doi:10.3969/j.issn.1000-436x.2014.09.002

基于种子—扩充的多态蠕虫特征自动提取方法

汪洁，何小贤
（中南大学 信息科学与工程学院，湖南 长沙 410083）

摘 要：提出基于种子—扩充的多态蠕虫特征自动提取方法 SESG。SESG 算法首先按序列的权重大小将其放入一个队列，然后依次对队列中的种子序列进行扩充，从而对各类蠕虫以及噪音序列进行分类，并从分类后的蠕虫序列中提取其特征。测试结果表明，SESG 算法能够在包含噪音的可疑池中很好地区分各类蠕虫序列，更易于提取有效的蠕虫特征。

关键词：信息安全；种子扩充算法；多态蠕虫；蠕虫检测；蠕虫特征
中图分类号：TP309.5 **文献标识码**：A **文章编号**：1000-436X(2014)09-0012-08

Automated polymorphic worm signature generation approach based on seed-extending

WANG Jie, HE Xiao-xian
(School of Information Science and Engineering, Central South University, Changsha 410083, China)

Abstract: A polymorphic worm signature generation approach based on seed-extending, SESG, was proposed. Firstly, algorithm SESG puts all sequences into a queue based on their weight. Seed sequence in the queue is extended, and all kinds of worm sequences and noise sequences are classified. Finally, worm signature is generated from classified worm sequences. Experiments are run to test SESG and compared with other approaches. Experiment results show that SESG can classify worm sequences and noise sequences from suspicious flow pool over other existed approaches, which can generate effective worm signature more easily.

Key words: information security; seed-extending algorithm; polymorphic worm; worm detection; worm signature

1 引言

长期以来，网络蠕虫给 Internet 安全造成了巨大的危害，因此，对于网络蠕虫的研究一直是一个重要的课题[1]。随着 Internet 技术的发展，网络蠕虫可以不再依赖于特定系统的漏洞，同时还具有较高的隐蔽性[2]。近来，频繁爆发的各类 OSN 蠕虫正是如此。例如，2005 年，在 MySpace 上爆发的 Samy 蠕虫，2008 年的 Koobface 蠕虫，2009 年的 Mikeyy 蠕虫以及 2010 年的 Clickjacking 蠕虫。另外，多态性是网络蠕虫发展的另一趋势。如 2008 年爆发的 Conficker 蠕虫，迄今出现了 A、B、C、E 4 个版本。Conficker 主要利用 Windows 操作系统 MS08-67 漏洞来传播，同时也能借助任何有 USB 接口的硬件设备来感染。对大规模网络如社交网络进行攻击的多态蠕虫给蠕虫检测和防御带来了新的挑战。

目前，针对蠕虫的研究主要包括蠕虫建模[3~7]、研究基于异常的蠕虫检测[8~10]和基于特征的蠕虫检测[11~14]。本文主要研究基于特征的蠕虫检测。为了有效防御多态蠕虫的传播，快速且准确地提取蠕虫特征至关重要。典型的蠕虫特征提取算法包括基于最大公共子串（LCS）或固定长度负载的特征提取方法[15~17]、基于语义的特征提取方法[18]、基于多个公共子序列串（tokens）[19~24]的特征提取方法、基于多序列比对[25]的特征提取方法、Y Tang

收稿日期：2014-06-16；**修回日期**：2014-08-30
基金项目：国家自然科学基金资助项目（61202495）
Foundation Item: The National Natural Science Foundation of China (61202495)

提出的位置意识分布特征 PADS[26]（position-aware distribution signature）及其产生方法、基于近邻关系的特征提取方法[27]和基于图的特征提取方法。

在以上蠕虫特征自动提取方法中只有文献[24]和文献[25]讨论了当可疑池中存在噪音序列时如何提取蠕虫特征，另外只有文献[26]讨论了当可疑池存在多类蠕虫序列或者攻击序列时如何提取特征。文献[26]提出了标准切割（normalized cuts）算法来对可疑流量池中的蠕虫进行聚类。该算法首先通过计算获得一个相似矩阵 $\boldsymbol{\Pi}=(\Omega_{ij})$，相似矩阵中 Ω_{ij} 为可疑池中序列 i 和序列 j 的相似性，其中，$\Omega_{ii}=0$。同时获得对角矩阵 $\boldsymbol{D}_{ii}=\sum_j \Omega_{ij}$。然后定义向量 $\boldsymbol{y}$，当第 i 条蠕虫序列属于第一类时，向量 $\boldsymbol{y}$ 的第 i 个值为 1。当第 i 条蠕虫序列属于第二类时，向量 $\boldsymbol{y}$ 的第 i 个值为-1。最后求向量 $\boldsymbol{y}$ 的值，该值能使公式 $\frac{\boldsymbol{y}^{\mathrm{T}}(\boldsymbol{D}-\boldsymbol{\Pi})\boldsymbol{y}}{\boldsymbol{y}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{y}}$ 最小。向量 $\boldsymbol{y}$ 的值即为各蠕虫序列的分类。此时，可疑池被分为了 2 类，然后按同样的方法对各类中的蠕虫序列进行进一步的分类。该方法没有考虑可疑池中存在噪音的情况，且分类存在一定偏差。除了这 3 种方法，其他特征提取算法均假设可疑池中仅存在一类蠕虫序列，且可疑流量池中不包含任何噪音序列。然而，目前许多攻击大规模社交网络 OSN 蠕虫，由于其不针对任何具体的漏洞，具有较强的隐蔽性。导致包含蠕虫的可疑池流量池会包含多类蠕虫或攻击序列以及噪音序列。另外，类似于 Conficker 蠕虫具有几类多态版本，同样会使捕获蠕虫的可疑池中包含多类蠕虫样本，致使已有的特征产生系统无法产生有效的蠕虫特征。

