ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.10, October 2010, pp.2599−2609
doi: 10.3724/SP.J.1001.2010.03653

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于彩色编码的多态蠕虫特征自动提取方法*

汪 洁, 王建新+, 陈建二

(中南大学 信息科学与工程学院,湖南 长沙 410083)

Automated Signature Generation Approach for Polymorphic Worm Based on Color Coding

WANG Jie, WANG Jian-Xin+, CHEN Jian-Er

(School of Information Science and Engineering, Central South University, Changsha 410083, China)

+ Corresponding author: E-mail: jxwang@mail.csu.edu.cn

Wang J, Wang JX, Chen JE. Automated signature generation approach for polymorphic worm based on color coding. *Journal of Software*, 2010,21(10):2599−2609. http://www.jos.org.cn/1000-9825/3653.htm

Abstract: A fast and accurate generation of worm signatures is essential in efficiently defending worm propagation. Most of the recent signature generation approaches do not generate accurate signatures for polymorphic worms in environments with noise. In this paper, a CCSF (color coding signature finding) algorithm is presented to solve the problem of a polymorphic worm signature generation with noise by using color coding. In the CCSF algorithm, *n* sequences are divided into *m* group, and signatures for every group sequence are generated by color coding. After filtering all signatures, an accurate worm signature is generated. CCSF's range of polymorphic worms is evaluated. When comparing CCSF with other existing approaches, CCSF shows a distinct advantages in generating accurate signatures for polymorphic worms in the presence of noise. Signatures generated do not contain fragments and can be used conveniently to detect polymorphic worms in IDS (intrusion detection system).

Key words: signatures generation; polymorphic worm; color coding; worm detection; intrusion detection

摘 要: 快速而准确地提取蠕虫特征对于有效防御多态蠕虫的传播至关重要,但是目前的特征产生方法在噪音干扰下无法产生正确的蠕虫特征.提出基于彩色编码的特征自动提取算法 CCSF(color coding signature finding)来解决有噪音干扰情况下的多态蠕虫特征提取问题.CCSF 算法将可疑池中的 *n* 条序列分成 *m* 组,然后运用彩色编码对每组序列进行特征提取.通过对每组提取出来的特征集合进行过滤筛选,最终产生正确的蠕虫特征.采用多类蠕虫对 CCSF 算法进行测试,并与其他蠕虫特征提取方法进行比较,结果表明,CCSF 算法能够在有噪音干扰的条件下准确地提取出多态蠕虫的特征,该特征不包含碎片,易于应用到 IDS(intrusion detection system)中对多态蠕虫进行检测.

关键词: 特征提取;多态蠕虫;彩色编码;蠕虫检测;入侵检测

中图法分类号: TP309 **文献标识码**: A

由于"零天"等多态变形蠕虫给 Internet 带来了巨大危害,蠕虫的检测和防御成为了一个重要问题[1],出现了

* Supported by the National Natural Science Foundation of China under Grant Nos.60673164, 60773111 (国家自然科学基金); the National Basic Research Program of China under Grant No.2008CB317107 (国家重点基础研究发展计划(973)); the Program for Changjiang Scholars and Innovative Research Team in University of China under Grant No.IRT0661 (长江学者和创新团队发展计划)

Received 2008-09-18; Accepted 2009-04-27

越来越多的针对多态蠕虫的检测方法和检测系统,如基于蠕虫特征的检测系统[2−4]和 Lee 等人[5]提出的 PolyI-D 系统.PolyI-D 系统是在蠕虫代码指令分布的基础上进行一些检查来检测多态蠕虫.本文仅考虑基于特征的多态蠕虫检测方法.蠕虫的检测能力主要依赖于蠕虫特征的质量和数量.攻击特征的提取最初是由人工进行的,然而,人们对于蠕虫的响应速度滞后于蠕虫的传播速度,对蠕虫攻击的防御都是在其对网络造成了较大危害之后才开始进行防御.因此,特征的自动提取逐渐成为研究者们越来越关注的问题[6].类似于“零天”这样能进行自我繁殖、形态多变的多态变形蠕虫的出现和快速传播,对特征自动提取技术提出了更高的要求.目前,人们针对多态蠕虫特征自动提取提出了许多解决方法,其中有影响力的方法包括以下几类:

(1) 最大公共子串(longest common substring,简称 LCS)[6]特征提取,如 Cai 等人设计的 WormShield 系统[2]、Portokalidis 等人设计的 SweetBait 系统[3]和 Ranjan 等人设计的 DoWicher 系统[4]等.这些系统都是采用基于 LCS 的方法来提取蠕虫特征,但是该方法要求进行特征提取的序列必须是无噪音的.而实际上,在特征提取时可疑池中均存在一定的噪音序列,所以,简单地提取最大公共子串无法获得正确的蠕虫特征.

(2) 基于多个连续字符串的特征(token)提取方法,如 Newsome 等人设计的 Polygraph 系统[7]、Li 等人设计的 Hamsa 系统[8]、Cavallaro 等人[9]设计的特征自动产生器 LISABETH 和 Mohammed MMZE 等人[10]设计的双 honeynet 系统等.Polygraph 系统[7]首先从可疑流量池中提取许多 token,这里的 token 是指在可疑池 n 个序列样本中出现至少 K 次的独立子串,通过对这些 token 进行组合,以及采用朴素贝叶斯方法来产生 3 种类型的简单特征,然后利用层次聚类方法来产生复杂特征.然而,采用层次聚类的方法在有噪音干扰的情况下会导致提取出错误的特征,Li[8]对此作了专门的论述.而且 Perdisci[11]提出了一种攻击方法,通过在可疑池中注入噪音来误导 Polygraph 产生错误的特征.Hamsa 系统是在 Polygraph 的基础上将 token 出现的次数作为了特征的一部分,通过得分函数 $score(COVs,FPs)$来确定 tokens 的集合作为特征对多态蠕虫进行检测.Hamsa 系统在特征产生的过程中虽然考虑了噪音干扰问题,但是它提取出来的特征比较复杂,将 token 在可疑池中出现的次数作为特征的一部分,这样对于 token 次数的严格限定会使得蠕虫在形态改变后可能不匹配特征.Cavallaro 等人设计的特征自动产生器 LISABETH[9]是基于蠕虫负载内容的,它所提取的特征与 Hamsa 的不同是,特征集合包含了所有出现在可疑池中的 token,所以比 Hamsa 抗攻击能力要强.然而,LISABETH 没有考虑可疑池中包含噪音的情况,所以在其所提取出来的所有 tokens 中,可能某些 tokens 是不正确的.Mohammed MMZE 等人[10]设计了一个双 honeynet 系统来捕获多态蠕虫流量,并提取在所有序列中至少出现 K 次的 tokens 作为蠕虫的特征.它的特征提取方法与 Polygraph[7]的 token 序列特征提取方法一致,当可疑池中出现噪音时,无法提取出正确的特征.

