第41卷 第6A期
2014年6月

计 算 机 科 学
Computer Science

Vol. 41 No. 6A
June 2014

基于PE文件冗余的空间多态技术

顾鼎锋[1,2] 马恒太[1]

(中国科学院软件研究所天基综合信息系统重点实验室 北京 100190)[1]

(中国科学院大学 北京 100049)[2]

摘 要 在传播过程中,越来越多的计算机病毒利用加密、多态、变形等技术来改变自身代码形态,提高自我保护能力,以躲避反病毒软件查杀。然而,传统的多态、变形技术存在体积膨胀、实现复杂等严重缺陷。针对这些问题,通过分析PE文件的框架结构,结合PE文件中存在冗余的特点,提出了空间多态的概念,并详细阐述了空间多态技术的工作原理,设计实现了空间多态引擎,最后进一步分析了空间多态技术的鲁棒性。

关键词 PE文件,恶意代码,空间多态,多态引擎

中图法分类号 TP309.5 **文献标识码** A

Space Polymorphic Technique Based on Redundance of PE File

GU Ding-feng[1,2] MA Heng-tai[1]

(Science and Technology Information System Laboratory, Institute of Software Chinese Academy of Sciences, Beijing 100190, China)[1]

(University of Chinese Academy of Sciences, Beijing 100049, China)[2]

Abstract Many computer viruses use polymorphic and metamorphic techniques to mutate their code on each replication as they propagate, thus protecting themselves from antiviruses. However, there are still some disadvantages existing in traditional polymorphic and metamorphic techniques. These techniques are too difficult to implement. What's more, it could lead to size expansion, when viruses spreading among computers. In response to these shortcomings, by analyzing the PE file frame structure, according to the characteristics that redundancy existing in the PE file, space polymorphic technique is proposed. Then, the principle of space polymorphic technique is introduced in detail, as well as the design implementation of space polymorphic engine. At last, robustness of space polymorphic technique is analysed for further research.

Keywords PE file, Malware, Space polymorphism, Polymorphic engine

1 引言

随着计算机网络的兴起与高速发展,计算机在人们的现代生活中发挥着不可估量的作用,随之而来的计算机安全问题也变得日益严峻,国内大约90%的计算机遭受过病毒攻击[1]。为了抵抗计算机病毒,各种杀毒软件应运而生。目前杀毒软件的主流技术是特征码检测。特征码是指杀毒软件从病毒样本中提取的一段可用于唯一标识病毒特征的二进制串[2,3],相当于是病毒的"指纹"。特征码检测技术发展成熟,能够快速准确识别大多数病毒,并且误警率很低,被大多数杀毒软件采用。因此,对特征码的免杀能力是恶意代码研究的重要内容。

恶意代码的自我保护技术是在保持代码自身功能不变的前提下,通过各种手段保护自己不被杀毒软件检测。目前主要的保护技术有加密、多态、变形等[1,4,5],这些技术能很好地保护恶意代码躲避特征码检测,但也有自身的弱点。本文在分析以上技术优缺点的基础上,提出一种基于PE(Portable Executable,可移植执行体)文件结构冗余的空间多态技术,该技术能很好地规避特征码检测,并且首次提出空间多态的概念,希望能以一种新的角度来探索多态变形技术,并给反病毒技术提供一定的参考和启示。

2 恶意代码自我保护技术

恶意代码自我保护技术能在不改变代码功能的前提下,通过各种手段来躲避杀毒软特征码查杀,是病毒规避杀毒软件查杀的主要技术。如果没有任何自我保护技术,恶意代码的复制传播过程可以简单用图1表示[6]。其中,$G_i(i=1,2,\cdots,n)$表示每次复制传播时生成的变种,V表示病毒体恶意代码,$File_i(i=1,2,\cdots,n)$表示被病毒感染的程序,即宿主文件。在这种情况下,恶意代码在传播过程中一直保持自身的形态不变,很容易被杀毒软件通过特征码检测来查杀。

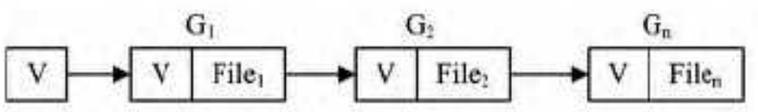

图1 恶意代码的传播过程

本文受科技部863计划(2012AA011206),中国科学院创新基金项目(CXJJ-11-S101)资助。

顾鼎锋(1989—),男,硕士生,主要研究方向为信息安全,E-mail:guformore@163.com;马恒太(1970—),男,博士,副研究员,主要研究方向为卫星组网与信息安全。

