一种改进的基于扩展攻击树模型的木马检测方法

陈燕红 欧毓毅 凌 捷
(广东工业大学计算机学院 广东 广州 510006)

摘 要 木马作为恶意程序的一种,经常被作为黑客入侵利用的手段,这对网络安全和信息安全将造成极大的危害。提出一种改进的基于扩展攻击树模型的木马检测方法。通过分析 PE 文件,采用静态分析和动态行为监控技术相结合的检测方法提取程序 API 调用序列;并用信息增益的方法筛选出木马关键 API 短序列集合,作为构建扩展攻击树模型的特征库;将待检测程序以 API 短序列为行为特征与模型节点进行匹配、分析,同时改进了匹配节点的权值和危险指数的算法。最后给出扩展攻击树模型调整与优化的方法。实验结果表明,改进后的方法不仅在木马检测效率、准确度方面有较好的表现,还能检测出经过升级变种的木马。

关键词 木马检测 扩展攻击树 API 调用 改进算法

中图分类号 TP309 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2016.08.068

AN IMPROVED TROJANS DETECTION METHOD BASED ON EXTENDED ATTACK TREE MODEL

Chen Yanhong Ou Yuyi Ling Jie
(*Faculty of Computer, Guangdong University of Technology, Guangzhou 510006, Guangdong, China*)

Abstract Trojans, as one kind of malwares, are often used as the hacking means, and this will cause great harm to the security of network and information. We proposed an improved Trojans detection method which is based on extended attack tree model. First, by analysing PE files, the method extracts the call sequence of program's API using the detection method combining the static analysis and dynamic behaviour monitoring; Then it sifts the key short call sequence set of API using the method of information as the feature library for building extended attack tree model; Furthermore, it takes the API short call sequence as the behavioural feature of the detecting program to match the model nodes and then analysis them, meanwhile improves the algorithms of matching nodes' weight and risk index as well. Finally, we introduced the method to adjust and optimise the extended attack tree model. Experimental results showed that the improved method had better performance in Trojans detection efficiency and accuracy, and could detect the upgraded variant of the Trojan as well.

Keywords Trojans detection Extended attack tree API calls Improved algorithm

0 引 言

木马检测一般分为静态和动态检测技术,然而这两类检测技术都有其不足:静态检测技术在面对已知木马程序的隐藏和变化时略显不足,对于未知木马程序亦是无能为力;而动态检测技术不仅因为占用较多的系统资源导致效率不高,对于已经发现的木马程序,其运行造成的危害已是事实。

攻击树模型有着直观的结构,能够较好地反映系统威胁路径,在木马程序入侵检测应用方面的研究相对较多。文献[1]在传统攻击树模型上,对每个节点加入属性信息实现了对模型的扩展,不足之处在于其对节点间关系定义不全面、危险指数算法过于简单,将影响检测的准确度;文献[2]提出的一种改进攻击树结构,并设计了危害权值算法,不仅更直观地体现了攻击路径,而且达到了更好的描述和判断程序的恶意性的目的。其不足在于未能考虑攻击树中不同层次节点对 PE 文件危险值比重的差异;文献[3]中改进的攻击树模型是在传统攻击树的基础上加入参数集合较好地描述了 API 调用序列情况,但是其对节点权重信息描述不明确;文献[4]提出一种通过静态分析 PE 文件获取 API 短序列与攻击树进行匹配,提高了匹配效率,然而当前很多木马程序通过动态加载 DLL 获得函数地址完成系统调用以躲避静态扫描[5],静态分析 API 序列方法的不足将日益凸显。

