基于 API 函数及其参数相结合的恶意软件行为检测*

韩兰胜[1]，高昆仑[2]，赵保华[2]，赵东艳[3]，王于波[3]，金文德[4]
(1. 华中科技大学 计算机学院 信息安全研究所，武汉 430074；2. 中国电力科学研究院 信息工程研究所，北京 100000；3. 国网电力科学研究院，北京 100000；4. 浙江省电力公司，杭州 310000)

摘　要：提出了一个较灵活、可扩展的方法，它是基于更细致的运行特征：API 函数调用名、API 函数的输入参数及两种特征的结合。抽取以上三类特征，借助信息论中的熵，定义了恶意代码信息增益值的概念，并计算相应的 API 及其参数在区分恶意软件和良性软件时的信息增益值，进而选择识别率高的特征以减少特征的数目从而减少分析时间。实验表明，少量的特征选取和较高的识别率使得基于 API 函数与参数相结合的检测方法明显优于当前主流的基于 API 序列的识别算法。

关键词：恶意软件检测；基于行为检测；API 调用名；输入参数；信息增益值
中图分类号：TP309　　**文献标志码**：A　　**文章编号**：1001-3695(2013)11-3407-04
doi:10.3969/j.issn.1001-3695.2013.11.052

Behavior detection of malware based on combination of API function and its parameters

HAN Lan-sheng[1], GAO Kun-lun[2], ZHAO Bao-hua[2], ZHAO Dong-yan[3], WANG Yu-bo[3], JIN Wen-de[4]
(1. *Institute of Information Security, School of Computer Science, Huazhong University of Science & Technology, Wuhan* 430074, *China*; 2. *Institute of Information Engineering, China Electric Power Research Institute, Beijing* 100000, *China*; 3. *China Electric Power Research Institute, Beijing* 100000, *China*; 4. *Zhejiang Electric Power Corporation, Hangzhou* 310000, *China*)

Abstract: This paper proposed a more flexible and scalable method, which was based on more detailed operation characteristics: API function call name, input parameters in API functions, the two types of the combination of features. It extracted three categories above, defined the concept of the information gain value of malicious code with the help of the entropy in information theory, then, calculated the information gain value of the corresponding API and its parameters in distinguishing the malware and begin software. And then selected the characteristic having higher recognition rate to reduce the number of features and analysis time. Experiment show that, a small amount of feature selection and higher accuracy makes it more superior to the algorithm of API based detection of malware.

Key words: malware detection; behavior-based detection; API calls; input parameters; information gain value

0　引言

随着网络和计算机应用的普及，软件程序已深入人们的生活，利用软件系统漏洞的恶意软件潜入网络和计算机系统，盗取或破坏他人信息的事件层出不穷而且每年都在急速增加。迈克菲公司报告[1]，仅 2010 年，多达上千万的恶意软件被检测到，然而据安全专家估计，将近有一倍的恶意代码没有被及时检测出来。这种增长趋势也使得传统的基于特征码的病毒监测方法无力应对，原因有：a) 基于特征码的检测技术只能识别已知恶意代码，而对未知的恶意代码无能为力；b) 基于特征码的识别技术必须依靠其收集到的恶意代码的特征库，受时间和存储空间的限制，特征库不可能装下所有的特征码，如当前较完善的杀毒软件 Kaspesky 常规保存约 130 万个特征码。因此据初步估计，超过近 50% 的恶意软件，在出现后的三个月内无法被检查出来[2,3]。因此必须研究开发一种不依赖于具体特征码库的分析技术来应对恶意代码的检测技术。

当前不少信息安全专家采用动态技术来检测恶意软件。他们采用分类技术将未知的恶意软件分类到已知的检测类型。常用的方法是，选用某些 APIs 和其他动态字段来检测某类恶意软件[4]。利用 API 调用产生的临时性的信息建立模型来区分良性软件和恶意软件，但是该方法数据庞杂，分析过程复杂，实用性很差。