2.1 加密技术

加密技术是恶意代码进行自我保护的一种最基本技术，它能改变恶意代码自身的表现形式，隐藏其主体内容。加密后的恶意代码包括两个部分：解密指令模块和加密后的恶意代码模块，其加密变形原理如图 2 所示[7]。原始恶意代码经过加密器加密，然后在加密后的主体部分之前附加一个解密指令模块，每次病毒运行时，先运行解密指令模块，将恶意代码解密释放，最后执行解密后的恶意代码，如图 3 所示[7]。

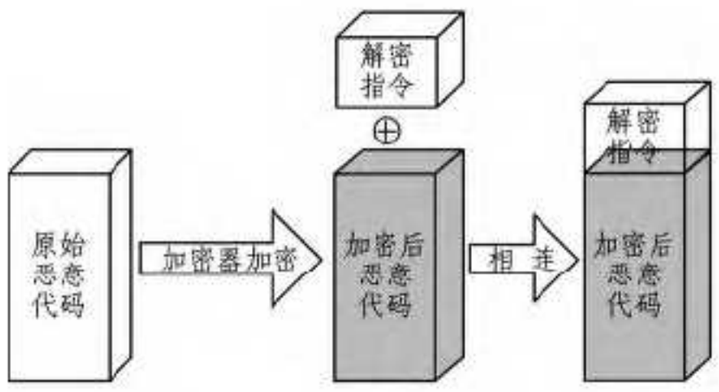

图 2 恶意代码加密原理示意图

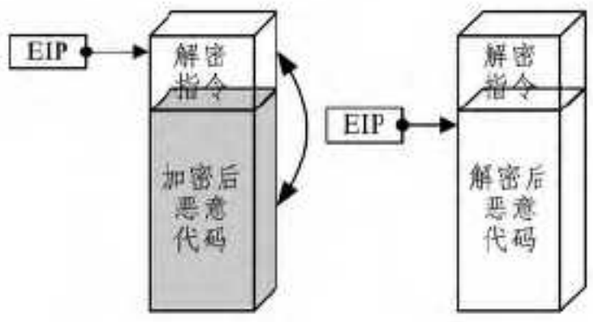

图 3 恶意代码解密原理示意图

最简单的加密算法可通过将恶意代码按字节逐一同密钥进行异或运算来实现，只要每次加密时使用的密钥不同，恶意代码的表现形式也就不同。因此，病毒体在传播感染正常程序的过程中也表现出不同的形式，如图 4 所示[6]。其中，D 表示解密指令模块，V 的不同背景表示恶意代码使用不同密钥加密后呈现出的不同形式。显然，在这种情况下，恶意代码的特征值消失了，杀毒软件无法通过特征码检测来查杀恶意代码。但是，加密技术也有其自身弱点，从图 4 中可以看出，这种技术下的解密指令模块无论在空间位置还是在自身的表现形式上都保持固定，容易成为新的特征码。因此，简单的加密技术保护能力有限，改变解密模块的不变性是关键。

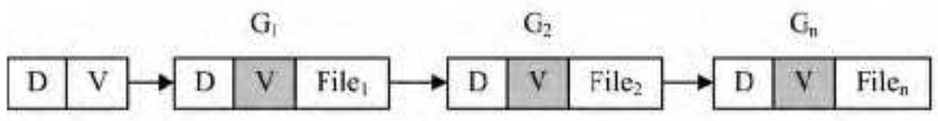

图 4 加密恶意代码的传播过程

2.2 多态和变形技术

多态技术主要是为了克服加密技术下解密指令模块相对固定的缺点。传统的多态技术主要通过指令位置变换、寄存器变换、插入垃圾指令、指令等价替换等技术，来改变解密模块的外在表现形式，但不影响其功能。经过多态引擎变换后的解密指令，能够产生大量不同的表现形式，来干扰特征码的提取与检测，其一般流程为：先加密恶意代码主体，将解密指令经过多态变化后，将其附加在加密后的恶意代码主体之前，运行时先运行解密模块，解密释放恶意代码后执行它。多态恶意代码的传播过程如图 5 所示，经过多态变化后，解密指令模块 D 也呈现出不同的形态。多态技术很好地解决了解密模块固定不变的问题，但由于多态技术一般会插入垃圾指令，而且在指令等价替换时容易扩展指令，将一条指令扩展成几条指令执行，这些都会导致解密模块体积膨胀。所以，当负载长度一定时，病毒只能具有极其有限的变种[8]。