(3) 基于多序列联配的蠕虫特征提取,如唐勇等人[12]提出的基于多序列联配的攻击特征自动提取方法.该方法包括奖励相邻匹配的全局联配算法 CMENW(contiguous matches encouraging needleman-wunsch)和层次式多序列联配算法 HMSA(hierarchical multi-sequence alignment).CMENW 算法不会产生碎片,能够尽量保留连续的特征片段.但是,这种算法不适合多态蠕虫的特征提取,因为多态蠕虫特征出现的顺序是可变的,而 CMENW 产生的特征片段是连续的.HMSA 算法采用层次式序列联配,与 Polygraph 一样,在有噪音干扰的情况下会导致无法提取正确的特征[8].

(4) 基于 Expectation-Maximization(EM)和 Gibbs 采样的位置权重分布特征(position-aware distribution signature,简称 PADS)提取[13].PADS 架起了基于特征的检测方法和基于异常的检测方法之间的桥梁,然而在有噪音存在的情况下,PADS 同样无法得到正常的蠕虫特征[11].

上述蠕虫特征提取方法都解决了在无噪音情况下蠕虫特征的自动提取问题,但只有少数算法考虑了可疑池中存在噪音的情况,而且它们在特征的提取过程中都存在一些不足,无法在有噪音干扰的情况下提取出正确的蠕虫特征.因此,本文提出基于彩色编码的多态蠕虫特征自动提取方法 CCSF(color coding signature finding)来解决有噪音情况下的特征自动提取问题.CCSF 算法将包含多态蠕虫的网络数据流表示为序列,将序列分成 m 组,每组包含 20 条序列,然后运用彩色编码对 20 条序列进行特征提取,通过对每组提取出来的特征集合进行过滤筛选,最终提取出正确的蠕虫特征,有效地解决了有噪音干扰情况下的多态蠕虫特征提取问题.

CCSF 算法在存在噪音序列的情况下能够准确地提取出蠕虫特征,并且有效地减小了搜索空间,提高了执

行效率.对算法实验测试结果表明,CCSF 算法能够提取出蠕虫序列中的所有特征子序列,可以应用于实际的多态蠕虫病毒检测和防御中.

本文第 1 节首先简单介绍彩色编码,然后详细描述基于彩色编码的特征自动提取算法的实现过程.第 2 节给出实验结果并进行分析和讨论,验证本文给出的特征提取算法的正确性和在有噪音情况下的有效性.第 3 节对本文进行总结.

1 CCSF 算法

CCSF 算法的目标是在包含 n 条序列、其中存在 k 条蠕虫序列的可疑池中提取出蠕虫特征.由于无法确定哪 k 条序列是蠕虫序列,所以为了提取蠕虫特征,需要对 n 条序列中的每个包含 k 条序列的组合进行特征提取,即需要 C_n^k 次对 k 条序列进行特征提取.但是,当可疑池中序列数目较大时,例如当可疑池中包含 2 000 条序列,即 n=2000,k>1000 时,C_n^k 的数目太大,使得 C_n^k 次进行特征提取不可行.本文采用分治的方法,将 n 条序列分成 m 组,每组包含 $g=\left\lfloor \frac{n}{m} \right\rfloor$ 条序列,m 组序列中必然有一组序列至少存在 $u=\left\lfloor \frac{k}{m} \right\rfloor$ 条蠕虫序列.所以,当对所有 g 条序列中的每个包含 u 条序列的组合进行特征提取时,至少存在一个 u 条序列的组合能够提取出正确的蠕虫特征,即进行 mC_g^u 次特征提取,其中至少有一次能够提取出正确的蠕虫特征.这里,当 g 取值过大时,mC_g^u 中的 C_g^u 数目仍然很大;然而,如果 g 取值过小,由于多态蠕虫具有多种蠕虫形态,又会使得从 g 中 u 条序列提取出来的特征无法具有一般性.从以上分析可以看出,g 取值越大,特征提取就越准确,而计算量也会越大.例如,在 g=20 时,$C_{20}^{15}=15504$, $C_{20}^{14}=38760$, $C_{20}^{13}=77520$, $C_{20}^{12}=125970$, $C_{20}^{11}=167960$.从这些数据可以看出,当 g=20 时,利用组合进行特征提取的计算量已经很大了,不适合在实际中应用.结合彩色编码的特点,本文将进一步讨论在 g=20 时如何利用彩色编码方案来降低特征提取次数 C_g^u.

1.1 彩色编码

彩色编码是由 Alon 等人[14]提出来的,是通过散列公式 f 将集合 Q={1,…,K}完全映射到集合 W={1,…,K}的方案,使得对于 W 中的每个元素 x 都能映射到 Q 的不同 $f(x)$上,称 f 为 W 的完全散列公式.若通过 f 对 Q 中每个元素着色,那么通过 f 映射后,W 中没有相同颜色的元素.Chen 等人[15]提出采用彩色编码可求解 k-PATH,LOG PATH 等 NP 难的问题.目前,这一技术已被应用在基序发现[16]等生物计算问题中.本文将彩色编码应用于蠕虫特征的提取问题.

彩色编码的基本原则是,采用尽可能少的着色数覆盖所有(g,u)组合.本文采用一种基于划分思想的着色方案[17]来对 g 个节点进行 u 着色.给定一个元素集合 U={e_1,…,e_g}和一个颜色集合 C={c_1,…,c_u},构造一个 g 元组 H=〈h_1,…,h_g〉,h_i=f(e_i).$f(e)$表示用一个颜色对 e 进行着色,$\forall i, h_i \in C$.这样,使得 U 中的每一个元素对应一种颜色,而 C 中的每个颜色被使用至少一次,那么 g 元组 H 称为(g,u)-*coloring* 着色.如果某种颜色 $c_i \in C$ 在 H 中只使用一次,则称该颜色为专有色,否则称为复合色.如果一个(g,u)-*coloring* 着色集合满足:对于任意(g,u)组合,至少存在一个(g,u)-*coloring* 着色覆盖该(g,u)组合,则称该着色集合为一个(g,u)-*coloring* 着色方案,表示为 *Coloring*(g,u).

