基于M序列的恶意代码分片插入机制

张登银,赵晓强
(南京邮电大学 计算机学院,江苏 南京 210003)

摘 要:恶意代码的生存周期包括恶意代码的产生、传播、隐藏和破坏。其中,隐藏是恶意代码生存周期中极其重要的一环。研究恶意代码的隐藏技术,了解隐藏技术的原理和关键技术,才能更好地防御恶意代码的攻击。目前,恶意代码的深层隐藏与检测技术已经成为当前计算机安全领域的一个研究热点。为了更深入地研究恶意代码,首先分析恶意代码的模糊变换和分片插入技术,然后利用M序列的随机性和状态遍历特性,提出了一种基于M序列的恶意代码分片插入机制。实验证明该机制能够有效提高恶意代码的随机性和抗分析能力。

关键词:恶意代码;模糊变换;分片插入;M序列
中图分类号:TP309 **文献标识码**:A **文章编号**:1673-629X(2010)10-0194-04

Malicious Code Splitted and Inserted Based on M Sequence

ZHANG Deng-yin, ZHAO Xiao-qiang
(College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: The life cycle of the malicious code includes generation, transmission, hidden and disrupt. Among there, hidden is extremely important in the life cycle. In order to prevent the attack of malicious code, the principle and key technology of the hiding technology must be researched. Now, the deep hiding and detecting technology of malicious code has already become one of the research hotspots in the field of computer security. In order to have a deep study on the malicious code, firstly introduces the technology of splitting and inserting and the technology of obfuscating transformation for malicious code. Then basic the randomicity and all condition reached characteristics of M sequence, a technology of splitting and inserting based on M sequence is proposed. Test showed this method can improve the randomicity and anti-analyze ability of malicious code.

Key words: malicious code; obfuscating transformation; splitting and inserting; M sequence

0 引 言

恶意代码具有非授权性和破坏性,是计算机系统安全的主要威胁之一。近年来,尽管国内外反病毒专家采用了各种各样的方法来检测计算机病毒,但是新的计算机病毒仍然不断出现,而且病毒程序越来越隐蔽。为了逃避防病毒软件的检测与查杀,计算机病毒程序普遍采用了加密、变形等大量新技术。

最初的恶意代码一般采用添加新节、插入式感染(插入单节空隙)的方法,由于改变了文件的大小,很容易被查杀。随着CIH病毒[1]的出现,分片式插入感染成为恶意代码的主流隐藏方式。但采取这种方式的恶意代码往往因为具有固定特征码而很容易被以反病毒软件为主的恶意代码检测软件所检测。因此,为了逃避这些检测工具的识别[2],恶意代码开始利用模糊变换等技术来提高自己的生存能力。模糊变换是恶意代码对抗分析和检测的一种有效手段[3]。它是在保留程序逻辑功能不变的前提下改变程序代码物理结构的一种手段。模糊变换技术的采用,使得同一种恶意代码可以产生不同的变形,几乎没有稳定代码,这样采用基于特征码的检测工具一般不能识别。文中在恶意代码模糊变换的基础上,提出一种基于M序列的恶意代码随机分片插入方法,可以进一步提高恶意代码的生存能力。

收稿日期:2010-02-26;**修回日期**:2010-05-07
基金项目:国家863计划(2007AA701302,2009AA701202)
作者简介:张登银(1964-),男,江苏靖江人,研究员,博士,CCF会员,研究方向为信号处理、信息安全。

1 恶意代码隐藏技术

1.1 模糊变换技术

模糊变换是指通过某种代码变换,将原程序演化为形式完全不同的新程序。两者虽然具有不同的表现形式,但却具有相同的逻辑功能[3]。目前,恶意代码主要使用三类模糊变换技术,即基于加密的模糊变换、基于多态的模糊变换及基于变形的模糊变换。