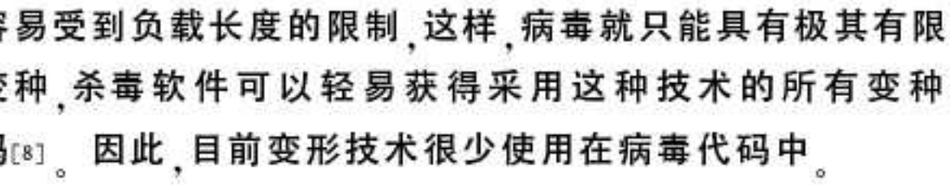

图 5 多态恶意代码的传播过程

变形技术是一种高级多态技术，从多态技术发展而来，但又提升了多态技术的能力，它进一步提升了恶意代码的生命力。变形技术不单单对解密模块进行多态变化，它对恶意代码主体甚至数据也进行了变换，使得恶意代码在存储、传播、复制的过程中每次都表现出不同的实例，但功能不变，如图 6 所示[6,9]。变形技术大大提高了恶意代码的免杀实力。但变形技术的缺陷也很明显：设计复杂，实现困难，可行性不高；而且与多态技术一样会导致文件体积膨胀。此外，现在的病毒容易受到负载长度的限制，这样，病毒就只能具有极其有限的变种，杀毒软件可以轻易获得采用这种技术的所有变种代码[8]。因此，目前变形技术很少使用在病毒代码中。

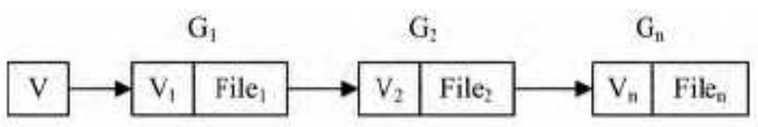

图 6 变形恶意代码的传播过程

3 PE 文件结构和冗余特点

3.1 PE 文件概述

PE 文件格式是微软制定的一种文件标准，是目前 Windows 平台上的主流可执行文件格式。Windows 平台上的病毒、木马等包含恶意代码的程序也都以 PE 文件形式存在。

PE 文件使用一个平面地址空间，所有代码和数据被合并在一起，组成一个大的结构。PE 文件框架结构如图 7 所示，它有 Dos 头(MS-DOS Header)、PE 文件头(IMAGE-NT-HEADERS)、节表(Section Table) 和节块(Sections) 4 个主要组成部分[10-12]。所有 PE 文件从 MS-DOS Header 中的 DOS-MZ-Header 开始，然后运行 DOS-STUB。DOS-STUB 是一个有效的可执行体，在 DOS 系统下执行时，它将简单显示一个错误提示，通常这部分由链接器生成。紧跟着 DOS-STUB 的是 PE 文件头，其中包含很多 PE 装载器用到的重要域和 PE 文件的一些属性值，这是 PE 文件的一个重要组成部分。在 PE 文件头与原始数据之间是节表，节表包含了节块的实际大小、对齐后大小、偏移、属性等信息。节表后面是对应的节块，常见的节块有代码节、数据节、资源节、输入表等，其中代码块主要存放 PE 文件的指令代码，数据节存放程序用到的数据，资源节中主要是图标、对话框、菜单等 PE 文件的资源，输入表则主要包括了导入函数和导入符号等信息。

Windows 操作系统在执行 PE 文件前，由 PE 装载器将文件从磁盘装入到内存中。PE 文件装入完成后，PE 装载器对 PE 文件的结构进行检查，其检查过程为：

(1)PE 文件被执行时，PE 装载器检查 MS-DOS Header 的有效性，若有效，则检查 MS-DOS Header 中 PE 文件头的偏移量，跳转到 PE 文件头；