在一次(g,u)-*coloring* 着色后,g 个节点中有 $g-u$ 个节点着复合色,这 $g-u$ 个节点正好使用 $g-u$ 种颜色.在这样的前提下,通过枚举 $g-u$ 个复合色的分布情况来产生 *Coloring*(g,u).首先,将 g 个节点划分为 g/2 个集合,每个集合包含两个节点;然后,枚举 g/2 集合中 $g-u$ 个着复合色节点的分布情况.例如,当 g=20,u=19 时,有一个节点着复合色,将 20 个节点划分为 10 个集合,则着复合色的节点在 10 个集合的位置有 10 种情况,所以着色数为 10.当 g=20,u=18 时,有两个节点着复合色,同样可以通过枚举这两个着复合色节点在所有集合中的位置来确定所有的着色数.当 u<18 时,可以通过递归到较小规模来解决,即对节点集合作进一步划分.

Coloring(20,16)着色方案已经应用于生物信息学的 motif 查找问题[16],本文计算了 *Coloring*(20,u)(u=11,12,…,19)着色方案,将其应用于特征提取问题.表 1 列出了当 g=20,u 取不同值时着色方案的着色数,并与不采用着色方案的 C_g^u 组合数进行了比较.

Table 1 Comparison between the size of *Coloring*(20,*u*) and the number of *u*-combinations of 20

表 1 比较 *Coloring*(20,*u*)着色数与(20,*u*)组合数

	Coloring(20,*u*)	*u*-Combinations of 20
u=19	10	20
u=18	50	190
u=17	170	1 140
u=16	403	4 845
u=15	862	15 504
u=14	1 220	38 760
u=13	2 036	77 520
u=12	2 085	125 970
u=11	3 250	167 960

从表 1 可以看出,(*g*,*u*)-*coloring* 着色数目远远小于(*g*,*u*)组合数 C_g^u,我们将 C_n^k 次从 *n* 条序列中提取特征的过程转换成 *m*×*Coloring*(20,*u*)次从 *g* 条序列提取特征的过程.下面我们讨论在 *g* 条序列中提取特征.

1.2 基于彩色编码的特征提取

本文首先讨论可疑池中仅包含蠕虫序列的特征提取问题.这里,设可疑池的规模为 *x*,即可疑池中包含 *x* 条序列,然后在此基础上讨论可疑池中同时包含蠕虫序列和噪音序列的特征提取问题.

我们用集合 *Signature*={tok_i}表示多态蠕虫的特征,其中,tok_i 为第 *i* 个字节序列,也称为第 *i* 个 token.如果一个序列包含了 *Signature* 中所有的 token,则认为这个序列匹配该蠕虫特征.Newsome[7]证明,这样的特征可以对多态蠕虫进行检测.

当可疑池中仅包含蠕虫序列时,设 *x* 条蠕虫序列为 S_x={y_1,…,y_x},算法 CTF(common token finding)是提取在所有序列中都出现的长度 *length* 在[*b*,*a*]之间的所有子序列,并将其放入集合 *Signature* 中.设序列 y_i 是所有 *x* 条蠕虫序列中最短的序列,其长度为 L_i,则 y_i 包含了 L_j–*length*+1 个子序列{V_j|*j*=1,2,…,L_i–*length*+1},如果 y_1,…,y_{i-1},y_{i+1},…,y_x 中每条序列都包含了 V_j,则 V_j 为一个 token.当找到一个 token 后,在每条序列中去用特殊字符替换该 token 中的所用字符,然后继续采用同样的方法在{y_1,y_2,…,y_x}中提取新的 token.在提取过程中,算法 CTF 首先提取所有长度为 *a* 的 token,然后提取所有长度为 *a*–1 的 token,直到提取所有长度为 *b* 的 token.算法 CTF 的具体描述如图 1 所示.

Algorithm *CTF*(*S*$_x$,*a*,*b*).
Input: *x* sequences S_x={y_1,…,y_x}, *a*, *b*;
Output: A set of token *Signature*={t_i} for the *x* sequences.
y_i is the sequence with the smallest length and L_i is its length; *length*=*a*;
While *length*>=*b*
{For *j*=1 to L_i–*length*+1
{Choosing V_j from y_i;
If V_j does not contain the special symbol and if all other sequences contain V_j
{*Signature*←*Signature*∪={V_i};
replace the characters of V_j with special symbols in all sequences;
}}
length=*length*−1;}
Return *Signature*

Fig.1 Description of algorithm CTF

图 1 算法 CTF 的描述

CTF 算法的时间复杂度为 *O*(*lmu*(*b*–*a*)),其中,*l* 为最短序列的长度,*m* 为最长序列的长度,*u* 为序列的总数,*b* 和 *a* 分别是算法中的参数.

Tang[13]指出 10 字节长的特征能够较好地检测蠕虫,本文设定 token 的长度为 3≤*length*≤15,即 CTF 算法中

a=15,b=3.

当可疑池中的噪音序列大于蠕虫序列时较难提取出蠕虫特征,所以我们考虑当序列数目为 g 的可疑池中存在 $u(u>g/2)$条蠕虫序列时,如何提取出蠕虫特征.首先将可疑池中的 g 条序列看作 g 个节点,通过前面介绍的彩色编码构造 *Coloring*(g,u)着色方案,构造过程用 Build *Coloring*(g,u)表示,*Coloring*(g,u)的着色数用 f 表示.在每次着色完成之后,将着同样颜色的序列合并成为同一条序列,这样我们将 g 条序列转化为了 u 条序列,然后采用 CTF 算法对 u 条序列进行特征提取.这一过程的描述如图 2 所示.