(1)基于加密的模糊变换。

基于加密模糊变换技术是恶意代码自我保护的一种手段。加密技术的使用,使得分析者无法正常调试和阅读恶意代码,无法掌握恶意代码的工作原理,也无法抽取特征串[4]。从加密的内容上划分,加密手段分为信息加密、数据加密和程序代码加密三种。从加密的载体上划分,加密手段分为两种,一种是对病毒代码的加密,增加反病毒人员的分析难度;另一种是对宿主文件的加密,让杀毒软件即使清除掉病毒也难以恢复原来的文件。但加密模糊变换的缺陷是其解密头不能被加密,而由于解密头的代码大都相同,所以检测工具只要将解密头作为特征码就可以检测出恶意代码的存在。

(2)基于多态的模糊变换。

多态技术能够改变自身的存储形式,从而使传统依靠特征码检测恶意代码的技术失效。成熟的多态病毒由密钥、解密代码和宿主程序构成。每次生成的解密代码和密钥都不同,使得每条解密指令都不是固定的,恶意代码每次复制自身的时候,这些代码都会随机改变。这样,不但使得恶意代码检测和防御软件的编写变得更加困难,而且还会增加反病毒软件的误报率。

(3)基于变形的模糊变换。

基于变形[5]的模糊变换是在多态的基础上更进一步,对解密代码进行了等价指令交换、指令扩展变换、指令位置变换、寄存器变换、指令收缩、垃圾指令插入等各种变化,构造出一个功能相同但代码不同的解密头,使不同病毒实体代码完全不同,不但没有固定的特征码,而且也无需还原成没有任何变化的病毒体。由于密钥和解密代码都进行了变化,因而无法找到不变的特征码,静态扫描技术也就彻底失效。

1.2　分片插入原理

采用分片插入机制[6]是为了在不扩大 PE 文件大小的前提下,将病毒测试程序分割成碎片插入到 PE 文件的节空隙中,从而更好地隐藏病毒。PE 文件格式是 Windows NT 3.1 引入的一种可执行文件格式,如图 1 所示。

从图 1 可见,PE 文件的节与节之间存在着自由空间,而这些空间一般都是以"0"填充的,所以在这些空隙中加入病毒代码是不会影响原 PE 文件正常运行的。但单个自由空间是有限的,要在单个自由空间中放入病毒代码有时是很困难的。所以需要将病毒代码分割成多个小模块,分别将这些模块插入到每个自由空间中,这样就相当于把整个病毒程序插入到 PE 文件[7]。其实,PE 文件的这种分片插入机制来源于早期 DOS 系统下的 CIH 病毒。

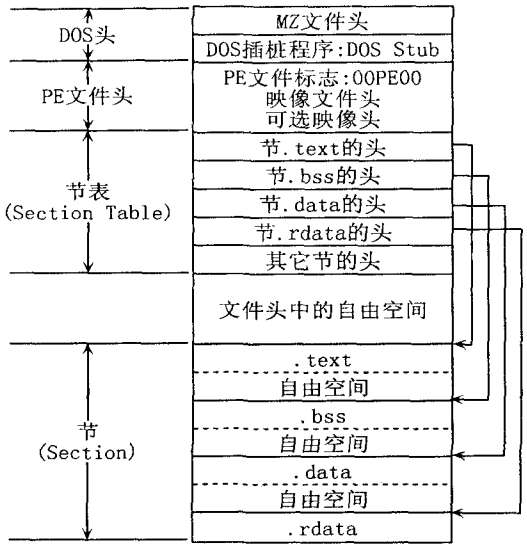

图 1　PE 文件格式

2　基于 M 序列的随机分片插入

2.1　M 序列及其特性

M 序列[8]是最长线性移位寄存器序列,是由移位寄存器加反馈后形成的。M 序列是伪随机序列中最重要的一种,由于其易于实现,具有优良的自相关特性,而广泛应用于密码技术、保密通信技术等方面。