(2)装载器检查 PE 文件头的有效性，若有效则跳转到 PE 文件头尾部；

(3)装载器读取 PE 文件头后的节表信息，并将节表信息

映射到内存，同时附上节表里定义的节块属性；

(4)PE 文件映射到内存后，装载器处理导入表的逻辑部分。

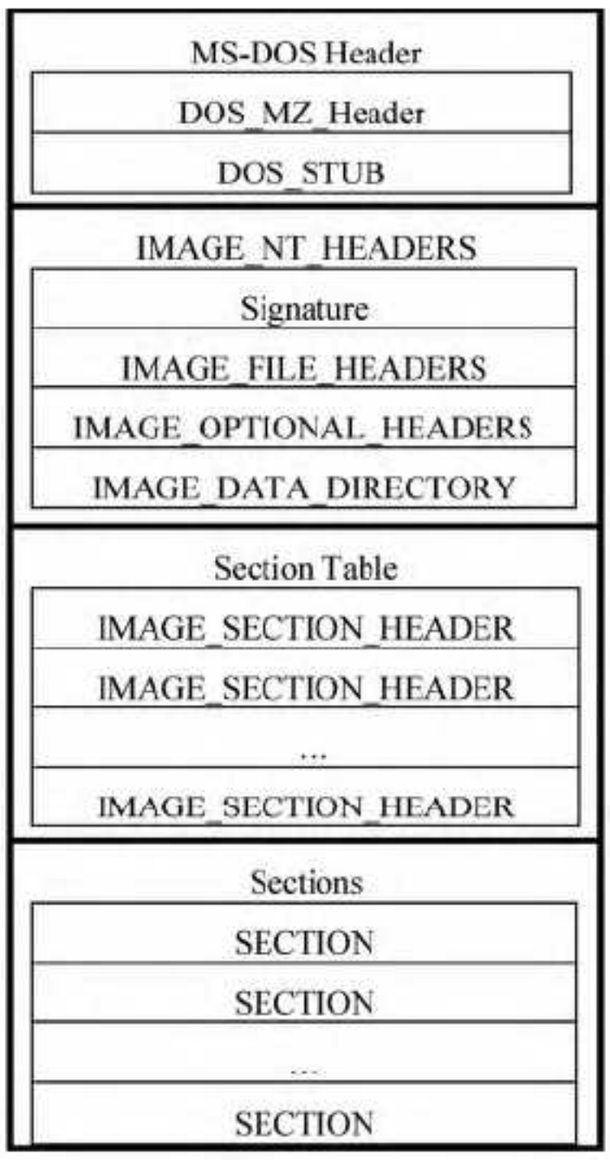

图 7　PE 文件的框架结构

3.2　PE 文件冗余特点

PE 文件中存在大量冗余空间，造成这种冗余主要有两大原因：数据对齐和保留域。

数据对齐是 CPU 结构的一部分，数据对齐的主要目的是提高 CPU 的运行效率。Windows 平台上的 PE 文件里的数据都是按要求对齐的[12]。PE 文件头里定义了节块在内存中以及在磁盘文件中的对齐值，通常，节块在内存中以 0x1000 字节对齐，在磁盘文件中以 0x200 字节对齐，不足部分以 0x00 填充。操作系统对 PE 文件的强制对齐特性，使得 PE 文件的节块中存在大量为对齐而补足的 0[13]。这些为对齐而补足的空间是可以利用的冗余空间。

PE 文件中有些保留域的值改变并不影响文件的正常加载和执行，这部分区域也是可以加以利用的冗余空间。比如 Dos 头中的 DOS-STUB 部分，这部分是为 16 位系统保留的，在 win32 平台上，这部分完全是多余的，可以替换为任意值；还有 PE 文件头部的数据结构中的部分字段，出于兼容性的考虑，PE 头部的数据结构中预留了很多字段，有的被强制设置为 0，有的未加限制，是完全可以利用的[9]。

PE 文件中的这两种冗余是由于 PE 文件的特性而自然存在的，对这部分空间的利用不会影响文件的正常执行，也不会改变文件的大小。这是一种非常理想的情况。下一节将详细阐述对 PE 文件的冗余空间加以利用，实现文件的空间多态。