文章想法基于如下事实：尽管恶意软件与某些合法程序有相似的行为，但必然有不同行为。比如，它们创建一个特定的文件时，它必须调用某些带特定参数的 API，这些 API 会是一样的，但这些 API 函数参数却不一定都相同。由此看来，区分代码的行为仅依据调用的 API 是不够的，还要考虑包括 API 函数参数在内的其他因素，由此可见，行为的界定取决于对行为

收稿日期：2013-02-09；**修回日期**：2013-03-18　　**基金项目**：国家自然科学基金资助项目(61272003,61272405)
作者简介：韩兰胜(1972-)，男，湖北武汉人，副教授，博士，主要研究方向为信息安全、网络安全、计算机病毒等(hanlansheng@hust.edu.cn)；高昆仑(1972-)，男，主要研究方向为电力系统及其自动化；赵保华(1984-)，男，主要研究方向为通信工程；赵东艳(1970-)，女，主要研究方向为自动控制；王于波(1969-)，男，主要研究方向为自动控制；金文德(1966-)，男，主要研究方向为数据挖掘.

的定义和描述。本文首先运行二进制文件,动态抽取 API 和相应的参数作为数据集;引入并定义信息增益值,作为抽取特征的重要依据;最后,利用有效的分类法获得 250 多个特征,取得了 95% 的准确率,只有很小的误报率。另外本文也参考了诸如 TP、FP、J48 等方法,用来比较本文所提方法和以前方法。

1 相关工作

当前两种常见的恶意软件的检测方法[5,6],即静态检测和动态检测。静态检测是最普遍的方法,静态地追寻、匹配可执行程序的字节或指令的顺序。而动态的方法是让恶意软件在一个安全的虚拟环境下运行,实时监测其 API 调用,从而实现分析和探测恶意软件的样本。

静态检测主要的优点是可以完整检测、匹配二进制代码的整个运行过程,因此可以更精确地获取其特征,是当前的主流技术。但静态检测无法检测到那些采用了加壳、打包、反追踪和反解压技术的文件,而这些技术通常被用在多态或者变异的恶意软件中来规避检测[7]。近几年,越来越多的反病毒专家开始研究用动态检测方法来自动分析恶意软件[8]。动态检测的主要优势是可应对某些变形技术,诸如加壳、打包等,更重要的是对于新出现的恶意代码,也需要动态技术的检测和判别[9]。动态检测技术不依赖庞大的特征库,但是它们也需要一定的计算资源,并且只能检测实时行为。

对于恶意软件样本分类的研究,大多是抽取静态和动态的特性来创建分析模型[10]。Hu[11] 采用静态方法,抽取了 PE 文件头部和中部的信息,通过分析这些信息特性来评价样本的区分效果,由于头部信息很容易被修改,因此以前的行为检测主要是将头部和中部信息结合在一起[12,13]。目前为止还没有文献提出单独利用 PE 头部或中部信息对样本检测的影响。本文也将沿着这个思路来评估它们单独和联合时对样本检测的影响。

Tian 等人[6] 提出一种动态分类方法,让代码在虚拟机环境中运行,通过 HookMe 动态工具运行 30 s,将一些行为信息从追踪报告中抽取出来,每个 API 调用和其他抽取的特征都被视做字符串信息,然后使用 WEKA 分类器进行分类。通过对一千多个恶意软件和四百多个良性软件检测后,识别率达到 95%。

Sami 等人[14] 提出把行为的空间特征和时间特征相结合。空间特征是指工具、空间指针和空间大小的信息;时间特征是指通过马尔可夫链调用 API 模型的顺序。他所提出的模型系统只在内存管理器和文件 I/O 分类时监测识别样本的 API 调用。

总之,当前的行为特征选取过程与 Tian 的方法类似,它们的特征选取包含大量 API 特征,数据量较大,分析时间较长,识别率较低。本文提出依据 API 及其参数相结合,分析它们在恶意代码中的表现,借助信息论中熵的概念,定义 API 及其参数在区分恶意代码时信息增益值的概念,减少了行为特征的数量,只有 200 多,这样可以加快识别的速度。为提高识别的准确率,适当延长了检测程序运行时间,扩展至最多 150 s。其好处是尽量减小特征库,增加检测软件的灵活性。另外采用了一个包含 950 个恶意软件样本和 450 个正常软件的大型数据集,最后本文也采用了 TP、FP、J48 等方法将本文的方法与以前的方法作比较,尽管恶意软件和数据集都比以前多,使用的方法也更简单,但是,取得了更高的准确率。