Algorithm *SGA*(S_g,u)
Input: g sequences $S_g=\{y_1,\ldots,y_g\}$, u;
Output: A set of Signature $Sig=\{Signature_i\}$ for g sequences.
Build *Coloring*(g,u);
For each (g,u)-*coloring* of *Coloring*(g,u)
{g sequences of S_g are colored with the (g,u)-*coloring*;
$S_u=\{x_1,\ldots,x_u\}$ is generated by merging sequences with the same color in S_g;
$Signature_i \leftarrow CCC(S_u,15,3)$;
If $Signature_i$ is not $\varnothing$
$Sig \leftarrow Sig \cup =\{Signature_i\}$;
}
Return *Sig*

Fig.2 Description of algorithm SGA
图 2 算法 SGA 的描述

算法 SGA(signature generation algorithm)选取 g=20,着色参数 u 的取值范围为 $11\leqslant u\leqslant 20$.算法的返回结果 *Sig* 是由特征 *Signature* 组成的集合,可能包含多个特征 *Signature*.由前面介绍的彩色编码基本原则可知,f 个(g,u)-*coloring* 着色覆盖了所有的 C_g^u 组合,所以 f 次在着色后产生的 u 条序列中提取的特征与 C_g^u 次在 g 条序列中选择 u 条序列提取的特征相同.

在 SGA 算法中,构造 *Coloring*(g,u)着色方案的时间复杂度为 $O(1)$,应用 *Coloring*(g,u)中的着色方案对 g 条序列进行着色并合并相同颜色的序列的时间复杂度为 $O(1)$,CTF 算法的时间复杂度为 $O(lmu(b-a))$,所以算法 SGA 的时间复杂度为 $O(f_u(1+lmu(b-a)))$.其中,f_u 是 *Coloring*(g,u)的着色数.

本节选取 u=15,采用 SGA 算法分别对 Apache-Knacker,ATPhttpd 和 BIND-TSIG 这 3 类蠕虫进行特征提取.实验分两组进行,均采用与 Polygraph[7]相同的多态蠕虫样本,第 1 组的噪音序列是随机产生的噪音,噪音序列之间无公共的 token,噪音序列与蠕虫样本序列之间也没有公共的 token;第 2 组的噪音序列是采用由 Polygraph[7]产生的噪音序列.每类蠕虫的可疑池中固定包含 20 个序列,其中,噪音序列的数目为 l,k'为可疑池中的蠕虫序列数目,$k'=20-l$.第 1 组测试的返回结果见表 2,第 2 组测试的返回结果见表 3.

从表 2 可以看出,当噪音序列与蠕虫序列之间不存在公共的子序列串,而且各条噪音序列之间也不存在公共的字符串时,SGA 算法返回的特征集合有以下两种情况:

(1) 当 $k'\geqslant u$ 时,算法返回的特征集合中仅包含蠕虫特征,而没有产生其他的干扰特征,见表 2,算法返回的特征集合中只包含了正确的 Apache-Knacker 蠕虫特征:
['*HTTP*/1.1\r\n','\r\n*Host*:','\r\n*Host*:','\xff\xbf','\r\n','\r\n','*GET*']

(2) 当 $k'<u$ 时,算法返回的特征集合为空.

从表 3 可以看出,当噪音序列与蠕虫序列之间、各噪音序列之间存在公共子序列串时,SGA 算法返回的特征集合有以下两种情况:

(1) 当 $k'\geqslant u$ 时,算法返回的特征集合中包含蠕虫特征,也包含蠕虫序列与噪音序列的公共子序列串集合.见表 3,SGA 算法产生的特征集合既包含了正确的 Apache-Knacker 蠕虫特征['*HTTP*/1.1\r\n', '\r\n*Host*:', '\r\n*Host*:','\xff\xbf','\r\n','\r\n','*GET*'],也包含了蠕虫序列与噪音序列的公共子序列串:

['*HTTP*/1.1\r\n','*Host*:','*GET*','\r\n','\r\n']

(2) 当 $k'<u$ 时,算法返回的特征集合包含了蠕虫序列和噪音序列所共有的子序列串集合和噪音序列与噪音序列之间所共有的子序列串集合.见表 3,SGA 算法产生了特征集合中包含的是错误的特征:

['*HTTP*/1.1\r\n','*Host*:','*GET*','\r\n','\r\n']

Table 2 Extracting signature from 20 sequences with random noise sequences

表 2 20 条序列的特征提取(可疑池包含随机产生的噪音序列)

Worm name	u	Signature	Generating probability
Apache-Knacker worm	$k'\geqslant u$	'*HTTP*/1.1\r\n', '\r\n*Host*:', '\r\n*Host*:', '\xff\xbf', '\r\n', '\r\n', '*GET*'	1
	$k'<u$	*Null*	0
ATPhttpd worm	$k'\geqslant u$	'*HTTP*/1.1\r\n', '\xff\xbf', '*GET*/'	1
	$k'<u$	*Null*	0
BIND-TSIG worm	$k'\geqslant u$	'\x00\x00\xfa', '\xff\xbf', '\x00'	1
	$k'<u$	*Null*	0

Table 3 Extracting signature from 20 sequences with noise sequences generated by Polygraph

表 3 20 条序列的特征提取(可疑池包含由 Polygraph 产生的噪音序列)

Worm name	u	Signature	Generating probability
Apache-Knacker worm	$k'\geqslant u$	'*HTTP*/1.1\r\n', '\r\n*Host*:', '\r\n*Host*:', '\xff\xbf', '\r\n', '\r\n', '*GET*'	1
		'*HTTP*/1.1\r\n', '*Host*:', '*GET*','\r\n', '\r\n'	
	$k'<u$	'*HTTP*/1.1\r\n', '*Host*:', '*GET*','\r\n', '\r\n'	0
ATPhttpd worm	$k'\geqslant u$	'*HTTP*/1.1\r\n', '\xff\xbf', '*GET*/'	1
		'*HTTP*/1.1\r\n', '*GET*/'	
	$k'<u$	'*HTTP*/1.1\r\n', '*GET*/'	0
BIND-TSIG worm	$k'\geqslant u$	'\x00\x00\xfa','\xff\xbf', '\x00'	1
		'\x00\x00\x', '\x00'	
	$k'<u$	'\x00\x00\x', '\x00'	0

综合表 2 和表 3,当 $k'\geqslant u$ 时,SGA 产生的特征集合中均包含了正确蠕虫特征,即提取蠕虫特征的概率为 1.当 $k'<u$ 时,由于受噪音的干扰,SGA 算法产生的特征集合中仅包含了错误的特征,提取蠕虫特征的概率为 0.

在 SGA 算法的基础上,我们进一步考虑可疑池中序列数目为 n 的蠕虫特征提取问题.