基于已有的蠕虫特征产生方法存在的不足，它们或者在产生蠕虫特征的过程中不能处理噪音问题，或者假设可疑池中只存在一类蠕虫序列，本文提出一个新的多态蠕虫特征自动提取方法 SESG（seed extending signature generation）。SESG 算法首先会对可疑池中的序列进行处理，去除噪音的干扰，并对蠕虫序列进行分类。SESG 算法将蠕虫序列的分类问题转化为图中节点的聚类问题。算法首先计算各节点权重，具有最大权重的节点被选择作为第 1 簇的种子节点进行扩充。在扩充完第 1 个簇之后，所有剩余节点的权重被重新计算，从而获得新的种子节点及新的簇，并进行进一步扩充。最终所有节点被分成了多个簇，并从每个簇提取蠕虫特征。本文设计了多组实验测试 SESG 算法。测试表明，与其他方法相比，在包含多类蠕虫序列和噪音序列的可疑流量池中，SESG 算法能够对蠕虫进行正确分类并去除噪音的干扰，从而使提取出的蠕虫特征具有较低的误报率和漏报率。

2　基于种子—扩充的多态蠕虫特征自动提取算法

将可疑流量池及其所包含的序列转化为图和图的节点，节点的权重用 W 表示，节点与节点之间边的权重用 Θ_{ij} 表示，其中，i 和 j 表示该边所连的 2 个节点为 S_i 和 S_j。

SESG 算法包含以下 4 个部分。

1) 节点权重计算。文献[28]中提出近邻关系特征，实验证明该特征能有效检测多态蠕虫，因此本文应用该文献中的近邻特征（NRS），以及产生该特征的 Gibbs 算法。假设可疑池转化成的图为完全图，即初始时刻图中任意 2 个节点之间都有边相连。首先计算图中所有边的权值，节点 S_i 与节点 S_j 之间相连边的权值计算过程为：使用 Gibbs 算法从 2 个节点提取 NRS 特征。然后将 NRS 特征与这 2 个节点分别进行匹配，从而获得两节点的匹配得分 $\Theta(NRS,S_i)$ 和 $\Theta(NRS,S_j)$。以上 2 个匹配得分之和即为节点 S_i 与节点 S_j 之间相邻边的权重，其表达式见式（1）。

$$\begin{aligned}\Theta_{ij}&=\Theta(NRS,S_i)+\Theta(NRS,S_j)\\&=\max_{a_i=1}^{l-w+1}\frac{1}{w-1}\sum_{p=1}^{w-1}\log\frac{f_p(d_{a_i+p,a_i+p+1})}{f}+\\&\quad\max_{a_j=1}^{l-w+1}\frac{1}{w-1}\sum_{p=1}^{w-1}\log\frac{f_p(d_{a_j+p,a_j+p+1})}{f}\end{aligned}\tag{1}$$

令 $\Theta_{ii}=0$，其中，d_{a_i+p,a_i+p+1} 表示位置 a_i+p 和 a_i+p+1 处字节的近邻距离，$f_p(d_{a_i+p,a_i+p+1})$ 表示近邻距离出现频率，f 表示正常流量字节的出现频率。

定义节点的权重为与该节点相邻的各边权重之和，即 $W_i=\sum_j \Theta_{ij}$，节点权重的计算过程如图 1 所示。

```
computing note weighting(m)
    For i=1 to m
    For j=1 to m
        {NRS←Gibbs(x_i, x_j);//运行 Gibbs算法从序列 i 和序列 j 提取NRS特征;
        if(i==j) then Θ_ij=0//计算序列 x_i 和序列 x_j 之间边的权重;
        else Θ_ij=Θ(NRS, x_i)+Θ(NRS, x_j); }
  For i=1 to m
    For j=1 to m
      W_i=∑_j Θ_ij;//计算每个节点的权重
```