本文在分析了 PE 文件头部特征信息[6]的基础上,重点研究了攻击树模型的扩展,API 调用序列的提取、分析以及算法的改进等,提出一种改进的基于扩展攻击树模型的木马检测方法。首先搜集总结了大量木马程序,并对其 API 调用序列进行提取、划分以及筛选作为构建模型的特征库;同时对攻击树模型节点信息进行改造扩展,基于扩展的模型,根据不同层次目标节点和叶子节点的权重、发生概率以及程序调用 API 短序列匹配度等的不同,改进了相应算法,再通过计算程序的危险指数与预先设置好的阈值进行比较,以判断程序是否为木马程序;最后给出实

收稿日期:2015 - 02 - 15。广东省自然科学基金重点项目(S2012020011071);广东省高校科技创新项目(2013KJCX0064)。**陈燕红**,硕士生,主研领域:信息安全技术。**欧毓毅**,副教授。**凌捷**,教授。

现攻击树模型的调整与优化的方法。

1　扩展攻击树模型

攻击树模型是一种采用树形结构来描述攻击者对目标系统进行攻击的方法、路径。在一棵攻击树的树形结构中,可以分为“或”(Or)、“与”(And)、“顺序与”(Sand)三种关系。现假设 G,a,b 三个节点分别代表父节点、左子节点、右子节点,则上述三种关系的图形表示如图 1 所示。

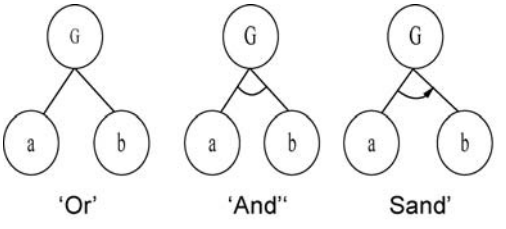

图 1　目标节点关系图

其中,‘Or’关系表示只要有包含一个以上的子节点都将导致父节点的完成;‘And’关系表示当且仅当所有子节点的完成才能导致父节点的完成;‘Sand’关系表示父节点的完成必须是所有子节点按相应顺序完成方可,这里指子节点从左到右的顺序。

扩展攻击树 $T = (V,E,Attribute,Parameter)$,其中 $V(T)$ 是上述三种关系的非叶子节点及叶子节点为集合元素的非空并且有限的集合,根节点记为‘ROOT’,用以表示攻击者要达成的最终攻击目标。自上而下,子节点代表攻击的子目标,逐步细分到叶节点代表危险 API 调用短序列。$E(T)$ 中的元素是 $V \times V$ 的子集,用来代表树中的每条边,即攻击的路径。$Attribute(T)$ 是节点属性集合,由七元组 (S,C,L,P,R,W,D) 组成,与每个节点相关联。S 是布尔型变量,表示节点在匹配过程中的状态:得到匹配的节点的 S 为 TRUE,记为“高亮节点”,反之记为“非高亮节点”。C 表示为 API 的名称或者目标的文本描述。L 表示节点所在攻击树中的层次,其中 $L(ROOT) = 0$,$ROOT$ 子节点的 L 值为 1,依次向下类推。P 指节点代表的事件发生的可能性。R 为三态变量(与/或/顺序与),表示子节点之间的关系,对于叶子节点无意义。W 和 D 是数值型变量,W 表示节点代表的事件发生后对系统产生的威胁,为恶意性权值;D 表示综合计算得到的节点的危险指数。$Parameter(T)$ 是参数信息集合,由 $time$、$paramlist$ 两个变量组成,用以记录 API 调用时的参数信息,$time$ 表示节点内部的时间序编号,$paramlist$ 变量则用来记录参数调用链表。对于非叶子节点该值为 NULL。初始状态下,模型中所有节点都处于‘非高亮’状态,Parameter(T)为空状态,而非叶子节点的 D 暂时也为空。

2　相关技术的研究

2.1　API 调用序列的提取以及木马库的生成

本文在分析了 PE 文件的特征字符串、API 调用的数量等头部特征信息的基础上,结合静态和动态方法获取 API 调用序列(包含调用 API 的参数信息),大致流程如图 2 所示。