2 算法

2.1 算法描述

从每个程序样本中选取一组特征值,将选取出来的特征传给分类器。除 APIs 外,适当挑选其他有效分类的参数。在虚拟机中运行恶意软件,并监测它们的行为,抽取出来的 API 及参数作为特征向量,按照本文提出的信息增益的算法,计算它们的信息增益值。最终,构造出测试数据集,使用 WEKA 库来分类和验证数据集,下面详述这个过程:

将程序是否调用相关 API 及相应的参数看做一种信息(特征),并将程序是否为恶意代码看做一个事件。通过计算该信息对事件的影响来衡量信息对判定的贡献程度,从而进一步得出对应特征属性的重要程度,完成对特征属性的筛选。换句话说,计算调用了特定 API 函数的程序是恶意代码的概率与未调用该 API 的程序是恶意代码概率之间的差值,差值越大说明该 API 出现在病毒中的几率越大,通过计算保留检测中作用较大的特征。

定义一个随机变量 X 的熵值为

$$H(X) = -\sum_{i=1}^{k} p_i(X=v_i)\log_2 p_i(X=v_i) \tag{1}$$

其中:X 共有 k 种取值,p_i 是 X 取 v_i 的概率。高熵值意味着 X 遵从一个均匀分布且缺少区分度,低熵值则代表 X 遵从一个区分度较大的分布。另外,定义在已知随机变量 Y 的条件下,随机变量 X 的条件熵为

$$H(X|Y) = -\sum_{j=1}^{m} p_j(Y=v_j) \times H(X|Y=v_j) \tag{2}$$

$H(X|Y)$ 表示在已知 Y 的条件下,变量 X 仍然存在的不确定度。在本文中,由于随机变量 X 在此处仅有两种取值,即恶意代码与非恶意代码。所以其熵值 $H(X)$ 可以表示为

$$-\frac{|v|}{|s|}\log_2\frac{|v|}{|s|} - \frac{|b|}{|s|}\log_2\frac{|b|}{|s|} \tag{3}$$

其中:v 表示恶意代码程序,b 表示正常程序,$|x|$ 表示某种集合的基数,则 $|s| = |v| + |b|$ 表示程序的总数量。令 Y 为一布尔值,$Y=1$ 表示 S_1 含有属性 Y,$Y=0$ 表示 $S_0 = S - S_1$,即表示某程序是否调用了特定的 API 函数。那么 $H(X|Y)$ 可表示为

$$-\sum_{k=0}^{1}\frac{|s_k|}{|s|}\left(-\frac{|v_k|}{|s_k|}\log_2\frac{|v_k|}{|s_k|} - \frac{|b_k|}{|s_k|}\log_2\frac{|b_k|}{|s_k|}\right) \tag{4}$$

由此,可定义信息增益为

$$IG(X|Y) = H(X|Y) - H(X) \tag{5}$$

由式(5)可知,IG 表示在已知 Y 的情况下,X 信息的增加值,也就是某一行为特征的信息增益,即表示这一行为特征 API 调用对检测恶意代码所起到的作用大小。根据定义,IG 的值越大,说明这一特征的区分度越高,分类能力越强。反之,那些 IG 较小的特征则对程序的鉴别贡献较小。因此,在处理特征集时可以计算这些特征的信息增益,将其中 IG 值较小的特征去掉,保留下来的 API 则会拥有较大的区分度,筛选过程也会减小特征集的维数,从而实现对所收集到 API 序列的精简,提高分类的效率。

在计算 API 及其相应参数的信息增益值时,本文是将具体的 API 及其参数值作为一个具体的信息计算,这显然增加了不少计算量,但相对于后期的检测分析所节省的时间,这是值得的。