图 1 节点权重计算过程

2) 种子选择。在初始时刻，图中具有最大权重的节点被设定为第 1 簇的种子节点，然后对第 1 簇进行扩充。当第 1 簇扩充完成后，从簇外的节点中重新选择新的种子，构成一个新的簇，然后再进行扩充。直到所有的节点均属于某一簇，不再有游离的节点。算法的具体描述如图 2 所示。

```
selecting seed
    Call Computing Note Weighting(m)//对图中所有不属于任何一个簇的节点进行计算，获得每个节点的权重；初始时刻m为所有节点的数目n;
    x←the note with greatest weight W; S={x};//取权重最大的节点作为种子x，构成一个新簇S，此时为簇S的初始状态;
    call extending cluster(S)
```

图 2 种子选择过程

3) 簇扩充。初始时刻簇中只有一个种子节点，从簇外选择一个节点作为候选节点，该节点与种子节点之间相连边的权值最大，且该权值大于 0，则该节点被加入簇中。然后从簇中 2 个节点提取 NRS 特征。并分别计算该特征与簇外所有节点的匹配得分，计算得到的匹配得分作为簇外节点与该簇的权值。权值最大且大于 0 时，节点被加入簇中。再次从簇中所有节点提取 NRS 特征，更新簇与簇外节点边的权值并进行簇扩充。直到簇外所有节点与该簇的权值均小于 0，簇扩充结束。扩充簇的具体过程如图 3 所示。

4) 特征产生。当图中不再存在游离的节点，即所有的节点都成为了簇中的节点，之后分别从每一个簇提取 NRS。首先构造一个过滤池 M，该过滤池包含 m 条正常流序列。然后使用它来对产生的 NRS 进行过滤。m 条序列分别与 NRS 进行匹配，每一条序列均会有相应的匹配得分。将匹配得分大于 0 的序列数与总序列数 m 相比，若比值小于 ε，则 NRS 所对应的簇为蠕虫序列，NRS 为该蠕虫序列的特征，在这里 ε 是一个预先设置的值。特征提取过程如图 4 所示。

```
extending cluster(S)
  For i=1 to n //初始时刻，簇S中仅包含节点x
    If x_i is not belong to any cluster then
      Q_s←Θ_xvi;//节点x与图中不属于任意簇的节点的权值放入集合Q_s
      If Θ_xv is the greatest value in the set Q_s
        cluster(S)←v;//选择与节点 x 之间边权值最高的节点v加入到S中
      NRS←Gibbs(S);//从簇S中提取NRS特征
      Flag=True;
      While(Flag)
      {Flag=False; score_max=-1;
       For i=1 to n
       {If note x_i is not belong to any cluster,then
          {Flag=True; score_i=match(NRS, x_i);//已求出的NRS 特征与节点 x_i 的匹配得分为Score;
          If(scorei>score_max)
          then{score_max=score_i; x=x_i;}}//x为与NRS匹配得分最高的节点，其匹配得分为score_max;
          If(score_max<0)then break;
          else cluster(S)←x;}}
```

图 3 簇扩充过程

```
signature generation
  For each cluster
      {NRS←Gibbs(Cluster);
       For i = 1 to m
        {Θ_i= match(NRS, M_i); //M_i为 噪音池中的第i条序列;
          If(Θ_i>0)then P++;}
          If( P/m ≥ε)then delete NRS;}
Return(cluster, NRS)
```

图 4 特征产生过程

3 算法测试

SESG 算法采用 C++语言实现，分别测试了 SESG 算法在有噪音和无噪音环境下聚类的正确性和所产生特征的有效性。采用 Apache-Knacker 蠕虫、Blaster 蠕虫、ATPhttpd 蠕虫和 SQL Slammer 蠕虫作为测试用例。另外，还应用多种多态技术（如加密、花指令和指令替代等）对这些蠕虫测试样本进行了多态处理。

3.1 测试内容

测试主要包含以下 4 部分。

1) 可疑池中包含噪音序列和 1 类蠕虫序列。具体为可疑池包含 50 条噪音序列和 50 条 Blaster 蠕虫序列。运用 SESG 算法和基于 Normalized Cuts[23]的分类方法（后面简称 NC 算法）来对可疑池中序列进行分类。由于 NC 算法需要人为设定分类的数目，所以，首先将 NC 算法设定为 2 类。SESG 算法的分类结果如表 1 所示，NC 算法的分类结果如表 2 所示。

表 1　SESG 算法分类结果（噪音序列和 1 类蠕虫）

类别	噪音序列	Blaster 蠕虫
1 组	16 条序列	0 条序列
2 组	30 条序列	0 条序列
3 组	0 条序列	50 条序列
4 组	4 条序列	0 条序列