4　空间多态变形技术

4.1　空间多态的工作原理

由于变形技术过于复杂，难以实现，传统的多态技术又存在传播过程中体积膨胀的缺点，可以考虑在空间上实现解密模块的多态性。

空间多态的主要原理是现行的计算机体系结构不区分数据和代码，即数据和代码在内存中没有本质区别，两者可以相互转换。PE 文件通过设定节块的属性来区分代码段和数据段，代码节具有可执行属性，而数据节通常没有。只要修改节块的属性值，添加可执行属性，那么在内存中，数据节就能像代码节一样执行。PE 文件节与节之间存在着冗余，如果修改节块的属性，使其可执行，那么这部分冗余就可以嵌入可执行代码。因此，可以利用这一部分冗余，把解密模块的指令分解到每一节的冗余中；接着修改节块的属性，使嵌入的指令能够被执行；然后以无条件跳转指令来控制指令流转，将原本固定在一起的解密指令模块分散到宿主 PE 文件的各个节块中。图 8 呈现了空间多态技术下恶意代码的传播过程，与图 6 相比，解密指令模块 $D_i(i=1,2,\cdots,n)$ 完全“寄生”于宿主 PE 文件的冗余空间中，而且解密指令也不固定在一起，每条指令随机分布到各个节块的冗余空间中，如同“消失”一般。

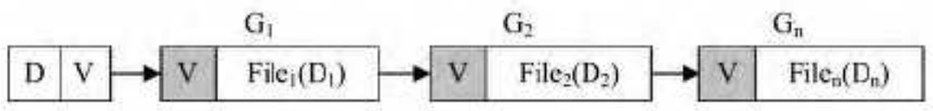

图 8　空间多态恶意代码的传播过程

空间多态技术能很好地保护恶意代码，与传统的多态技术相比，有两大优点：(1)解密模块指令分散，任一条单一的指令又很普通，不足以成为解密模块的特征码，因此，不但加密后恶意代码的特征码消失了，而且解密模块的特征码也消失了；(2)解密指令模块不占用任何额外空间，在传播过程中，不会造成文件体积膨胀，不用考虑负载限制。

如果采用异或运算进行加密，一个解密指令模块可以简单表示如下：

```
MOV          EAX,    CryptStart      B8 xxxxxxx
XOR          ECX,    ECX             33C9
DecryptLoop:
MOV          BL,     [ECX+EAX]       8A1C08
XOR          BL,     Key             80F3 xx
MOV          [ECX+EAX], BL           881C08
INC          ECX                     41
CMP          ECX,    Size            81 xxxxxxxx
JNE          DecryptLoop             75 EE
```

其中左边为基于 intel x86 处理器的汇编指令，右边为对应的以十六进制表示的字节码，CryptStart 为恶意代码主体的起始地址，Key 为加密密钥，Size 为恶意代码主体的大小，字节码中 xx 为用十六进制表示的一个字节，具体数值与恶意代码主体的起始地址、密钥和大小有关。空间多态的目标就是将解密指令模块的字节码即上面这段指令的字节码稍作调整后，随机嵌入宿主 PE 文件各节块的冗余中，然后在每条指令后加上一条无条件跳转指令来实现解密指令顺序执行，解密并释放恶意代码主体。

4.2　设计实现

空间多态引擎是实现空间多态技术的关键，直接决定了代码变形效果的好坏。该引擎主要分为 3 个模块：宿主文件分析模块、加密模块和空间多态模块，各模块之间通过全局变量来传递数据。

宿主文件分析模块主要根据宿主 PE 文件的头部信息来记录各节块的信息。在此模块，为记录各节块的信息，定义了一个重要的结构体 SectionInfo，具体定义如下。

```
typedef struct
{
  BYTE pbSecName[8]; //节块名称
  DWORD dwRedundancy; //冗余大小
  DWORD dwRealSize; //未对齐前真实大小
  DWORD dwFileOffset; //文件中偏移
  DWORD dwMemOffset; //内存中偏移
  DWORD dwAlignSize; //对齐后大小
  BOOL bFlag; //标志,初始置 false,节块嵌入解密指令后置 true
}SectionInfo;
```

空间多态引擎读取宿主 PE 文件后,根据头部信息定位到节表项,读取节块的名称、未对齐前大小、对齐后大小以及节块在文件和内存中的偏移,然后将节块对齐后大小和未对齐前真实大小相减得到冗余大小,最后将这些结果保存在结构体 SectionInfo 中。

加密模块主要就是对恶意代码主体加密,然后记录恶意代码的初始位置、大小以及加密使用的密钥等参数。该模块原理清楚,实现简单,在此不做赘述。

空间多态模块是空间多态引擎的核心模块,其基本流程如图 9 所示,具体算法如下:

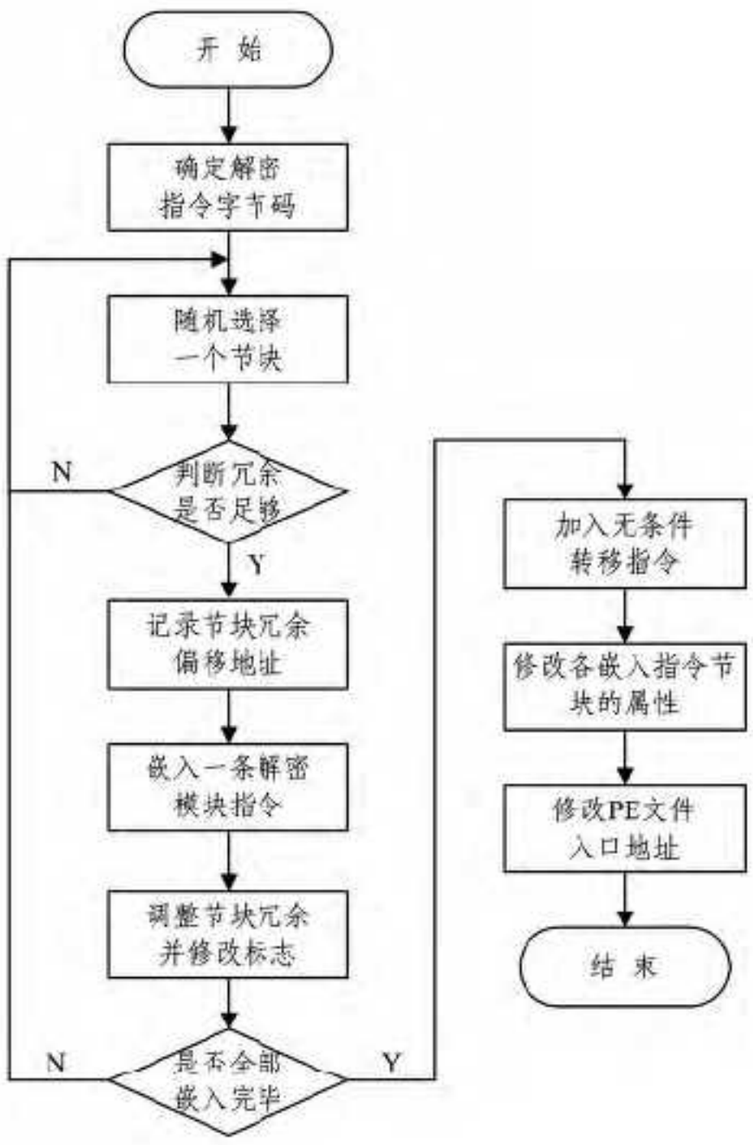

图 9 空间多态变形模块流程

(1)确定最终的解密指令字节码:将加密模块中记录的恶意代码主体的起始地址、大小和密钥等参数写入解密指令模块,将解密模块的最后一条条件转移指令分解为条件转移指令(短转移)加上一条无条件转移指令(长转移),便于实现节块间转移;

(2)从宿主文件的所有节块中随机选取一个节块;

(3)判断该节块冗余大小 dwRedundancy 是否足够嵌入一条指令的字节码(包括 5 个字节的无条件转移指令),若不满足跳转到第(2)步;

(4)记录节块冗余部分的文件偏移地址 dwFileOffset 和内存偏移地址 dwMemOffset,这两个地址分别是嵌入指令在文件和内存的起始地址;

(5)在节块的冗余部分嵌入一条解密指令的字节码,然后修改 PE 文件的校验值;

(6)调整节块的冗余值 dwRedundancy,即将原冗余值减去嵌入指令的大小(包括 5 个字节的无条件转移指令),并将标志位置 true;

(7)判断解密模块的所有指令是否嵌入完毕,如果没有完毕跳转到第(2)步;

(8)在每条嵌入的指令后加上一条无条件跳转指令,其字节码为 E9 Offset,其中 Offset 为偏移地址,共 32bit,其计算公式为:

Offset=目标地址-起始地址-5

(9)对于标志位为 true 的节块,修改其属性值,增加可读、可写和可执行的属性,修改实际大小 dwRealSize,将其加上解密指令的字节数;

(10)修改 PE 文件的代码入口地址,将其指向解密模块的第一条指令;

(11)结束退出。

4.3 技术比较

恶意代码保护技术是在不改变代码功能的前提下,保护自身不被杀毒软件检测。加密技术通过对恶意代码进行加密,来躲避特征码检测,但引入的解密指令模块位置和形态固定,又容易成为新的特征码检测目标。传统多态技术通过指令变换等操作打破了解密指令模块固定的形态,但没有解决解密指令模块位置固定的问题,并且这些操作容易造成文件体积膨胀,因此受到负载长度限制。变形技术对整个恶意代码主体进行变换混淆,彻底改变了恶意代码的形态,具有较强的保护能力,但此技术复杂,难以实现,并且同样受到负载长度限制。空间多态技术将解密指令模块分解,然后嵌入到宿主 PE 文件的冗余中,整个解密模块如同"消失"一般,此举大大增加了解密指令模块被检测的难度,进一步提高了恶意代码的保护水平。空间多态技术面临的最大挑战是受到宿主文件的冗余限制。表 1 对恶意代码的各种自我保护技术进行了总结比较。

表 1 恶意代码自我保护技术比较

技术类型	技术特点	优点	缺点
加密技术	加密恶意代码,并引入解密指令模块	实现简单	解密指令模块难以绕过特征码检测
传统多态技术	对解密指令模块在形态上进行变换	改变了解密指令模块的固定形态	解密指令模块位置固定,受到负载长度限制
变形技术	对整个恶意代码主体进行变换	保护能力强,不易被检测到	技术复杂,难以实现;受到负载长度限制
空间多态技术	解密代码模块分解嵌入到宿主文件的冗余中	大大增加了解密指令模块被检测的难度	受到宿主文件的类型和冗余限制

4.4 进一步讨论

本文方案采用空间多态技术,将原本固定在一起的解密指令分解并嵌入到被感染的宿主 PE 文件各节块的冗余中。虽然改变了解密指令的空间位置关系,但通过跳转指令控制流转,不影响其按序执行,解密恶意代码。空间多态技术利用 PE 文件的冗余空间,可以不改变文件的大小,不影响病毒的存储空间和网络传输带宽,也不会约束负载恶意代码的大小。

另一方面，PE 文件作为可执行文件，文件压缩处理时会采用无损压缩，这种操作不会影响到嵌入 PE 文件冗余空间的解密指令；而且，在嵌入解密指令后，可修改 PE 文件的校验值，避免产生文件的校验错误。因此，该空间多态技术具有很好的鲁棒性。

另外，在具体的空间多态实现中，可以根据宿主 PE 文件的空间冗余情况，采取灵活的空间多态方案。如采用块嵌入，以减少跳转指令；采用选择性嵌入，解决冗余空间较小的问题。

结束语　本文在分析传统的加密、多态、变形技术的基础上，结合 PE 文件中存在冗余的特点，提出了空间多态的概念，详细阐述了空间多态技术的实现原理以及空间多态引擎的设计实现。空间多态技术弥补了传统多态技术的一些缺点，给恶意代码的自我保护提供了一种新思路。本文方案利用了 PE 文件节块之间的冗余空间，如何利用 PE 文件中的其它冗余空间实现空间多态，甚至人为构造冗余，将整个恶意代码主体嵌入宿主 PE 文件，实现空间变形技术，是值得进一步深入研究的内容。

参 考 文 献

[1] 肖英，邹福泰. 计算机病毒及其发展趋势[J]. 计算机工程，2011，37(11)：149-151

[2] 吴伟民，范炜锋，王志月，等. 基于特征码的 PE 文件自动免杀策略[J]. 计算机工程，2012，38(12)：118-121

[3] 范吴平. Win32 PE 文件病毒的检测方法研究[D]. 成都：电子科技大学，2011

[4] 吴丹飞，王春刚，郝兴伟. 恶意代码的变形技术研究[J]. 计算机应用与软件，2012，29(3)：74-77

[5] 周梅红，刘宇峰，胡晓雯，等. 恶意代码多态变形技术的研究[J]. 计算机与数字工程，2008，36(10)：149-153

[6] Holloway R. University of London. Metamorphic Virus：Analysis and Detection[R]. Konstantinou E，Wolthusen S：Royal Holloway，University of London，2008