经过筛选之后,获取 IG 值较高的 API 构成行为特征集。本文收集了恶意代码中部分最常见,且经过计算后区分度较高的 API。

2.2　数据采集

选择一个包含 450 个正常软件和 950 个恶意软件的样本用于这项研究。恶意软件含有木马、恶意代码、蠕虫等。良性的文件是从 Windows 系统和广泛使用的应用式工具软件中选择的。不同类别的恶意软件样本和它们的数量以及百分比,如表 1 所示。

表 1　不同类别的恶意软件样本和它们的数量以及百分比

类型	文件数	样本比例	类型	文件数	样本比例
virus	50	3.6	hacktool	180	12.9
trojan	100	7.1	P2Pwarm	140	10.0
backdoor	200	14.3	exploit	80	5.7
constructor	200	14.3	Bengin	450	32.1

2.3　特征选取

程序在虚拟机中运行,通过 WINAPIO VERRIDE32 工具来监测[15]。每个样本运行最长可达 150 s,当然也可以提前结束。选取这个时间是因为对于绝大多数恶意软件而言,150 s 是足够执行它的所有程序,在分析完成之后,生成的运行追踪日志存储为 XML 文件,然后系统被转换至初始状态,运行下一个文件,这个过程会持续到所有的文件都被检测结束为止。

从九个最重要的动态链接库中选择 126 个 API 构成记录调用的子集。这些链接库包括 kernel32. dll、advapi32. dll、shell32. dll、wininet. dll、ntdll. dl、user32. dll 和 ws2_32. dll 等。每个 DLL 函数在表 2 中都给出功能说明。这些 API 用做创建、删除或者修改注册表中的键值,比如创建一个进程、文件或者目录,搜索、删除或者移动一个文件,创建关联等。依据式(1)~(5)计算,选取贡献较大的 API。

表 2　选取的主要 dll 及功能

名称	功能
kernel32. dll	用于内存管理和资源处理的底层操作系统函数
advapi32. dll	支持众多 API 包括许多安全性和注册表调用的高级 API 服务库
ntdll. dll	控制 NT 系统功能的 NT 内核层动态链接库
ws2_32. dll	网络和互联网应用程序利用所包含的 Windows 套接字 API 来操纵它们的连接
wininet. dll	使应用程序能够访问标准网络协议,如 FTP 和 HTTP
gdi32. dll	包含用于画图和显示文本的函数
shell32. dll	用于打开网页和文件,建立文件时的默认文件名的设置等大量功能
odbc32. dl	用于 ODBC 数据库查询相关文件
user32. dll	用于消息处理、计时器、菜单和通信的窗口管理函数

说明:由于日志文件不超过 100 KB,仅调用 2~15 个 API,所以被忽略,这导致了这些文件不能成功运行。由于缺少必要的文件来执行这些程序,一些可执行格式无法被识别,或者某些恶意软件识别出监测环境,不再表现出它们的恶意行为。

总共运行了 2 000 个可执行文件,其中 110 个少于 100 KB 的日志被略去,生成的日志样本在图 1 中说明。

图 1　部分可执行程序的追踪报告

正如先前所述,从日志中抽取各 API 及其相应的参数特征,本文中只选取了 API 名字和参数。首先,分别单独地统计和分析有效的参数值(对单独参数不必计算信息增益值)和 API 调用并计算其信息增益值,然后再将参数值和 API 调用结合到一起来统计分析并计算其整体信息增益值、比较这三种方法的效果。

为此,需要建立三种特征集,对每一种特征要经过相同的验证过程,所有的 API、参数被抽取出来,命名为 ListP 的字符串。

接下来的例子给出了特征选取的过程,考虑以下三个日志文件(Logfile):

Logfile1:{Api1(P1,P2,P3), Api2(P4,P1,P5), Api3(P2), Mal}

Logfile2:{Api1(P1,P2,P3),Api5(P6,P7),Api3(P2),Mal}

Logfile3:{Api6(P10,P11),Api2(P8,P9,P5),Api6(P10,P11), Ben}

从运行日志中抽取的字符串列如下:

APIString = {Api1,Api2,Api3,Api5,Api6}

PString = {P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11}

APString = {Api1-1P1, Api1-2P2, Api1-3P3, Api2-1P4, Api2-2P1, Api2-3P5, Api3-1P2, Api5-1P6, Api5-2P7, Api6-1P10, Api6-2P11, Api2-1P8, Api2-2P9, Api2-3P5}

例子中 Api1、Api2 表示 API 的名字,字母表示参数值,在 AP 字符串列表中,Api1-2P2 特征表示 Api1 的第二个参数在内存中的地址为 2P2。

2.4　特征选取

在特征选取的过程中,样本数量是确定的。某个特征在一个程序中可能被调用多次,本文仅考虑特征所在文件被使用的总次数。具体描述如下:

假设想选取字符串列表 APIString 对应的特征,需建立一个表 3。频率分布显示了每种特征所属样本的数目。由于某些特征不易被检测到,可以忽略这些频率(frequency)小于某一个特定临界值的特征,这也有助于减少特征的数量。

按式(1)~(5)计算各 API 及参数的信息增益值,根据增益值将特征数目减少至 1 500 个以内。因此减少了处理的时间。突出的特征被选取出后,为每个被检测样本生成一个特征向量空间,具体算法是:

```
procedure Creat_Sector_Space()//特征向量空间构造函数
for Each Sample;// 对每个样本
check ListP;//检查每个字符串列表的特征
if Feature_ exist_ Logfile()//如果选取的特征值在日志文件中存在
then Vector_Space_File = 1;//那么特征的值被设置成 1
else Vector_Space_File = 0;//反之为 0
end
```

表 3 就是前述中对恶意代码检测比较突出的几个 API 的频率及信息增益值(具体版本环境 Windows 9X/Windows 2000/Windows XP/Windows 2003)。从表中也可以看出,API 频率与其信息增益并不是相称的,尽管信息增益相对计算较繁琐,但一旦计算出来,则相对稳定。

表 3　恶意代码检测中常见的 API 的频率及信息增益值

Feature	API Name	Frequency(≥2)	*IG*(≥2%)/%
user32. dll	APi1	5	12.18
kernel32. dll	APi2	4	21.30
advapi32. dll	APi3	3	8.12
ntdll. dll	APi5	3	4.53
ws2_32. dll	APi6	3	2.31
wininet. dll	APi7	2	2.1
gdi32. dll	Api8	2	2.12
shell32. dll	APi9	2	2.21
odbc32. dll	APi10	2	2.08

选择表 3 中 *IG* 最高的 kernel32. dll,进一步计算其不同参数在木马类恶意代码检测中的增益值,具体结果及相应解释如

表 4(具体版本环境 Windows 9X/Windows 2000/Windows XP/Windows 2003)所示。

表 4 Kernel32. dll 及其参数在检测木马类恶意代码中(100 个样本)的信息增益值

Kernel32. dll 函数	参数	*IG*/%	功能说明
OpenProcess //打开进程	dwDesiredAccess	12. 11	需要的进程权限
	bInheritHandle	6. 01	是否可以继承
VirtualAllocEx //分配空间	hProcess	14. 21	进程句柄
	lpAddress	4. 55	地址指针
	flProtect	4. 55	页面保护属性
	dwSize	4. 55	大小
WriteProcessMemory //写内存	hProcess	11. 78	进程句柄
	lpBaseAddress	4. 31	地址
	lpBuffer	4. 31	要写入的内容,字节
	nSize	4. 31	大小
	lpNumberOfBytesWritten	4. 31	传址,实际写入的大小
CreateRemoteThread //创建远程进程	hProcess	17. 31	进程句柄
	lpThreadAttributes	15. 01	线程安全属性
	dwStackSize	7. 03	堆栈大小
	lpStartAddress	7. 03	过程地址
	lpParameter	7. 03	参数
	dwCreationFlag	7. 03	*s* 建立标志
	lpThreadId	7. 03	线程表示符