表 2　NC 算法分类结果（噪音序列和 1 类蠕虫）

类别	噪音序列	Blaster 蠕虫
1 组	8 条序列	50 条
2 组	42 条序列	0 条

从表 1 可以看出，Blaster 蠕虫将蠕虫序列从噪音序列中分类出来。另外，由于噪音序列各自具有不同的特性，因此噪音序列被分成了 3 个类。在表 2 中，可以看到 NC 算法的分类结果。所有 Blaster 蠕虫和 8 条噪音序列被划分为了 1 类，具体是第 1 组，所以算法没能完全区分噪音序列和蠕虫序列。

由于在产生 NRS 时设定了过滤机制，所以完全由噪音序列产生的 NRS 特征会被过滤掉。接下来从 SESG 算法分类的蠕虫序列和 NC 算法分类的蠕虫序列中提取 NRS，并使用 10 000 条正常流量序列和 10 000 条 Blaster 蠕虫序列进行误报率和漏报率的测试。另外，直接从 100 条用于分类的样本序列中应用 polygraph[28]中的算法、CCSF[24]算法和 Gibbs 算法[27]提取蠕虫，并对其进行了漏报率和误报率的测试。由于可疑池包含噪音，polygraph[19]中的算法无法产生特征。测试结果如表 3 所示，表中 Gibbs 算法产生的特征简称为 GNRS。

表 3　Blaster 蠕虫（噪音序列和 1 类蠕虫）

测试	GNRS	CCSF	Normalized Cuts+NRS	SESG+NRS
误报率	0.4315	0	0.1017	0
漏报率	0	0	0	0

从表 3 可以看出，SESG 算法所产生的 NRS 特征，其误报率和漏报率均为 0，这是因为 SESG 完全分离了蠕虫序列和噪音序列，提取特征样本序列中仅包含了 Blaster 蠕虫，所以误报率和漏报率的检测结果好。而 NC 提取特征的样本序列中很明显包含了 8 条噪音序列，所以产生的 NRS 具有一定的误报率，具体为 0.101 7。另外产生 GNRS 特征的样本序列中包含了 50 条噪音序列，所以具有较高的误报率。CCSF 算法具有抗噪音能力，所以其漏报率和误报率也均为 0。

2) 可疑流量池仅包含 2 类蠕虫序列。在该测试中，可疑池包含 SQL Slammer 蠕虫和 Blaster 蠕虫，数目各为 50 条。表 4 显示了运行 SESG 算法后，可疑流量池中序列的分类情况。表 5 显示运行 NC 算法后，可疑池中序列的分类情况。

表 4　SESG 算法分类结果（2 类蠕虫）

类别	SQL Slammer 蠕虫序列	Blaster 蠕虫序列
1 组	0 条序列	50 条序列
2 组	50 条序列	0 条序列

表 5　NC 算法分类结果（2 类蠕虫）

类别	SQL Slammer 蠕虫序列	Blaster 蠕虫序列
1 组	50 条序列	2 条序列
2 组	0 条序列	48 条序列

从表 4 可以看出，通过运行 SESG 算法，2 类蠕虫被区分开来，表 5 显示 NC 算法同样将可疑池中的序列分成了 2 组。其中第 2 组仅包含 Blaster 蠕虫序列，然而第 1 组包含 50 条 SQL Slammer 蠕虫序列和 2 条 Blaster 蠕虫序列。

与测试 1)一样，应用各类算法提取蠕虫特征。由于可疑池包含 2 类蠕虫，polygraph[19]中的算法和 CCSF[24]算法无法产生特征。在与第一个测试相同的环境下对 SESG 算法和 NC 算法所提取的特征进行了误报率和漏报率测试。测试结果发现，它们所产生特征的漏报率和误报率均为 0。

3) 可疑流量池中包含噪音序列和 2 类蠕虫序列。具体可疑流量池包含 33 条噪音序列、33 条 SQL Slammer 蠕虫和 34 条 Blaster 蠕虫。表 6 显示了运行 SESG 算法后可疑流量池中序列的分类结果，表 7 显示了运行 NC 算法后可疑流量池的分类结果。

表 6　SESG 算法分类结果（噪音序列和 2 类蠕虫）

类别	SQL Slammer 蠕虫序列(33)	噪音序列(33)	Blaster 蠕虫序列(34)
1 组	33 条	0 条	0 条
2 组	0 条	0 条	34 条
3 组	0 条	22 条	0 条
4 组	0 条	9 条	0 条
5 组	0 条	2 条	0 条

表 7　NC 算法分类结果（噪音序列和 2 类蠕虫）

类别	SQL Slammer 蠕虫序列(33)	噪音序列(33)	Blaster 蠕虫序列(34)
1 组	0 条	12 条	14 条
2 组	19 条	0 条	0 条
3 组	0 条	0 条	20 条
4 组	14 条	21 条	0 条