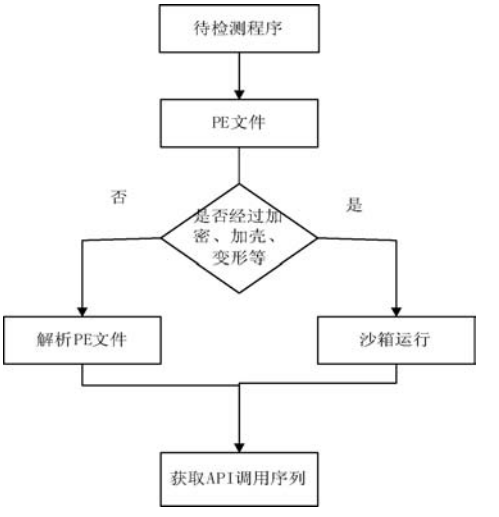

图 2　API 调用序列提取流程

在程序运行前,首先判断 PE 文件是否经过加壳、变形等,若无则通过静态解析 PE 文件的方法提取 API 调用序列以尽可能地提高检测效率;若是则将程序置于沙箱中运行以避免其对系统的危害,通过行为监控技术[7,8]拦截程序执行过程中的 API 调用以提高检测准确率。再将提取到的 API 调用序列,采用 $K = 6$ 长度的滑动窗口生成 API 短序列,如图 3 所示。

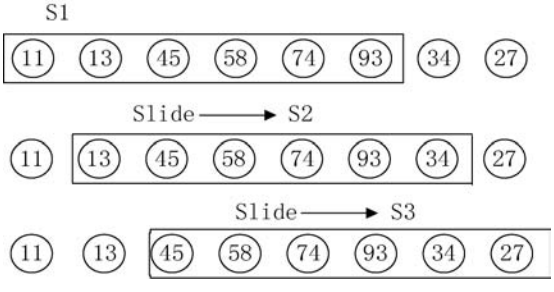

图 3　滑动窗口大小 $K = 6$ 时 API 调用短序列

针对部分木马短序列无法有效区分木马程序与正常程序,同时造成不必要的信息冗余,采用信息增益筛选出高识别度、高准确度的关键 API 短序列集合 $A(F) = \{F1,F2,\cdots,Fm\}$ 作为构建模型的特征库。信息增益公式见下:

$$Grain(A) = Info(Q) - Info_A(Q)$$

其中,Q 为标记类的训练集,包括:木马程序($C1$)、正常程序($C2$)。$Info(Q)$ 即表示对 Q 进行分类所需要的期望信息,而 $Info_A(Q)$ 表示按 A 划分后再对 Q 进行分类所需的期望信息,A 为木马调用的 API 短序列,公式具体为:

$$Info(Q) = -\frac{|C1|}{|Q|}\log_2\frac{|C1|}{|Q|} - \frac{|C2|}{|Q|}\log_2\frac{|C2|}{|Q|}$$

其中,$|C1|$、$|C2|$ 分别表示 Q 训练集中木马和正常程序的数量,$|Q| = |C1| + |C2|$ 表示程序的总数量。

$$InfoA(Q) = -\sum_{j=0}^{1}\frac{|QJ|}{|Q|}\left(-\frac{|C1j|}{|QJ|}\log_2\frac{|C1j|}{|Qj|} - \frac{|C2j|}{|Qj|}\log_2\frac{|C2j|}{|Qj|}\right)$$

其中,$|Qj|$ 即为按 A 划分 Q 得到的子集的数量,有 $|Q1|$ 和 $|Q0|$ 分别表示 Q 训练集中程序调用、未调用 A 短序列得到的子集数量。$Grain(A)$ 值越大,表示对应 API 短序列对木马程序的检测影响就会越大,反之相反。