1.3 CCSF算法的过程

CCSF 算法的目标是从序列数目为 n 的可疑池中提取出蠕虫特征.首先,CCSF 算法按顺序将 n 条序列分成 20 条序列一组,最后一组如果不足 20 条,则随机抽取其他序列的副本补足 20 条.设 n 条序列分成了 $m=\left\lceil \frac{n}{20} \right\rceil$ 组,应用 SGA 算法,分别对每组序列进行特征提取,根据每组序列的返回结果设置权值 $w_1,w_2,\ldots,w_m$.每组序列的处理过程如下:

SGA 算法的参数初始选择 u=20,如果算法没有返回结果,则参数 $u=u-1$,继续执行该算法,直到有特征集合产生,或者参数 u<11.当算法有特征集合产生时,由前面的分析可知,在噪音对特征提取存在干扰时,SGA 算法返回的特征集合包含了蠕虫特征和其他错误的非蠕虫特征.所以在 SGA 算法产生特征集合 Sig_i 后,Sig_i 中的特征必须经过过滤池进行筛选,过滤池由大量的正常流序列组成.

特征筛选过程如下:

当特征中所有的 token 都在过滤池中出现时,判断该特征为非蠕虫特征;当特征中存在 token 未在正常流中出现时,则判断该特征为蠕虫特征.例如,当 SGA 算法对 ATPhttpd 蠕虫进行特征提取时,获得了如表 3 所示的两类特征:一类为集合{'*HTTP*/1.1\r\n','\xff\xbf','*GET*/'},另外一类为集合{'*HTTP*/1.1\r\n','*GET*/'}.在过滤池中进行筛选时,'*HTTP*/1.1\r\n'和'*GET*/'均在过滤池中出现,所以判断集合{'*HTTP*/1.1\r\n', '*GET*/'}为非蠕虫特征;而'\xff\xbf'未在过滤池中出现,所以判断集合{'*HTTP*/1.1\r\n','\xff\xbf','*GET*/'}为蠕虫特征.

在筛选完成之后,如果集合 Sig_i 中所有的特征都被过滤掉,则 SGA 算法的参数 $u=u-1$,重新对该组序列进行特征提取.直到 Sig_i 不为空,或者 SGA 算法的参数 $u<11$.当集合不为空时,$w_i=u$;否则,$w_i=10$.

这样,m 组序列将得到 m 个特征集合 $Sig_1,Sig_2,\ldots,Sig_m$ 和 m 个值 $w_1,w_2,\ldots,w_m$.当 $w_1+w_2+\ldots+w_m>n/2$ 时,认为 CCSF 算法提取出了蠕虫特征.然后对所有不为空的集合 Sig_i 中的特征进行公共 token 提取,提取出来的 token 集合作为该类蠕虫的特征.进行公共token提取的原因是由于多态蠕虫存在多态特性,一类蠕虫可能有多个不同形态的蠕虫样本,所以,对不同形态的蠕虫提取出的特征也会有一定的差异,即 $Sig_1,Sig_2,\ldots,Sig_m$ 中的特征可能有一定的差异,因此进行公共 token 提取后所产生的特征将更具一般性,能够检测具有不同形态的蠕虫.CCSF 算法的描述如图 3 所示.

Algorithm *CCSF*(*S*,*a*,*b*,*N*).
Input: n sequences $S=\{y_1,\ldots,y_n\}$, a, b, N is a filter pool for filtering wrong signature;
Output: Worm signature $Signature=\{tok_i\}$.
$S\rightarrow S_1,S_2,\ldots,S_m$;
$Sig=NULL$;
For i=1 to m;
{$u=a$; $Flag$=TRUE;
While ($Flag$==True);
{$Sig_i\leftarrow SGA(S_i,u)$;
For each $Signature$ in Sig_i
If the $Signature$ appears in N, delete it from Sig;
If $Sig_i\neq\varnothing$
{$w_i=u$; $Flag$=FALSE;}
Else {$u=u-1$;
If ($u<b$) Then {w_i=10; $Flag$=FALSE;}}
$Sig=Sig\cup\{Sig_i\}$;}
If $w_1+w_2+\ldots+w_m>n/2$ then
{Extracting the same tokens from all $Signature$ in Sig and put them into the set of tokens $\{tok_i\}$;
return ($\{tok_i\}$);}
Else return ("not finding signature");

Fig.3 Description of algorithm CCSF

图 3 算法 CCSF 的描述

在 CCSF 算法中,SGA 算法的时间复杂度为 $O(f_u(1+lmu(b-a)))$,其中,f_u 是 $Coloring(g,u)$的着色数.所以,算法 CCSF 的时间复杂度为 $O\left(\left\lceil\frac{n}{20}\right\rceil\sum_{i=11}^{20}f_i(1+lmi(b-a))\right)$,其中,$f_i$ 是 $Coloring(g,i)$的着色数.

由于 SGA 算法的着色参数 u 的取值为 $11\leqslant u\leqslant 20$,所以算法中设置 a=20,b=11.CCSF 算法将 n 条序列分成了 m 组 20 条序列.当 m 组 20 条序列中存在至少一组序列,该组序列中蠕虫的数目 $k'\geqslant u$ 时,由 SGA 算法分析可知,在特征产生过程中,能够以概率为 1 产生蠕虫特征.

2 算法测试

我们用 C++语言实现了 CCSF 算法,分别在有噪音环境下和无噪音环境下进行了测试.测试采用与 Polygraph[7]相同的测试用例,包括 Apache-Knacker 和 ATPhttpd 蠕虫,并对 CCSF 算法与 Polygraph[7]中 token 序列特征提取方法和层次式多序列联配方法 HMSA[12]进行了比较.

2.1 无噪音环境下的测试

首先分别产生了 200 条 Apache-Knacker 和 ATPhttpd 蠕虫序列,在无噪音情况下,采用 CCSF 算法进行特征提取,并将结果与 Polygraph 和 HMSA 产生的特征进行了比较.比较结果见表 4.

从表 4 可以看出,在可疑池中无噪音的情况下,CCSF 算法与其他蠕虫特征提取方法都可以得出正确的蠕虫

特征.接下来对 CCSF 算法的抗噪音能力进行测试.