从表 6 可以看出，通过运行 SESG 算法，可疑流量池中的序列被分成了 5 组。其中，第 1 组为 SQL Slammer 蠕虫，第 2 组为 Blaster 蠕虫，其余 3 组均为噪音序列。从这里可以看出，噪音序列和蠕虫序列被 SESG 算法成功区分开。从表 7 可以看出，通过运行 NC 算法，可疑流量池的序列被分成了 4 组。部分噪音序列和部分 Blaster 蠕虫被分到了第 1 组。部分噪音序列和部分 SQL Slammer 蠕虫被分到了第 4 组。从测试结果可以看出，NC 算法无法将噪音序列和蠕虫序列区分开来。

和前面的测试相同，应用各类算法提取蠕虫特征。由于可疑池包含 2 类蠕虫和噪音序列，polygraph[19]中的算法和 CCSF[24]算法无法产生特征。在同样的环境下对 SESG 算法和 NC 算法所提取的特征进行了误报率和漏报率测试。测试结果如表 8 和表 9 所示。

表 8　Blaster 蠕虫（噪音序列和 2 类蠕虫）

测试	GNRS	NC+NRS		SESG+NRS
		第 1 组	第 3 组	
漏报率	0	0	0	0
误报率	0.326 6	0.422 1	0	0

表 9　SQL Slammer 蠕虫（噪音序列和 1 类蠕虫）

测试	GNRS	NC+NRS		SESG+NRS
		第 2 组	第 4 组	
漏报率	0	0	0	0
误报率	0. 385 9	0	0.434 9	0

从表 8 和表 9 可以看出，简单地运用 Gibbs 提取 NRS 无法去除噪音的干扰。当可疑流量池中包含噪音序列时，其产生的 NRS 具有一定噪音特性，所以 GNRS 具有较高的误报率。NC 算法对可疑流量池中序列分类后，第 1 组和第 4 组序列中均混合了部分噪音序列，因此产生的特征误报率较高。

4) 可疑流量池包含 4 类蠕虫序列。这 4 类蠕虫分别是 ATPhttp 蠕虫、Blaster 蠕虫、Apache-Knacker 蠕虫、SQL Slammer 蠕虫，其数目各为 25 条，在运行 SESG 算法后，可疑流量池序列的分类情况如表 10 所示。运行 NC 算法后，可疑流量池中序列的分类结果如表 11 所示。

表 10　SESG 算法分类结果（4 类蠕虫）

类别	ATPhttp 蠕虫序列（25 条）	Blaster 蠕虫序列（25 条）	Apache-Knacker 蠕虫序列（25 条）	SQL Slammer 蠕虫序列（25 条）
1 组	0 条	0 条	0 条	25 条
2 组	0 条	25 条	0 条	0 条
3 组	25 条	0 条	0 条	0 条
4 组	0 条	0 条	25 条	0 条

表 11　NC 算法分类结果（4 类蠕虫）

类别	ATPhttp 蠕虫序列（25 条）	Blaster 蠕虫序列（25 条）	Apache-Knacker 蠕虫序列（25 条）	SQL Slammer 蠕虫序列（25 条）
1 组	0 条	2 条	25 条	2 条
2 组	0 条	23 条	0 条	0 条
3 组	25 条	0 条	0 条	0 条
4 组	0 条	0 条	0 条	23 条

从表 10 可以看出，通过运行 SESG 算法，可疑流量池中的 4 类蠕虫被成功地区分开。在表 11 中可以看到，在应用 NC 算法将可疑池中的流量分成 4 组时，有些组中混合了多类蠕虫。例如 2 条 SQL Slammer 蠕虫、2 条 Blaster 蠕虫和 25 条 Apache-Knacker 蠕虫都被划分到了第 1 组。

接下来，应用各类算法提取蠕虫特征。由于可疑池包含 4 类蠕虫，polygraph[19]中的算法和 CCSF[24]算法无法产生特征。在相同的测试环境下对 SESG 算法和 NC 算法所提取的特征进行误报率和漏报率测试。同时从未分类的可疑流量池中提取 NRS 并进行测试。测试结果表明 3 类 NRS 的误报率和漏报率均为 0。

5) 可疑流量池包含噪音序列和 4 类蠕虫序列。4 类蠕虫分别是 ATPhttp 蠕虫、Blaster 蠕虫、Apache-Knacker 蠕虫、SQL Slammer 蠕虫，它们和噪音序列各 20 条。在运行 SESG 算法后，可疑流量池序列的分类情况如表 12 所示，运行 NC 算法后，可疑流量池中序列的分类结果如表 13 所示。