利用特征选取算法将特征抽取出来。把每个文件中存在或不存在的特征都检查一遍,程序的向量空间如表 5 所示。创建的程序向量空间集被用做分类器和模型建立阶段的输入。

表 5 程序向量空间

分类	API1	API2	API3	…
Logfile 1	1	1	1	…
Logfile 2	1	0	1	…
Logfile 3	0	1	0	…

2.5 分类器

通过上面特征的选取,所生成的特征向量空间被转换为 WEKA'S ARFF 的格式,然后输入分类器进行分类。使用 WEKA 3.6 进行数据挖掘处理。对比采用 TP、FP、J48 分类器,分类过程中采用多层交叉验证来防止过度匹配。在这些数据处理过程中,数据集被随机划分为 10 个相等的部分,每个分类器都运行 10 次,将数据的九个部分都用来训练数据,建立模型,剩余的那一部分数据用做测试数据,计算每次运行的准确率。最后进行评估,评估的因素包括特征数、平均检出时间、总体消耗时间、准确率和方差等[15,16]。下面给出实验结果和分析。

3 评价

在接下来的实验中,比较在 2.2 节所讨论的三种特征集的分类结果。同样,创建了三个数据集,收集了选择的恶意的和良性的程序,每条数据集总共有 1 200 条记录,这些数据集有相同的记录,只在特征集上有所不同。然后通过特征筛选过程减少特征数量。对收集到的数据进行了三组实验。在第一个实验中,使用 API 名称的数据集 APIString。在第二个实验中,使用的是基于参数的存储地址的数据集 PString。第三个实验中,APString 是通过函数名和参数的顺序创建的,也就是两者的结合。在实验中,恶性的程序文件被标记为显性,正常的程序被标记为隐性。

使用多分类器来训练测试数据集,由此选出最好的分类方法。由于本文数据是平衡分布的,最终发现 J48 分类器比其他方法要好,所以下面的评价实验中的数据均由 J48 分类所得[17],这样具有更好的可比性。所有的实验都是在 2.0 GHz, Intel™ I7 芯片的内存 4 GB,Windows XP 操作系统下进行。

表 6 是在 $IG \geq 2.0\%$ 时,分别依据 APIString、PString、APString 三种检测方法对恶意代码的检测结果。从表中可以看出,采用 API 与其参数相结合的检测方法,检测时间大幅节省,而检出准确率却有明显提高,享有最好的方差,说明其检出的稳定性更可靠。

表 6 三种方式的检测情况比较($IG \geq 2.0\%$,采用 J48 分类器)

分类	APIString	PString	APString
特征数	2 488	2 115	165
平均检出时间/s	12	23	31
耗时/h	7. 3	11. 2	1. 04
准确率/%	86. 2	67. 3	94. 5
方差	0. 23	0. 55	0. 16

为了进一步说明临界值对这三种检测方法的影响,本文的第二组实验设定临界值为 2.5%,实验样本保持其不变,三种检测结果如表 7 所示。

表 7 三种方式的检测情况比较($IG \geq 2.5\%$,采用 J48 分类器)

分类	APIString	PString	APString
特征数	1 887	1 834	153
平均检出时间/s	14	25	35
耗时/h	6. 2	10. 1	1. 01
准确率/%	81. 2	58. 3	91. 2
方差	0. 24	0. 58	0. 16

从表 6、7 可以看出,当提高特征的信息增益值的临界值时,检测的时间虽然相应减少,但并不是线性于减少的特征数目,准确率也有一些降低,所以并不是临界值越高越好。而降低或取消临界值则明显增加了分类和检测的时间。

4 结束语

本文提出了一种基于 API 及其参数相结合的行为动态分析恶意软件的方法,通过引入信息增益值来选择 API 及其参数特征,较好地解决了特征的选取方法。通过实时选取这三个特征,来研究它们在识别恶意软件和良性软件之间的有效性。这些分类包括 API 名字、输入参数和以上两种特征集的结合。因为以往方法获取所有的 API,数据量很大,增加了分析时间,并可能导致内存过载,而本文提出只使用 API 的一小部分集合和它们相关的特征来分类。特征选取技术也用于减少特征的数量。相关的实验用来验证这些特征选取方法的有效性。获得的结论表明了上述辨识恶意软件方法的良好表现。