Table 4 Signatures generated by different algorithms

表 4 各算法产生的特征

	Apache-Knacker signature	ATPHttpd signature
Polygraph	*'GET', 'HTTP/1.1\r\n', ':', '\r\nHost:', '\r\n', ':', '\r\nHost:', '\xff\xbf', '\r\n'*	*'GET/', '\xff\xbf', 'HTTP/1.1\r\n'*
HMSA	*GET~*~HTTP/1.1\r\n*:~*\r\nHost:~*\ r\n*:~*\r\nHost:~*\xFF\xBF??????????\r\n*	*'GET~/*\xff\xbf??????????[108 个]HTTP/1.1\r\n*
CCSF	*'HTTP/1.1\r\n', '\r\nHost:', '\r\nHost:', '\xff\xbf', '\r\n', '\r\n', 'GET'*	*'HTTP/1.1\r\n', '\xff\xbf', 'GET/'*

2.2 有噪音环境下的测试

本文选用 Apache-Knacker 和 ATPhttpd 蠕虫作为测试用例.可疑池中固定测试样本序列为 200 条,并不断地用噪音序列替代其中的蠕虫序列,分别测试 3 种算法返回正确结果的概率.由于噪音序列的存在,产生的特征不一定是正确的,所以我们采用以下两个概率来评估 3 种算法:(1) 返回正确特征的概率 P_1.在这里,正确特征是指在特征中包含了所有的蠕虫特征 token;(2) 返回精确特征的概率 P_2.精确特征是指在特征中仅包含所有的蠕虫特征 token.如特征{'\xff\xbf','\r\nHost:','\xb0\xib','GET','M','\r\n','HTTP/1.1\r\n'},它包含了所有的 Apache-Knacker 蠕虫特征 token:'\xff\xbf','\r\nHost:','GET','\r\n','HTTP/1.1\r\n',所以它被认为是正确的特征;但是由于其包含了碎片'\xb0\xib'和'M',所以它不是精确的特征.图 4 显示了 CCSF,Polygraph 和 HMSA 这 3 种算法返回正确特征的概率 P_1.图 5 显示了 3 种算法返回精确特征的概率 P_2.

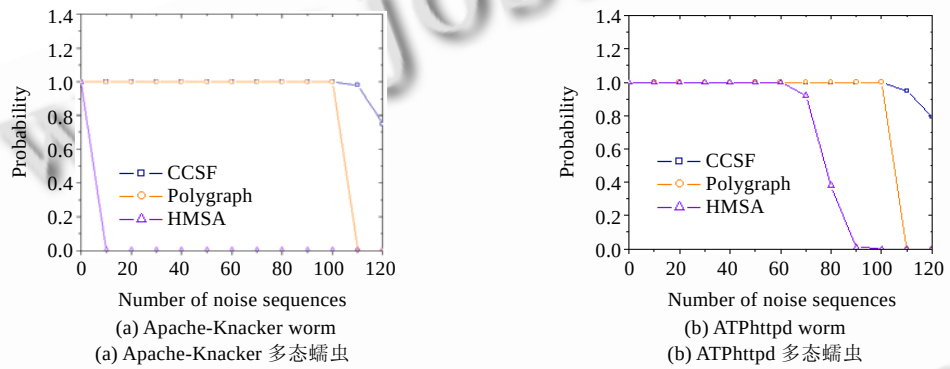

Fig.4 Probability of three algorithms returning accurate results (there are probably fragments in results)

图 4 3 种算法返回正确结果的概率(结果可能包含碎片)

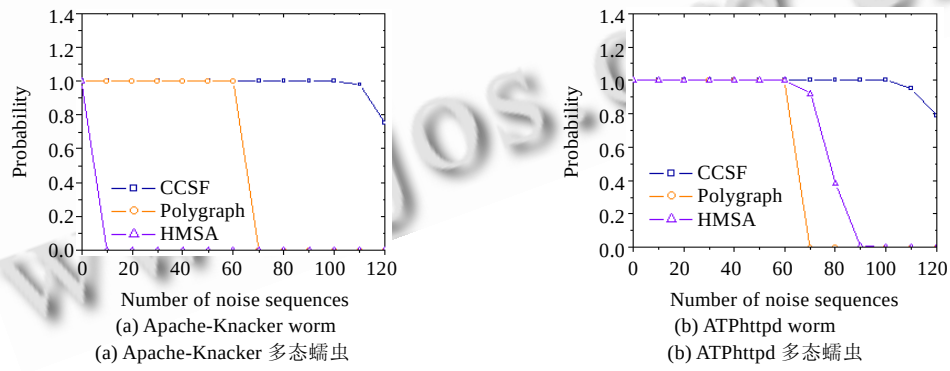

Fig.5 Probability of three algorithms returning accurate results (there are not fragments in results)

图 5 3 种算法返回正确结果的概率(结果不包含碎片)

如图 4 所示,CCSF 和 Polygraph 都能以较高的概率产生正确的蠕虫特征.当可疑池中噪音序列超过 10 条时,HMSA 不能产生正确的 Apache-Knacker 蠕虫特征.从图 4 可以看出,HMSA 产生正确 Apache-Knacker 蠕虫特征的概率低于产生 ATPhttpd 蠕虫特征的概率.这是因为 HMSA 层次聚类是在两两序列匹配的基础上进行的,当蠕虫序列和噪音序列之间的相似性较高的时候,HMSA 产生正确蠕虫特征的概率将会很低.从图 5 可以看出,当可疑池中噪音序列的概率高于 35%时,Polygraph 无法产生精确的蠕虫特征.Polygraph 提取 n 条序列中独立出现 k 次的子序列串作为特征.当噪音序列和部分蠕虫序列之间有公共的子序列串时,Polygraph 将会把这些子序列串作为特征的一部分提取出来,从而在最终产生的特征中将会包含碎片.CCSF 由于应用了彩色编码进行特征提取,因此能以较高的概率提取出精确的蠕虫特征.综合图 4 和图 5,CCSF 和 Polygraph 与 HMSA 相比,能够更好地产生精确的蠕虫特征.

本文进一步对 CCSF 算法与 Polygraph 所提取的特征进行了比较.我们采用 Apache-Knacker 和 ATPhttpd 蠕虫作为测试用例,分别产生了 2 000 个测试样本.固定总量 2 000 条序列不变,不断采用噪音序列替代其中的蠕虫序列,测试算法的返回结果.本文对噪音序列数目分别为 100,200,300,400,500,600,700,800,900 条时的返回结果与 Polygraph 蠕虫特征提取的返回结果进行了比较,比较结果见表 5.