表 12　SESG 算法分类结果（噪音序列和 4 类蠕虫）

类别	噪音序列（20 条）	ATPhttp 蠕虫序列（20 条）	Apache-Knacker 蠕虫序列（20 条）	SQL Slammer 蠕虫序列（20 条）	Blaster 蠕虫序列（20 条）
1 组	0 条	0 条	0 条	20 条	0 条
2 组	0 条	0 条	0 条	0 条	20 条
3 组	0 条	20 条	0 条	0 条	0 条
4 组	0 条	0 条	20 条	0 条	0 条
5 组	13 条	0 条	0 条	0 条	0 条
6 组	5 条	0 条	0 条	0 条	0 条
7 组	2 条	0 条	0 条	0 条	0 条

表 13　Normalized Cuts 算法分类结果（噪音序列和 4 类蠕虫）

类别	噪音序列（20 条）	ATPhttp 蠕虫序列（20 条）	Apache-Knacker 蠕虫序列（20 条）	SQL Slammer 蠕虫序列（20 条）	Blaster 蠕虫序列（20 条）
1 组	0 条	0 条	0 条	0 条	20 条
2 组	2 条	0 条	1 条	20 条	0 条
3 组	2 条	3 条	20 条	0 条	0 条
4 组	16 条	0 条	0 条	0 条	0 条
5 组	0 条	17 条	0 条	0 条	0 条

从表 12 中可以看出，SESG 算法从噪音序列识别出了蠕虫序列，且各类蠕虫序列也被区分开。在表 13 中，应用 NC 算法可疑流量池中的序列被分成了 5 组，但是有些组中混合了噪音序列和多类蠕虫。例如，第 2 组中包含了 2 条噪音序列、1 条 Apache-Knacker 蠕虫和 20 条 SQL Slammer，第 3 组序列包含了 2 条噪音序列、3 条 ATPhttp 蠕虫和 20 条 Apache-Knacker 蠕虫。NC 算法是逐步对序列进行分类，具体分类的数目需提前设定。它会首先把可疑池中序列分成 2 类，然后再继续对分类后的序列进一步分类。所以前面分类的正确性往往会对后继分类及最终结果产生较大影响。因此 NC 算法往往无法正确地对可疑池中的蠕虫及噪音进行分类。

接下来，应用各类算法提取蠕虫特征。由于可疑池包含 4 类蠕虫和噪音序列，polygraph[19]中的算法和 CCSF[24]算法无法产生特征。在同样的测试环境下，对 SESG 算法和 NC 算法所提取的特征进行了误报率和漏报率测试。测试时，NC 算法从第 2 组序列中提取的特征被用来检测 SQL Slammer 蠕虫，从第 3 组序列中提取的特征被用来检测 Apache-Knacker 蠕虫。

表 14~表 17 显示了 3 类算法所产生的特征误报率和漏报率。从表中可以看出，当可疑流量池中包含噪音序列时，简单运用 Gibbs 提取的 NRS 具有一定噪音特性，所以 GNRS 具有较高的误报率。从前面的分析可知，NC 算法对可疑流量池中序列分类后，Apache-Knacker 蠕虫和 SQL Slammer 蠕虫所在的分组均包含了噪音序列，所以从它们中提取的 NRS 特征有一定的误报率。而 SESG 算法对可疑池中的序列进行了正确的分类，而且去除了噪音的干扰，所以产生特征的漏报率和误报率均为 0。

表 14　Blaster 蠕虫特征的漏报率和误报率

测试	GNRS	Normalized Cuts+NRS	SESG+NRS
漏报率	0	0	0
误报率	0.314 6	0	0

表 15　SQL Slammer 蠕虫特征的漏报率和误报率

测试	GNRS	Normalized Cuts+NRS	SESG+NRS
漏报率	0	0	0
误报率	0.276 3	0.047	0

表 16　Apache-Knacker 蠕虫特征的漏报率和误报率

测试	GNRS	Normalized Cuts+NRS	SESG+NRS
漏报率	0	0	0
误报率	0.327 6	0. 076	0

表 17　ATPhttp 蠕虫特征的漏报率和误报率

测试	GNRS	Normalized Cuts+NRS	SESG+NRS
漏报率	0	0	0
误报率	0.325 7	0	0

3.2　比较分析

从以上测试可以看出，当可疑池中包含噪音序列和一类蠕虫序列时，CCSF 算法和 SESG 算法均能去除噪音的干扰，且所产生特征的误报率和漏报率均为 0。然而，当可疑池中包含多类蠕虫序列时，CCSF 算法无法提取蠕虫特征。而在已有的特征产生系统中能够进行分类的方法为 Normalized Cuts，但是测试结果表明，该方法在可疑池中包含噪音时，无法去除噪音的干扰，导致产生的蠕虫特征具有较高的误报率。