[7] 王清，等. 0day 安全：软件漏洞分析技术(第 2 版)[M]. 北京：电子工业出版社，2011

[8] 汪洁，王建新，刘绪崇. 基于近邻关系特征的多态蠕虫防御方法[J]. 通信学报，2011，32(8)：150-158

[9] Bashari B，Masrom M. Metamorphic Virus Detection in Portable Executables Using Opcodes Statistical Feature[C]// Proceeding of the International Conference on Advanced Science，Engineering and Information Technology. 2011

[10] Corporation M. Microsoft Portable Executable and Common Object File Format Specification[EB/OL]. Revision 6.0，1999，2

[11] 白金荣，王俊峰，赵宗渠. 基于 PE 静态结构特征的恶意软件检测方法[J]. 计算机科学，2013，40(1)：122-126

[12] 段钢. 加密与解密(第 2 版)[M]. 北京：电子工业出版社，2006

[13] 戚利. Windows PE 权威指南[M]. 北京：电子工业出版社，2011

(上接第 317 页)

法相比，其调度性能更优，作业执行时间较短，提高了用户的满意度，从而满足服务提供商和服务消费者双方协商的 SLA，如图 4 所示。

表 1　作业信息表

作业 ID	每个作业大小	截止期限	每个作业执行时间
1－10	30MB	没有	30min
11－25	20MB	没有	30min
26	30MB	120min	30min
27	20MB	120min	40min
28	30MB	120min	50min
29	20MB	120min	40min
30	30MB	120min	40min
31－50	10MB	没有	30min

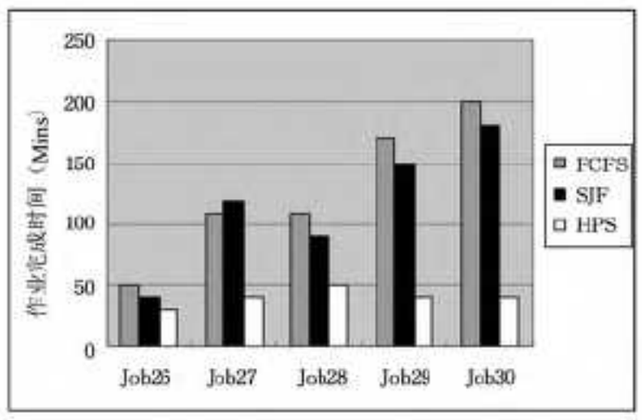

图 4　FIFO、SJF、HPS 调度算法性能分析结果

结束语　作业以分层的模式进行调度，并且使用多级队列优先保证了基于截止期限的作业能在规定时间内完成，同时也兼顾了短作业优先级别，最终作业的执行和管理是由各自节点中的节点控制器所完成。目前的实践中，只是面向 SLA 满足了用户对作业处理时间的需求。下一步的工作可以在云控制器中集成不同的任务调度器，从而实现不同用户更多类型的服务质量需求。

参 考 文 献

[1] Fito J O，Goiri I，Guitart J. SLA-driven Elastic Cloud Hosting Provider [C]// Proceeding of the 18th Euromicro Conference on Parallel，Distributed and Network-based Processing. Pisa，Italy，2010：111-118

[2] 左利云，曹志波. 云计算中调度问题研究综述[J]. 计算机应用研究，2012，29(11)：4023-4027

[3] White. Hadoop 权威指南[M]. 曾大聃，周傲英，译. 北京：清华大学出版社，2010

[4] The Apache Software Foundation. Capacity Scheduler Guide [EB/OL]. http://hadoop.apache.org/docs/r1.1.1/capacity scheduler.html，2012-03-12

[5] Polo J，Carrera D，Becerra Y，et al. Performance-driven task co-scheduling for MapReduce environments[C]// Network Operations and Management Symposium (NOMS). Osaka，Japan，2010：373-380

[6] Kc K，Anyanwu K. Scheduling hadoop jobs to meet deadlines [C]// 2nd IEEE International Conference on Cloud Computing Technology and Science(CloudCom). Indiana，USA，2010：388-392

[7] Rodrigo N C，Rajiv R，Anton B. CloudSim：a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms [J]. Software：Practice and Experience (SPE)，2011，41(1)：23-50