Table 5 Signatures generated by CCSF and Polygraph comparison in the presence of noises

表 5 CCSF 算法与 Polygraph 方法在有噪音情况下产生的特征的对比

Number of noise sequences	Worm name	Polygraph	CCSF
l=100	Apache-Knacker worm	‘\xff\xbf’, ‘:’, ‘\r\nHost:’, ‘GET’, ‘Iu’, ‘\r\n’, ‘HTTP/1.1\r\n’	‘HTTP/1.1\r\n’, ‘\r\nHost:’, ‘\xff\xbf’, ‘\r\n’, ‘GET’
	ATPhttpd worm	‘\xff\xbf’, ‘O\xde’, ‘F\x18’, ‘HTTP/1.1\r\n’, ‘GET /’, ‘\xba\xc3’	‘HTTP/1.1\r\n’, ‘\xff\xbf’, ‘GET/’
l=200	Apache-Knacker worm	‘\xff\xbf’, ‘:’, ‘\r\nHost:’, ‘\xb0\xib’, ‘GET’, ‘M’, ‘\r\n’, ‘HTTP/1.1\r\n’	‘HTTP/1.1\r\n’, ‘\r\nHost:’, ‘\xff\xbf’, ‘\r\n’, ‘GET’
	ATPhttpd worm	‘\xff\xbf’, ‘%\x80’, ‘L%’, ‘HTTP/1.1\r\n’, ‘GET/’	‘HTTP/1.1\r\n’, ‘\xff\xbf’, ‘GET/’
l=300	Apache-Knacker worm	‘\xff\xbf’, ‘:’, ‘\r\nHost:’, ‘Z’, ‘GET’, ‘9’, ‘\r\n’, ‘A\xf4’, ‘HTTP/1.1\r\n’	‘HTTP/1.1\r\n’, ‘\r\nHost:’, ‘\xff\xbf’, ‘\r\n’, ‘GET’
	ATPhttpd worm	‘\xff\xbf’, ‘HTTP/1.1\r\n’, ‘GET/’	‘HTTP/1.1\r\n’, ‘\xff\xbf’, ‘GET/’
l=400	Apache-Knacker worm	‘\xff\xbf’, ‘:’, ‘\r\nHost:’, ‘GET’, ‘\r\n’, ‘HTTP/1.1\r\n’	‘HTTP/1.1\r\n’, ‘\r\nHost:’, ‘\xff\xbf’, ‘\r\n’, ‘GET’
	ATPhttpd worm	‘\xff\xbf’, ‘\xcb)’, ‘HTTP/1.1\r\n’, ‘GET/’	‘HTTP/1.1\r\n’, ‘\xff\xbf’, ‘GET/’
l=500	Apache-Knacker worm	‘\xff\xbf’, ‘:’, ‘\r\nHost:’, ‘Z’, ‘1g’, ‘GET’, ‘\x02\xd7’, ‘6\r\nHost:’, ‘\r\n’, ‘HTTP/1.1\r\n’	‘HTTP/1.1\r\n’, ‘\r\nHost:’, ‘\xff\xbf’, ‘\r\n’, ‘GET’
	ATPhttpd worm	‘\xff\xbf’, ‘\xcfS’, ‘HTTP/1.1\r\n’, ‘GET/’, ‘,\xcf’	‘HTTP/1.1\r\n’, ‘\xff\xbf’, ‘GET/’
l=600	Apache-Knacker worm	‘\xff\xbf’, ‘ze’, ‘:’, ‘>.’, ‘\r\nHost:’, ‘3n’, ‘GET’, ‘\r\n’, ‘HTTP/1.1\r\n’	‘HTTP/1.1\r\n’, ‘\r\nHost:’, ‘\xff\xbf’, ‘\r\n’, ‘GET’
	ATPhttpd worm	‘\xff\xbf’, ‘\xbf\x94’, ‘HTTP/1.1\r\n’, ‘GET/’	‘HTTP/1.1\r\n’, ‘\xff\xbf’, ‘GET/’
l=700	Apache-Knacker worm	‘\xff\xbf’, ‘:’, ‘\r\nHost:’, ‘Y\xec’, ‘GET’, ‘\r\n’, ‘HTTP/1.1\r\n’	‘HTTP/1.1\r\n’, ‘\r\nHost:’, ‘\xff\xbf’, ‘\r\n’, ‘GET’
	ATPhttpd worm	‘\xff\xbf’, ‘HTTP/1.1\r\n’, ‘GET/’, ‘\xe0\xf9’	‘HTTP/1.1\r\n’, ‘\xff\xbf’, ‘GET/’
l=800	Apache-Knacker worm	‘\xff\xbf’, ‘HY’, ‘:’, ‘\x8a\x93’, ‘\r\nHost:’, ‘F’, ‘\xdf\xbe’, ‘GET’, ‘d\r\nHost:’, ‘\xf1\xf1’, ‘\r\n’, ‘HTTP/1.1\r\n’	‘HTTP/1.1\r\n’, ‘\r\nHost:’, ‘\xff\xbf’, ‘\r\n’, ‘GET’
	ATPhttpd worm	‘\xff\xbf’, ‘HTTP/1.1\r\n’, ‘GET/’, ‘\x82\x17’	‘HTTP/1.1\r\n’, ‘\xff\xbf’, ‘GET/’
l=900	Apache-Knacker worm	‘\xff\xbf’, ‘:’, ‘\r\nHost:’, ‘Wg’, ‘GET’, ‘\r\n’, ‘HTTP/1.1\r\n’	‘HTTP/1.1\r\n’, ‘\r\nHost:’, ‘\xff\xbf’, ‘\r\n’, ‘GET’
	ATPhttpd worm	‘\xff\xbf’, ‘HTTP/1.1\r\n’, ‘GET/’, ‘\xba\xf7’, ‘\x94\x04’	‘HTTP/1.1\r\n’, ‘\xff\xbf’, ‘GET/’

从表 5 可以看出,CCSF 算法产生了正确的蠕虫特征,而 Polygraph 也产生了正确的蠕虫子序列特征.但是由

于受噪音的影响,Polygraph 还产生了多余的 token 子序列,如在提取 Apache-Knacker 蠕虫特征时,在可疑池中包含了 200 条噪音序列的情况下,Polygraph 除了产生蠕虫子序列特征外,还产生了':','\xb0\xib'和'M'子序列,而 CCSF 则成功地提取了所有 token 作为蠕虫的特征.因此,CCSF 产生的特征比 Polygraph 产生的特征更精确.