2.2　原始扩展攻击树的构建

通过分析木马常见的恶意行为,将原始扩展攻击树划分为 8 个部分,如图 4 所示。

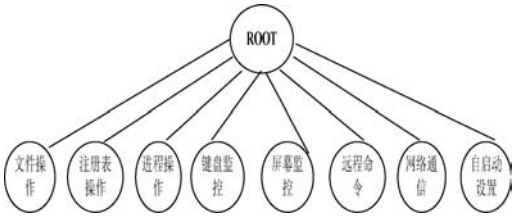

图4 原始扩展攻击树的划分

具体地,根据对上述恶意行为的划分,将木马特征库 A(F)中的 API 短序列按照序列达成的目标、目标间的依赖关系构建出若干最大木马扩展攻击树,再将这些树以或的关系合并在一起作为 ROOT 根节点的子节点,即完成原始扩展攻击树的构建,如图5所示(图中省略了部分目标节点、叶子节点)。

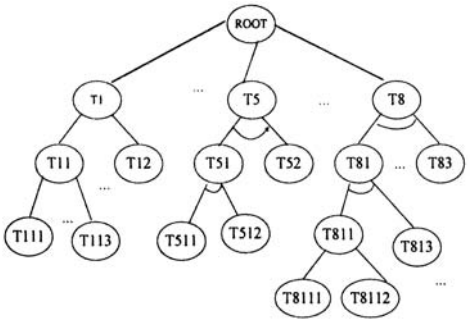

图5 构建的攻击树实例

对于攻击树中节点的命名采用 < T + 父节点编号 + 子节点顺序号 > 的方法,不仅可以直观地体现攻击过程,还能准确地定位节点所在的层次位置,便于节点权重计算等。

2.3 相关算法研究

1) 节点发生的可能性 P 的推算方法[9] 其中 F 表示父节点,Kn 表示子节点。

(1) 对于"Or 关系"的子节点,其父节点的发生可能性等于各子节点发生可能性的最大值。

$$P(F) = \max\{P(K1), P(K2), \cdots, P(Kn)\}$$

(2) 对于"And 关系"的子节点,其父节点的发生可能性等于各子节点发生可能性的乘积。

$$P(F) = P(K1) \times P(K2) \times \cdots \times P(Kn)$$

(3) 对于"Sand 关系"的子节点,其父节点的发生可能性需要借助条件概率来计算。

$$P(F) = P(K1) \times P(K2/K1) \times \cdots \times P(Kn/\ K1\ K2 \cdots Kn - 1)$$

(4) 对于叶子节点 *leaf* 采用多属性效用理论[10] 计算其发生的可能性,给每个叶子节点附上三个基本属性,*cost*、*det*、*diff* 分别用来表示完成该叶子节点所需要的成本、可能被检测到的等级和难易度,算法如下。

$$P(leaf) = Wcost \times U(cost) + Wdet \times U(det) + Wdiff \times U(diff)$$

其中: *leaf* 表示任意的一个叶子节点;$P(leaf)$ 表示该叶子节点发生的可能性;$Wcost$、$Wdet$、$Wdiff$ 用以表示对应参数的权重系数;而 $U(cost)$、$U(diff)$、$U(det)$ 则用以表示相应参数的效用性。

2) 恶意性权值 W

叶子节点是由具体 API 函数代表的一个行为事件,通过行为要素得出单个叶子节点的恶意性权值[11]。主体 U 为调用 API 函数的进程或线程,客体 O 为 API 函数的操作对象,动作 B 为 API 函数代表的操作,API 部分参数值作为行为的备注部分 N。设 Wu、Wo、Wb、Wn 分别代表主体、客体、行为操作以及附加参数值代表的权重,L 为当前节点所在的层数,为不同层节点设置权值可以更好地体现危害差异性。对于匹配的叶子节点,其所在层次离总目标越远其权重相应越低,本文采用如下公式计算其恶意性权值。