另外，对 2 类分类算法的时间复杂度进行了分析。SESG 的时间复杂度为 $O(n)$，其中 n 为可疑池中序列的数目。Normalized Cuts 的时间复杂度为 $O(nm)$，其中 n 为可疑池中序列数目，m 为分类的次数。从时间复杂度可以看出，Normalized Cuts 运行时间极大地依赖于分类的次数。然而可疑池中包

含的蠕虫和噪音序列的类别是未知的，所以无法确定具体要对可疑池分成几类。而 Normalized Cuts 算法则要求在算法运行前就人为确定分类的次数，即需要确定最后分类的数目，因此给最后的分类结果带来了极大的不可预知性，从而将导致最后产生的特征不能有效检测蠕虫序列。

4 结束语

为了能够从包含多类蠕虫序列和包含噪音序列的可疑池中提取出正确的蠕虫特征，本文提出了基于种子—扩充的多态蠕虫特征自动提取算法 SESG。SESG 算法分为计算序列权重、选择种子、扩充簇和产生特征 4 个子算法。对算法的测试结果表明，与其他算法相比，SESG 算法能更好地对多类蠕虫和噪音序列进行分类。通过与 NRS 算法的结合，SESG 算法可以作为一个单独的多态蠕虫特征提取算法，也可作为其他特征提取算法（如 Polygraph 中的算法和 CCSF 算法等）的前期分类算法。

参考文献：

[1] 文伟平, 卿斯汉, 蒋建春等. 网络蠕虫研究与进展[J]. 软件学报, 2004, 15(8): 1208-1219.
WENG W P, QING S H, JIANG J C, *et al*. Research and development of internet worms[J]. Journal of Software, 2004, 15(8):1208-1219.

[2] 和亮, 冯登国, 王蕊等. 基于 MapReduce 的大规模在线社交网络蠕虫仿真[J]. 软件学报, 2013, 24(7): 1666-1682.
HE L, FENG D G, WANG R, *et al*. Mapreduce-based large-scale online social network worm simulation[J]. Journal of Software, 2013, 24(7):1666-1682.

[3] 苏飞, 林昭文, 马严等. IPv6 网络环境下的蠕虫传播模型研究[J]. 通信学报, 2011, 32(9):51-60.
SU F, LIN Z W, MA Y, *et al*. Research on worm propagation model in IPv6 networks[J]. Journal on Communications, 2011, 32(9):51-60.

[4] 吴国政, 秦志光. 大规模对等网络蠕虫仿真技术研究[J]. 通信学报, 2011, 32(8):128-135.
WU G Z, QIN Z G. Research on large-scale P2P worm simulation[J]. Journal on Communications, 2011, 32(8):128-135.

[5] 张伟, 王汝传, 李鹏. 基于云安全环境下的蠕虫传播模型[J]. 通信学报, 2012, 33(4):17-24.
ZHANG W, WANG R C, LI P. Worm propagation modeling in cloud security[J]. Journal on Communications, 2012, 33(4):17-24.

[6] 刘波, 王怀民, 肖枫涛等. 面向异构网络环境下的蠕虫传播模型 Enhanced-AAWP[J]. 通信学报, 2011, 32(12):103-113.
LIU B, WANG H M, XIAO F T, *et al*. Enhanced-AAWP, a heterogeneous network oriented worm propagation model[J]. Journal on Communications, 2011, 32(12):103-113.

[7] 杨峰, 段海新, 李星. 网络蠕虫扩散中蠕虫和良性蠕虫交互过程建模与分析[J]. 中国科学(E 辑), 2004, 34(8): 841-856.
YANG F, DUAN H X, LI X. Modeling and analyzing interaction between network worm and antiworm during the propagation process[J]. Science in China Ser E, 2004, 34(8): 841-856.

[8] 肖枫涛, 胡华平, 刘波. HPBR: 用于蠕虫检测的主机报文行为评级模型[J]. 通信学报, 2008, 29(10):108-116.
XIAO F T, HU H P, LIU B. HPBR: host packet behavior ranking model used in worm detection[J]. Journal on Communications, 2008, 29(10): 108-116.

[9] COMAR P M, LIU L, SAHA S, *et al*. Combining supervised and unsupervised learning for zero-day malware detection[A]. Proceedings of 32nd Annual IEEE International Conference on Computer Communications (INFOCOM 2013)[C]. Turin, Italy, 2013.2022-2030.

[10] KAUR R., SINGH M. Efficient hybrid technique for detecting zero-day polymorphic worms[A]. 2014 IEEE International Advance Computing Conference (IACC)[C]. Gurgaon, India, 2014.95-100.

[11] 唐勇, 诸葛建伟, 陈曙晖等. 蠕虫正则表达式特征自动提取技术研究[J]. 通信学报, 2013, 34(3):141-147.
TANG Y, ZHUGE J W, CHEN S H, *et al*. Automatic generating regular expression signatures for real network worms[J]. Journal on Communications, 2013, 34(3):141-147.