(1)线性反馈移位寄存器序列。

反馈移位寄存器是序列密码中密钥生成器的一个重要组成部分。如图 2 所示。

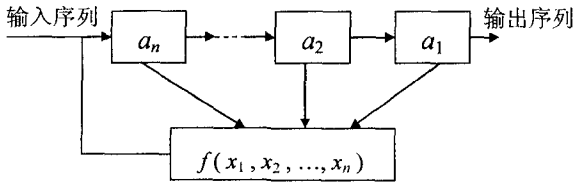

图 2　反馈移位寄存器

图中标有 $a_1, a_2, \cdots, a_{n-1}, a_n$ 的小框表示二值(0, 1)存储单元。这 n 个二值存储单元称为该反馈移位寄存器的级。$f(x_1, x_2, \cdots, x_n) = c_1 x_n \oplus c_2 x_{n-1} \oplus \cdots \oplus c_n x_1$ 是反馈函数,用于生产下一个存储单元的值,其中 $c_1, c_2, \cdots, c_n$ 称为反馈系数。每个反馈移位寄存器的状态对应一个 n 维向量,共有 2^n 种可能的状态。其中每个时刻的状态可用 n 维向量 $(a_1, a_2, \cdots, a_n)$ 表示。

如果初始状态 $s_i = (a_i, a_{i+1}, \cdots, a_{i+n-1})$, $a_{i+n} = f(a_i, a_{i+1}, \cdots a_{i+n-1})$, 从而下一个状态 $s_{i+1} = (a_{i+1}, a_{i+2}, \cdots, a_{i+n})$, 由此得到的一系列数据 $a_1, a_2, \cdots, a_{n-1}, a_n$ 称为线性反馈移位寄存器序列。

初始输入为非零的 n 级线性反馈移位寄存器序列具有周期性,其周期 $r \leqslant 2^n - 1$。

(2)M 序列。

定义 1:当 n 级线性移位寄存器产生的序列的周期为 2^n-1 时,则称其为 n 级 M 序列。n 级 M 序列具有 2^n-1 个状态,不同的 M 序列只能是这些状态的不同排列。可见 n 级 M 序列具有很好的随机特性。

(3)M 序列的生成。

设 n 级反馈移位寄存器的反馈系数为 $c_1,c_2,\cdots,c_n$,则输出序列满足递推关系 $a_{i+n+1}=c_1a_{i+n}\oplus c_2a_{i+n-1}\oplus\cdots\oplus c_na_{i+1},i=0,1,2,\cdots$,这种递推关系可用一个一元高次多项式 $p_n(x)=1+c_1x+c_2x^2+\cdots+c_nx^n$ 表示,称为该线性移位寄存器的特征多项式。线性反馈移位寄存器序列可由 $p_n(x)$ 完全确定。

定理 1:设 $p_n(x)$ 为 n 次多项式,是序列 $\{a_i\}$ 的特征多项式,$p_n(x)\mid(x^m+1)$,则 $\{a_i\}$ 的周期是 $r\mid m$。

定义 2:$p_n(x)$ 是 n 次不可约多项式,若 $p_n(x)$ 阶为 2^n-1,则称 $p_n(x)$ 是 n 次本原多项式。

定理 2:$\{a_i\mid i=0,1,\cdots\}$ 为 n 级 M 序列的充要条件是其特征多项式 $p_n(x)$ 为 n 次本原多项式。

常见的本原多项式如表 1[9,10] 所示。

表 1 常见的本原多项式($1<n\leqslant 20$)

x^2+x+1	$x^{12}+x^6+x^4+x+1$
x^3+x+1	$x^{13}+x^4+x^3+x+1$
x^4+x+1	$x^{14}+x^{10}+x^6+x+1$
x^5+x^2+1	$x^{15}+x+1$
x^6+x+1	$x^{16}+x^{12}+x^3+x+1$
x^7+x^3+1	$x^{17}+x^3+1$
$x^8+x^4+x^3+x^2+1$	$x^{18}+x^7+1$
x^9+x^4+1	$x^{19}+x^5+x^2+x+1$
$x^{10}+x^3+1$	$x^{20}+x^3+1$
$x^{11}+x^2+1$	

2.2 M 序列的随机分片插入

通过模糊变换对恶意代码进行加密及变形之后,恶意代码连同解密头需要分片插入 PE 各个节空隙来隐藏自己。传统的分片插入机制是线性的,即顺序将恶意代码插入各个节空隙。这样,恶意代码一旦被跟踪,其恶意行为很容易被分析。如果先将恶意代码随机排序,再通过分片插入节空隙中,那么即使被跟踪,其强大的抗分析能力也会使得分析者无功而返。基于 M 序列的恶意代码隐藏流程,如图 3 所示。

