mov eax,1

[0034] retn 0C

[0035] push 0040CD6B

[0036] retn

[0037] 由上面的指令可以看出，popad 还原环境后 retn 指令将跳转到 OEP，此时当前堆栈的内容 ESP 指向 0040CD6B，所以原程序的 OEP 是 CD6B 基址是 00400000，执行完 retn，跳转到 0040CD6B 处执行，此处汇编指令为：

[0038] push ebp

[0039] mov ebp,esp

[0040] push-1

[0041] push 0042D100

[0042] 为了对比此处是 OEP，将未加过保护壳的程序调试并与 0040CD6B 处的汇编指令对比，经比较发现，未加过保护壳的程序的在开始执行处的指令和 0040CD6B 处的指令完全相同，说明 0040CD6B 处是经 ASPACK 加壳保护过的程序的 OEP，若破解者在此处 dump 内存中数据则成功脱掉保护壳并进行下一步破解。

[0043] 作为对比，用本发明的方法对要保护的软件加保护壳并分析，用 Ollydbg 加载经过本发明的方法保护过的软件，前五条汇编指令如下：

[0044] push ebp

[0045] mov ebp,esp

[0046] sub esp,134

[0047] push ebx

[0048] push esi

[0049] 在以上汇编指令中并没有保存当前环境的指令以及其他方式保存当前环境的指令，而是在执行单独的壳模版程序，壳模板是独立与被保护程序的，因此壳模板并不符合堆

栈平衡定律，壳模板是模拟操作系统加载 PE 文件的过程来加载并执行被保护的软件的，壳模版通过解密被保护程序，处理被保护程序的重定位，修正被保护程序的导入表等操作，最终执行到被保护程序的入口点，再执行被保护软件，壳模板并没有按照堆栈平衡来执行 shellcode，而是一个独立的程序。

[0050] 本发明方法作为一种新的软件保护壳，区别与以往的将壳代码写入被保护软件的思路，本方法相当于模拟了操作系统的 PE Loarder 功能，软件破解者的工作量大大增加，使用本方法还可以在壳模板中加入大量花指令，反调试技术和反 dump 技术，这样破解者的工作量及难度也将大大增加。

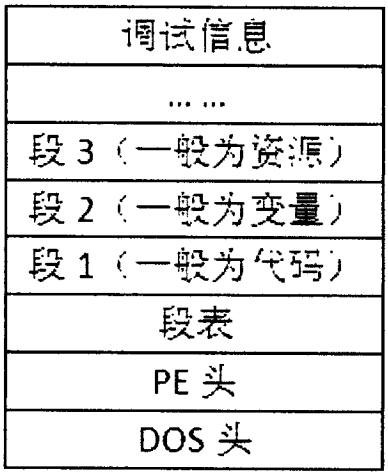

调试信息
……
段 3（一般为资源）
段 2（一般为变量）
段 1（一般为代码）
段表
PE 头
DOS 头

图 1

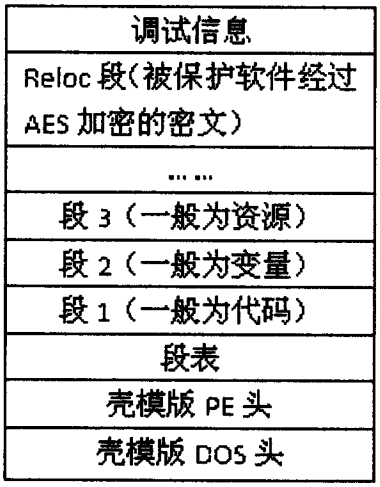

调试信息
Reloc 段（被保护软件经过 AES 加密的密文）
……
段 3（一般为资源）
段 2（一般为变量）
段 1（一般为代码）
段表
壳模版 PE 头
壳模版 DOS 头

图 2