$$W(leaf) = (Wu + Wo + Wb + Wn) \times 1/L$$

对于非叶子节点,其恶意性权值 W 由其所有子节点的危险指数及包含的叶子点间的相互关系综合决定。

$$W(F) = Fun(K1, K2, \cdots, Kn)$$

为'Or'节点时,W 取子节点中最大的危险指数为值;为'And'或者'Sand'节点时,将子节点危险指数求和作为 W 值,以体现子节点共同作用导致父节点目标的达成,公式如下:

$$W(F) = \mathrm{Max}\{D(K1), D(K2) \cdots D(Kn)\} \times 1/L$$

$$W(F) = (D(K1) + D(K2) + \cdots + D(Kn)) \times 1/L$$

3) 危险指数 D

危险指数标识危险级别,表示节点与木马程序的相似程度,由自身权值及发生概率计算而得。所有最大高亮子树根节点的危险指数 D 值的和为对应木马程序的危险指数 $D(F)$。为了有效地降低漏报率和误报率,文献[12]在计算程序危险指数的时候将程序调用 API 集合的规模考虑在内,本文考虑 API 序列间的依赖关系,通过计算 API 短序列匹配度的方法以更准确地描述程序的恶意性:假设目标 API 调用短序列的总数为 n,在扩展攻击树中匹配到的短序列个数为 m,那么 m/n 的即为目标短序列集合中 API 序列的匹配度,记为 $P(H)$。通过这种方法可以有效地降低正常程序调用 API 过多引起的高匹配度从而产生误报,以及木马程序调用 API 较少造成低匹配度从而产生漏报。式(1)、式(2)分别表示计算子目标和最终目标的危险指数。

$$D = P \times W \tag{1}$$

$$D(F) = \mathrm{sum}(D) \times P(H) \tag{2}$$

3 基于扩展攻击树的木马检测方法

3.1 匹配分析

将待检测程序中提取、划分的目标 API 短序列集合 $G(A) = \{A1, A2, \cdots, Am\}$ 以短序列为单位同扩展攻击树模型进行匹配,同时把相应 API 调用参数信息记录在 Parameter(T)中,再对攻击树模型进行叶子节点及非叶子节点按照相应规则(与/或/顺序与)作高亮标记。设置一个阈值作为判断标准,最后计算危险指数的值来判断程序是否为木马程序。

例如某可疑木马程序部分的调用序列及参数调用记录如如下:

(1) 可疑木马程序代码片段

…

OpenFile(IN:N:XXXOUT:I:32),

WriteFile(IN:I:32 B:XXXOUT:),

RegOpenkeyEx(IN:N:HKLM\SYSTEM\CurrentControlSet\Services\XXXOUT:0),

CreateFile(IN:N:X:\filename.exeOUT:0)

RegCreateKey(IN:N:HKLM\SYSTEM\CurrentControlSet\Services\XXXOUT:0)

CreateProcess(IN:N:ProcessNameOUT:0)

RegCloseKey(IN:N:HKLM\SYSTEM\CurrentControlSet\Services\XXXOUT:0),

…

(2) 提取的 API 调用序列

Index	APIName
11	OpenFile( )
13	WriteFile( )
45	RegOpenkeyEx( )
58	CreateFile( )
74	RegCreateKey( )
93	CreateProcess( )
34	RegCloseKey( )
105	MessageBox( )
46	Send( )
…	

将上述可疑 API 调用序列通过滑动窗口划分成短序列,按相应规则匹配攻击树后得到的高亮节点扩展攻击树,如图 6 所示。

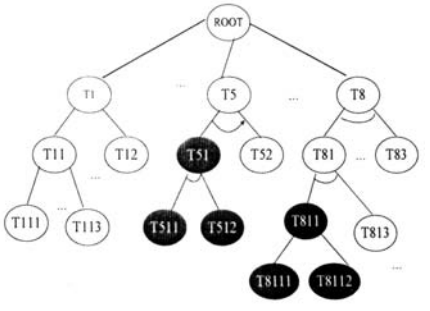

图 6　匹配攻击树

3.2　动态调整扩展攻击树方法

原始扩展攻击树作为木马检测的核心特征库,需要不断地更新和补充使得检测更加准确有效。检测过程中,将程序中出现的未被匹配的敏感 API 调用序列按照一定的规则添到新的模型分支中:对于相同的操作对象而动作不同,归为'And'或者'Sand'关系,并通过参数信息进行区分;对于相同动作而操作对象不同,归为'Or'关系。同时,采用如下方法对扩展攻击树模型进行调整优化:

1) 或节点合并法

针对今后出现的新的攻击方式实现的相同攻击目标,使用或节点进行合并,例如:完成攻击树中 T21 目标需要调用 API 序列(A1,A2,A3),若发现不同的 API 调用(A4,A5)序列亦可完成 T21 目标,则将序列(A1,A2,A3)和序列(A4,A5)分别置于 T21 新的子节点 T211,T212 下实现合并。

2) 节点精简法

通过舍去扩展攻击树模型中无作用或作用不大的节点以实现展攻击树模型的优化。即保证完成攻击目标序列的任意子序列无法达成该攻击目标。

3) 相似攻击树统一法

对于树中出现的相似的攻击树,按结构从简的原则选择其一,从而提高程序运行效率并节省存储空间,如图 7 所示。

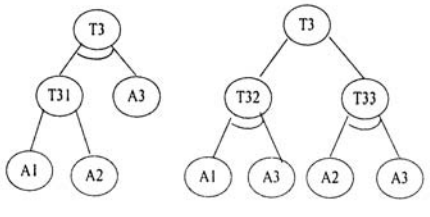

图 7　相似攻击树

通过分析得到这两颗攻击树中达成目标 T3 具有相同的 API 调用序列,都是{(A1,A3)(A2,A3)}和(A1,A2,A3),因此它们是相似的,通过分析选左图作为模型子树。

4　实验结果与分析

(1) 通过分析"红狼远控"和"灰鸽子黑防"这两种远程控制木马程序,对其进行升级都附上新的攻击目标:获取目标机上的.jpg 图像文件之后将其删除的功能作为新的攻击目标。通过匹配之后,再计算危险指数得到实验结果如表 1 所示。

表 1　实验结果 1

木马程序	升级前危险指数	升级后危险指数
红狼远控	206	245
灰鸽子黑防	213	271

实验结果表明,本文方法可以有效地检测出经过升级后的木马。

(2) 从网上下载了 639 个木马文件以及 Windows XP 系统目录下的纯净 PE 文件。以 2:8 的比例将这些文件分别作为测试集和训练数据。将本文方法与文献[1]、文献[4]方法进行分析比较 FN(木马被识别为正常程序)和 FP(正常程序被识别为木马)指数得到如表 2 所示。

表 2　实验结果 2

方法比较	FN %	FP %
文献[1]	15.3	13.4
文献[4]	10.3	8.6
本文	8.7	6.8

通过表 2 的实验数据比较分析看到,改进后的方法较大地降低了了 FN 和 FP 指数,提高了检测的准确度。

5　结　语

本文提出一种基于攻击树模型的木马检测方法,先通过扩展攻击树节点信息等实现更精确的匹配模型;再采用静态分析和动态行为监控技术相结合的方法可以比较有效而准确地提取到程序的 API 调用序列,并对相关算法进行研究,改进了权值、危险指数的算法。最后给出扩展攻击树模型调整与优化的方法以实现模型特征库的更新与补充。实验结果表明,改进后的方法在兼顾木马检测效率、准确度方面有较好表现的同时,还可以检测出经过升级变种的木马。

参　考　文　献

[1] 杨彦,黄皓. 基于攻击树的木马检测方法[J]. 计算机工程与设计,2008,29(11):2711-2714.

[2] 谢乐川,袁平. 改进攻击树恶意代码检测方法[J]. 计算机工程与设计,2013,34(5):1599-1604.