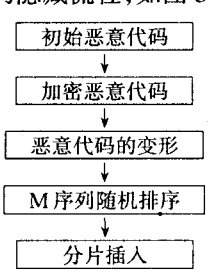

图 3 恶意代码的隐藏流程

M 序列工作的基本原理是:对于加密变形后的连续恶意代码,首先提取这些代码的地址索引,形成一个集合 $S=\{1,2,\cdots,\text{size}\}$,其中 $2^{n-1}<\text{size}<2^n$。然后选取一个 n 次本原多项式 $p_n(x)$。这样,可以生成一个以该本原多项式为特征多项式的 M 序列,且周期为 2^n-1。以该 M 序列的输出状态序列对应的非 0 值作为地址索引,则能够对恶意代码主体进行随机地址的排序。M 序列工作的具体算法描述如下:

```
CodeSize = virus; //恶意代码主体大小
n = log2 CodeSize + 1; //计算本原多项式的次数
int array[CodeSize]; //用于存放循环序列
StartAddr = a1a2…an = 000…01; //二进制表示 LFSR 的首状态,一共 n 位
int k = 0;
array[k] = 000…01; //首地址存入数组
查表得到本原多项式 pn(x) = 1 + c1x + c2x^2 + … + cnx^n
Index = StartAddr;
do
{
k ++;
if(Index < CodeSize)
  {Reorder();}//如果没有超出范围,恶意代码重新排序
an+1 = f(a1,a2,…,an) = c1an ⊕ c2an-1 ⊕ … ⊕ cna1;
a1a2…an = a2a3…an+1;
a[k] = a1a2…an;//存放的循环序列用于病毒代码的重组
Index = ana1-1…a1;//指向下一地址
} while(Index! = StartAddr&&k < CodeSize)
```

例如,恶意代码主体代码长度为 14,即 size = 14,则通过 $2^{n-1}<\text{size}<2^n$ 得出 $n=4$,查询表 1,获取相应的本原多项式为 x^4+x+1,即 $c_1=c_4=1$。

起始地址 StartAddr = 1 = $(0001)_2$,则线形反馈函数为 $f(a_4,a_3,a_2,a_1)=a_4+a_1$,则输出序列为:000111101011001000,000111101011001000,输出状态为 1,3,7,(15,)14,13,10,5,11,6, 12,9,2,4,8。

恶意代码经过 M 序列变形前后的顺序,如图 4 所示。

1	2	3	4	5	6	7	8	9	10	11	12	13	14

变形前的顺序

1	3	7	14	13	10	5	11	6	12	9	2	4	8

图 4 简单 M 序列排序

需要注意的是,线性移位寄存器的初始状态就是恶意代码的起始地址。重新排序的内容只包括 M 序列状态中对应的索引值不大于恶意代码的长度 size 的内容。大于 size 的内容不作任何处理。

经过 M 序列分片插入后的恶意代码在执行之前需要重组代码。首先申请恶意代码大小的内存块,从

PE 各个节空隙中循环取出代码碎片,然后按照数组 array[k]中的序列将恶意代码的原顺序进行还原,再通过首地址定位到病毒的入口地址,就可以顺利地执行病毒程序。这部分工作由一个引导程序完成,由于引导程序中不含解密的特征码,所以不会被杀毒软件查杀。

通过 M 序列变换,恶意代码的排序有了更大的随机性,也使得其抗分析能力大大增强。

3　实验及结果分析

3.1　实验环境

针对文中提出的基于 M 序列的分片插入方法,通过一组对照实验来验证其效果。实验环境如下:

(1)操作系统:Windows XP;

(2)检测软件:瑞星 2009,趋势网络版,卡巴斯基 6.0 单机版;

(3)感染程序:CIH 病毒,修改注册表的恶意代码;