面对不断增加的庞大 API 库,将来的工作,需要拓展数据集,进一步寻找 API 与其相应参数对特定恶意代码的信息增益值,寻找合适的临界值,并增加模型来辨识新的恶意软件。本文提出了一个对软件行为刻画、检测的新思路。

致谢 感谢中国国家电力科学研究院《基于公网通信的电力可信计算平台应用技术研究》项目支持,华中科技大学信息安全实验室 2011 届同学们的有益探讨和对选取合适的 API 集的建议。

参考文献:

[1] SEIFERT C, WELCH I, KOMISARCZUK P. Identification of malicious Web pages with static heuristics[C]//Proc of Australasian Telecommunication Networks and Applications Conference. 2008:91-96.

[2] RICHARDSON R. 12th annual edition of the CSI computer crime and security survey[R]. [S. l.]: Computer Security Institution, 2008.

[3] McAfee threats report: fourth quarter 2010[R]. [S. l.]: McAfee Labs, 2011.

(下转第 3425 页)

佳功率分配关系:当功率小于 11 时干扰功率占 30% 最优;当中继总功率大于 11 小于 61 时干扰功率占 20% 最优;当功率大于 61 时干扰功率占 10% 最优。上述变化说明功率越大,干扰功率占的比例越小,则保密速率越大。但是当干扰功率减小为零($u=0$)时,即传统的译码转发并波束成型的方法,其性能远远落后于有干扰时的性能。

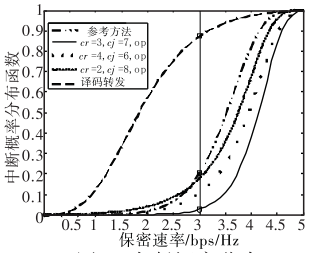

图 5　中断概率分布

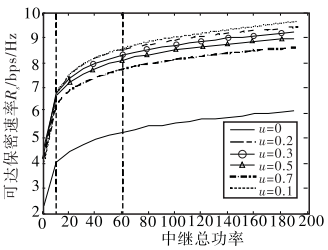

图 6　中继总功率受限下的功率分配与保密速率关系

4　结束语

为了克服传统方法中第二跳转发信息不安全的问题,本文提出一种基于协作波束成型的中继分组协作安全方案。在对中继进行分组的前提下加入协同干扰组,利用协同干扰组与协同中继组的相互协作,提高系统保密速率。此外对功率进行优化分配,初步探究最大保密速率与功率分配之间的关系。仿真结果进一步验证了该方法的正确性与高效性。此外,在不同功率下如何分组保密速率性能最大还需进一步研究,下一步工作为探究在大数量 relays 的情况下分组数量与功率分配对保密速率的影响的定量关系;推导不同功率下不同分组情况中最大保密速率与最优干扰功率比例的闭式解等。

参考文献:

[1] WYNER A D. The wire-tap channel[J]. Bell System Technical Journal,1975,54(8):1355-1387.

[2] LAI L,GAMAL H E. The relay-eavesdropper channel:cooperation for secrecy[J]. IEEE Trans on Information Theory,2008,54(9):4005-4019.

[3] KIM J,IKHLEF A,SCHOBER R. Combined relay selection and cooperative beamforming for physical layer security[J]. Journal of Communications and Networks,2012,14(4):364-373.

[4] HUANG Jing,SWINDLEHURST A L. Cooperative jamming for secure communications in MIMO relay networks[J]. IEEE Trans on Signal Processing,2011,59(10):4871-4883.

[5] ZHU Han,MARINA N,DEBBAH M. Physical layer security game:interaction between source, eavesdropper and friendly jammer[J]. Journal on Wireless Communications and Networking,2009,31(6):850-854.

[6] 李翔宇,金梁,黄开枝. 基于联合信道特征的中继物理层安全传输机制[J]. 计算机学报,2012,12(3):123-127.

[7] HUANG Jing,SWINDLEHURST A L. Secure communications via cooperative jamming in two-hop relay systems[C]//Proc of IEEE Conference on Communications Society. 2010:524-528.