[12] 王平, 方滨兴, 云晓春. 基于自动特征提取的大规模网络蠕虫检测[J]. 通信学报, 2006, 27(6): 87-93.
WANG P, FANG B X, YUN X C. Large scale network worm detection using automatic signature extraction[J]. Journal on Communications, 2006, 27(6):87-93.

[13] KAUR R, SINGH M. A survey on zero-day polymorphic worm detection techniques[J]. IEEE Communications Surveys & Tutorials, 2014:1-30.

[14] PORTOKALIDIS G, BOS H. Sweetbait: zero-hour worm detection and containment using low-and high-interaction honeypots[J]. Computer Networks, 2007, 51(5):1256-1274.

[15] CAI M, HWANG K, PAN J, *et al*. Wormshield: fast worm signature generation with distributed fingerprint aggregation[J]. IEEE Transactions on Dependable and Secure Computing, 2007, 4(2):88-104.

[16] RANJAN S, SHAH S, NUCCI A, *et al*. Dowitcher: effective worm detection and containment in the internet core[A]. IEEE INFOCOM 2007[C]. Alaska, USA, 2007.2541-2545.

[17] MOHAMMED MMZE, CHAN H A, VENTURA N, *et al*. An automated signature generation method for zero-day polymorphic worms based on multilayer perceptron model[A]. 2013 International Conference on Advanced Computer Science Applications and Technologies (ACSAT)[C]. Zhengzhou, China, 2013.450-455.

[18] YEGNESWARAN V, GIFFIN J T, BARFORD P, *et al*. An architecture for generating semantics-aware signatures[A]. Proceedings of the 14th

Conference on USENIX Security Symposium[C]. Baltimore, 2005.

[19] NEWSOME J, KARP B, SONG D. Polygraph: automatically generating signatures for polymorphic worms[A]. Proceedings of 2005 IEEE Symposium on Security and Privacy Symposium[C]. Oakland, California, 2005.226-241.

[20] LI Z, SANGHI M, CHEN Y, *et al*. Hamsa: fast signature generation for zero-day polymorphic worms with provable attack resilience[A]. Proceedings of IEEE Symposium on Security and Privacy[C]. Berkeley/Oakland, California, 2006.32-47.

[21] CAVALLARO L, LANZI A, MAYER L, *et al*. LISABETH: automated content-based signature generator for zero-day polymorphic worms[A]. Proceedings of the Fourth International Workshop on Software Engineering for Secure Systems[C]. Berlin, Germany, 2008. 41-48.

[22] BAYOGLU B, SOGUKPINAR I. Polymorphic worm detection using token-pair signatures[A]. Proceedings of the 4th International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing[C]. New York, USA, 2008.7-12.

[23] MOHAMMED MMZE, CHAN H A, VENTURA N. Honeycyber: automated signature generation for zero-day polymorphic worms[A]. IEEE Military Communications Conference, MILCOM 2008[C]. New York, USA, 2008.1-6.

[24] WANG J, WANG J X, CHEN J E, *et al*. An automated signature generation approach for polymorphic worm based on color coding[A]. IEEE ICC 2009[C]. Dresden, Germany, 2009.1-6.

[25] TANG Y, XIAO B, LU X. Using a bioinformatics approach to generate accurate exploit-based signatures for polymorphic worms[J]. Computers & Security, 2009, 28(8):827-842.

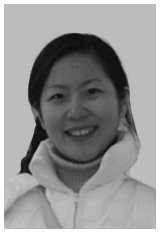
[26] TANG Y, CHEN S. An automated signature-based approach against polymorphic internet worms[J]. IEEE Transactions on Parallel and Distributed Systems, 2007,18(7):879-892.

[27] BAYOGLU B, SOGUKPINAR L. Graph based signature classes for detecting polymorphic worms via content analysis[J]. Computer Networks, 2012, 56(2):832-844.

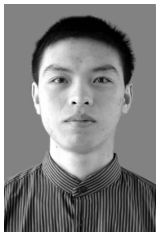
[28] 汪洁, 王建新, 刘绪崇. 基于近邻关系特征的多态蠕虫防御方法[J]. 通信学报, 2011, 32(8):150-158.

WANG J, WANG J X, LIU X C. Novel approach based on neighborhood relation signature against polymorphic Internet worms[J]. Journal on Communications, 2011, 32(8):150-158.

作者简介：



汪洁（1980-），女，湖南桃江人，博士，中南大学副教授，主要研究方向为网络与信息安全等。



何小贤（1980-），男，湖南新化人，博士，中南大学讲师，主要研究方向为群体智能、自然计算、复杂系统等。