(4)感染方式:普通分片插入,基于 M 序列的分片插入。

3.2　实验结果

本实验中,笔者使用瑞星 2009、趋势网络版和卡巴斯基 6.0 单机版三款检测软件,先后对著名的 CIH 病毒以及本人设计的一个修改注册表的恶意代码进行了对照测试。在测试中,首先分别使用三种防病毒软件对普通分片插入的感染程序进行扫描,然后再对基于 M 序列分片插入的感染程序进行扫描,测试结果见表 2 和表 3。

表 2　CIH 实验结果

实例	瑞星 2009	趋势网络版	卡巴斯基 6.0 单机版
普通分片插入	未被检测	报警	报警
基于 M 序列的分片插入	未被检测	未被检测	未被检测

表 3　修改注册表的恶意代码实验结果

实例	瑞星 2009	趋势网络版	卡巴斯基 6.0 单机版
普通分片插入	未被检测	报警	报警
基于 M 序列的分片插入	未被检测	未被检测	未被检测

从表 2 和表 3 可见,无论是传统的 CIH 病毒还是自己设计的修改注册表的恶意代码,在经过基于 M 序列的分片插入之后,瑞星 2009、卡巴斯基和趋势这三种检测软件均不能检测出来,表明恶意代码的生存能力相对于普通分片插入有了进一步的提升。

4　结束语

从简单的添加新节到分片插入[11],恶意代码在与检测软件的对抗中不断发展、进步。目前,恶意代码融合了多种技术来躲避杀毒软件的检测,并且已经出现了直接破坏杀毒软件然后完成恶意行为的恶意代码。特别是模糊变换、分片插入技术的运用,使得恶意代码更新的速度飞速增长,而特征码提取的相对滞后,使得恶意代码的检测技术显得相对低效。

恶意代码正向着破坏性和传染性越来越大、隐蔽性越来越高、保护性越来越强、技术越来越复杂的方向不断发展[12]。基于 M 序列的恶意代码分片插入机制,能够有效提高程序代码的随机性及其抗分析能力。文中通过对恶意代码的研究,旨在为安全防护和信息对抗提供参考。

参考文献:

[1] 张仁斌,李　钢. 计算机病毒与反病毒技术[M]. 北京:清华大学出版社,2006:215－219.

[2] Zhang Xi, Saha D, Chen Hsiao－Hwa. Analysis of virus and anti－virus spreading dynamics[C]//Global Telecommunications Conference, 2005. GLOBECOM' 05. [s. l.]: IEEE, 2005:1－5.

[3] Collberg C S, Thomborson C. Watermarking, Tamper－proofing and Obfuscation Tools for Software Protection[J]. IEEE Transactions on Software Engineering, 2002, 28(8): 735－746.

[4] Sherriff L. Encryption vs antivirus[EB/OL]. 2001. http://www. theregister. co. uk/2001/02/07/encryption-vs- antivirus.

[5] 张　勐,杨大全,辛义忠,等. 计算机病毒变形技术研究[J]. 沈阳工业大学学报,2004,26(3):309－312.

[6] 罗云彬. Windows 环境下 32 位汇编语言程序设计[M]. 北京:电子工业出版社,2002:667－728.

[7] Sun Jianhua, Qin Jizha, Chen Shu, et al. A Virus Immunization Model Based on Communities in Large Scale Networks[C]// Eighth ACIS International Conference, 2007. Qingdao: [s. n.], 2007:917－922.

[8] 章照止. 现代密码学基础[M]. 北京:北京邮电大学出版社,2004.

[9] Zivkovi M. A Table of Primitive Binary Polynomicals[J]. Mathematics of Computation, 1994, 62(205): 385－386.

[10] Shift Registers and Counters[EB/OL]. 1997. http://www. eelab. usyd. edu. au/digital_ tutorial/part2/register07. html.

[11] 张登银,洪福鑫. 典型 shellcode 引擎特征检测方法研究[J]. 计算机技术与发展,2010,20(1):18－22.

[12] 郑　羽,杨春生,于　江. 加密与解密实战入门[M]. 北京:电子工业出版社,2006.