[8] LIAO Wei-chuan, CHANG Tong-han, MA W K, *et al*. QoS-based transmit beamforming in the presence of eavesdroppers:an optimized artificial-noise-aided approach[J]. IEEE Trans on Signal Processing,2011,59(3):453-459.

[9] ZHOU Xiang-yun, McKAY M R. Secure transmission with artificial noise over fading channels:achievable rate and optimal power allocation[J]. IEEE Trans on Vehicular Technology,2010,59(8):1221-1228.

[10] DONG L,HAN Z,PETROPULU A P,*et al*. Improving wireless physical layer security via cooperating relays[J]. IEEE Trans on Signal Processing,2010,58(3):1875-1888.

[11] EKREM E,ULUKUS S. Secrecy in cooperative relay broadcast channels[J]. IEEE Trans on Information Theory,2011,57(1):137-155.

[12] WANG C L. A new cooperative transmission strategy for physical-layer security with multiple eavesdroppers[C]//Proc of the 75th IEEE Vehicular Technology Conference. 2012:155-159.

[13] CHEN Jing-chao, ZHANG Rong-qing, SONG Ling-yang, *et al*. Joint relay and jammer selection for secure two-way relay networks[J]. IEEE Trans on Information Forensics and Security,2012,7(1):310-320.

[14] ZHANG Jun-wei, GURSOY M C. Relay beamforming strategies for physical layer security[C]//Proc of IEEE Conference on Information Sciences and Systems. 2010:1321-1326.

(上接第 3410 页)

[4] HAN Lan-sheng,FU Cai,ZOU De-qing,*et al*. Task-based behavior detection of illegal codes[J]. Mathematical and Computer Modelling,2012,55(1):80-86.

[5] WANG C,PANG J M,ZHAO R C,*et al*. Malware detection based on suspicious behavior identification[C]//Proc of the 1st International Workshop on Education Technology and Computer Science. 2009:198-202.

[6] TIAN R,BATTEN L M,ISLAM R,*et al*. Differentiating malware from cleanware using behavioural analysis[C]//Proc of the 5th IEEE International Conference on Malicious and Unwanted Software. 2010:23-30.

[7] RIECK K,LASKOV P. Linear-time computation of similarity measures for sequential data[J]. Journal of Machine Learning Research,2008,9(6/1):23-48.

[8] AHMED F,HAMEED H,SHAFIQ M Z,*et al*. Using spatio-temporal information in API calls with machine learning algorithms for malware detection[C]//Proc of the 2nd ACM Workshop on Security and Artificial Intelligence. New York:ACM Press,2009:55-62.

[9] SZOR P. The art of computer virus research and defense[M]. [S. l.]:Addison-Wesley Professional,2005.

[10] YASON V M. The art of unpacking[EB/OL]. http://www.blackhat.com/presentations/bh-usa:07/yason/whitepaper/bh-usa-07-yason-wp.pdf.

[11] HU X. Large-scale malware analysis,detection,and signature generation[D]. Michigan:University of Michigan, 2011.

[12] MOSER A,KRUEGEL C,KIRDA E. Limits of static analysis for malware detection[C]//Proc of Annual Computer Security Application Conference. 2007:421-430.

[13] CHENG J,KAMOI K,WATANABE Y. User identification by signature code for noisy multiple-access adder channel[C]//Proc of IEEE International Symposium on Information Theory. 2006:1974-1977.

[14] SAMI A,RAHIMI H,YADEGARI B,*et al*. Malware detection based on mining API calls[C]//Proc of ACM Symposium on Applied Computing-Data Mining Track. 2010:1020-1025.

[15] TIAN R,BATTEN L,ISLAM R,*et al*. An automated classification system based on the strings of trojan and virus families[C]//Proc of the 4th International Conference on Malicious and Unwanted Software. 2009:23-30.

[16] KARBALAIE F,SAMI A,AHMADI M. Semantic malware detection by deploying graph mining[J]. International Journal of Computer Science Issues,2012,9(1):373-379.

[17] MOSKOVITCH R,STOPEL D,FEHER C,*et al*. Unknown malcode detection via text categorization and the imbalance problem[C]//Proc of IEEE International Conference on Intelligence and Security Informations. 2008:156-161.