3 结 论

为了解决有噪音干扰情况下多态蠕虫的特征提取问题,本文提出了基于彩色编码的多态蠕虫特征自动提取算法(CCSF).CCSF 算法首先将 n 条蠕虫序列分成多组 20 序列,并运用彩色编码对 20 序列进行特征提取,在有效解决有噪音干扰的蠕虫特征提取问题的同时,极大地降低了问题的复杂性.当可疑池中噪音序列数目少于蠕虫序列数目时,CCSF 能够正确提取出蠕虫特征.

本文只考虑了可疑池中包含一类蠕虫的问题,下一步工作将解决在可疑池中包含多类蠕虫的特征提取问题.我们首先在可疑池中对各类蠕虫进行分类,然后再分别提取各类蠕虫的特征.此外,还会将我们的方法进一步应用到真实的网络环境中.

References:

[1] Wen WP, Qing SH, Jiang JC, Wang YJ. Research and development of Internet worms. Journal of Software, 2004,15(8):1208−1219 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1208.htm

[2] Cai M, Hwang K, Pan J, Christos P. WormShield: Fast worm signature generation with distributed fingerprint aggregation. IEEE Trans. on Dependable and Secure Computing, 2007,5(2):88−104. [doi: 10.1109/TDSC.2007.1000]

[3] Portokalidis G, Bos H. SweetBait: Zero-Hour worm detection and containment using low-and high-interaction honeypots. Computer Networks, 2007,51(11):1256−1274. [doi: 10.1016/j.comnet.2006.09.005]

[4] Ranjan S, Shah S, Nucci A, Munafo M, Cruz R, Muthukrishnan S. DoWitcher: Effective worm detection and containment in the Internet core. In: Proc. of the 26th IEEE Int'l Conf. on Computer Communications (INFOCOM 2007). Alaska: IEEE, 2007. 2541−2545. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4215899

[5] Lee KH, Kim Y, Hong SJ, Kim J. PolyI-D: Polymorphic worm detection based on instruction distribution. Lecture Notes in Computer Science, 2007,4298:45−59. [doi: 10.1007/978-3-540-71093-6_4]

[6] Kreibich C, Crowcroft J. Honeycomb: Creating intrusion detection signatures using honeypots. In: Balakrishnan H, ed. Proc. of the 2nd Workshop on Hot Topics in Networks (HotNets-II). ACM, 2003. 51−56.

[7] Newsome J, Karp B, Song D. Polygraph: Automatically generation signatures for polymorphic worms. In: Proc. of the 2005 IEEE Symp. on Security and Privacy Symp. California: IEEE, 2005. 226−241. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1425070

[8] Li ZC, Sanghi M, Chen Y, Kao M, Chavez B. Hamsa: Fast signature generation for zero-day polymorphic worms with provable attack resilience. In: Proc. of the IEEE Symp. on Security and Privacy. Washington: IEEE, 2006. 32−47. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1623999

[9] Cavallaro L, Lanzi A, Mayer L, Monga M. LISABETH: Automated content-based signature generator for zero-day polymorphic worms. In: Win BD, Lee SW, Monga M, eds. Proc. of the 4th Int'l Workshop on Software Engineering for Secure Systems (SESS 2008). Leipzig: ACM Press, 2008. 41−48.

[10] Mohammed MMZE, Chan HA, Ventura N. HONEYCYBER: Automated signature generation for zero-day polymorphic worms. In: Proc. of the Military Communications Conf. (MILCOM 2008). 2008. 1−6. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4753178

[11] Perdisci R, Dagon D, Lee W, Fogla P, Sharif M. Misleading worm signature generators using deliberate noise injection. In: Proc. of the 2006 IEEE Symp. on Security and Privacy. Atlanta: IEEE, 2006. 17−31. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1623998

[12] Tang Y, Lu XC, Hu HP, Zhu PD. Automatic generation of attack signatures based on multi-sequence alignment. Chinese Journal of Computers, 2006,29(9):1531−1539 (in Chinese with English abstract).

[13] Tang Y, Chen SG. An automated signature-based approach against polymorphic internet worms. IEEE Trans. on Parallel and Distributed Systems, 2007,18(7):879−892. [doi: 10.1109/TPDS.2007.1050]

[14] Alon N, Yuster R, Zwich U. Color-Coding. Journal of the ACM, 1995,42(4):844−856. [doi: 10.1145/210332.210337]

[15] Chen J, Lu S, Sze SH, Zhang F. Improved algorithms for path, matching, and packing problems. In: Bansal N, Pruhs K, Stein C, eds. Proc. of the 18th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA 2007). New Orleans: SIAM, 2007. 208−307.

[16] Wang JX, Huang YN, Chen JE. A motif finding algorithm based on color coding technology. Journal of Software, 2007,18(6): 1298−1307 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/1298.htm [doi: 10.1360/jos181298]

[17] Wang JX, Liu YL, Chen JE. An effective coloring algorithm for close relationship between the scales of element set and color set and its application. Chinese Journal of Computers, 2008,31(1):32−42 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2008.00032]

附中文参考文献:

[1] 文伟平,卿斯汉,蒋建春,王业君.网络蠕虫研究与进展.软件学报,2004,15(8):1208−1219. http://www.jos.org.cn/1000-9825/15/1208.htm

[12] 唐勇,卢锡城,胡华平,朱培栋.基于多序列联配的攻击特征自动提取技术研究.计算机学报,2006,29(9):1531−1539.

[16] 王建新,黄元南,陈建二.一种基于彩色编码技术的基序发现算法.软件学报,2007,18(6):1298−1307. http://www.jos.org.cn/1000-9825/18/1298.htm [doi: 10.1360/jos181298]

[17] 王建新,刘云龙,陈建二.一种在元素与颜色规模相近时的有效着色算法及其应用.计算机学报,2008,31(1):32−42. [doi: 10.3724/SP.J.1016.2008.00032]

汪洁(1980－),女,湖南桃江人,博士生,讲师,主要研究领域为网络与信息安全.

陈建二(1954－),男,博士,教授,博士生导师,主要研究领域为生物信息学,计算机理论,计算复杂性及优化,计算机网络优化算法,计算机图形理论与算法.

王建新(1969－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为生物信息学,信息安全,网络优化理论.