[3] 陈丹伟,唐平,周书桃. 基于沙盒技术的恶意程序检测模型[J]. 计算机科学,2012,39(6A):12-14.

[4] 牛冰茹,刘培玉,段林珊. 一种改进的基于攻击树的木马分析与检测[J]. 计算机应用与软件,2014,31(3):277-280,330.

(下转第 333 页)

高,检测时间比 HOG 等算法少得多,这主要是因为使用 WTA hash 编码后滤除图像特征中的冗余信息使图像特征变得较为稀疏,从而使后续算法较为迅速,降低了检测时间。

表 1　各种算法检测效率对比

方法	人体数目	检测正确数	检测率	平均检测时间
HOG	300	288	96%	23.69 毫秒
MHOG		289	96.33%	51.71 毫秒
HOG + LBP		289	96.33%	26.48 毫秒
多尺度多视角算法		292	97%	6.53 毫秒

6　结　语

本文针对人体检测面临的背景复杂、相互遮挡等难题,使用扩展多尺度方向和经 WTA hash 编码的多尺度梯度方向直方图两种特征提取方法来提取图像的粗特征和精特征,并使用多层多视角的级联分类器进行人体检测。其中扩展的多尺度方向特征能有效表征人体边缘特征,而 WTA hash 编码的多尺度梯度方向直方图能有效滤除精特征中的冗余信息从而提高算法的检测效率,多视角结合复杂背景处理和特征重装可以在一定程度上解决复杂背景下人群相互遮挡的问题。实验证明,多尺度多视角的检测方法在提高检测精度的同时检测效率也保持较高的水平,但 WTA hash 编码后造成复杂背景下像素较低的人体目标检测存在一定缺陷,这也需要在以后的研究中不断改进的重点问题。

参　考　文　献

[1] Aggarwal J, Ryoo M. Human activity analysis: a review [J]. ACM Computing Surveys, 2011, 43(3): 1-47.

[2] 欧阳毅,张三元,张引. 基于窗口边缘梯度热能的人体遮挡多惊讶检测算法[J]. 电子与信息学报,2012,34(4): 858-864.

[3] 雷庆,陈锻生,李绍滋. 复杂场景下的人体行为识别研究新进展[J]. 计算机科学, 2014,41(12):1-7.

[4] Ji Shuiwang , Xu Wei, Yang Ming, et al. 3D Convolutional Neural Networks for Human Action Recognition [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35(1): 221-231.

[5] 邵彧. 基于局部图块匹配的行人跟踪算法[J]. 计算机工程与设计,2014,35(10):3518-3522.

[6] Ni E Z, Zhou C L, Jiang M J. A radical cascade classifier for handwritten Chinese character recognition [J]. Journal of Software, 2012, 7(10): 2294-2300.

[7] 黄炎,范赐恩,朱秋平,等. 联合梯度直方图和局部二值模式特征的人体检测[J]. 光学精密工程,2013,21(4):1047-1053.

[8] 刘亚洲. 基于时空分析和多粒度特征表示的人体检测方法研究[D]. 哈尔滨:哈尔滨工业大学,2009.

[9] Lin Zhe, Hua Gang, Davis L S. Multi-scale shared features for cascade object detection[C]//Proceeding of the IEEE International Conference on Image Processing, Orlando, FL, 2012: 1865-1868.

[10] Wang Xiaoyu, Han T X, Yan Shuicheng. An HOG-LBP human detector with partial occlusion handling[C]//Proceeding of IEEE International Conference on Computer Vision, Kyoto, 2009:32-39.

[11] 叶齐祥,焦建彬,蒋树强. 基于多尺度方向特征的快速鲁棒人体检测算法[J]. 软件学报,2011,22(12):3004-3014.

[12] 李梦涵,熊淑华,熊文,等. 多尺度级联行人检测算法的研究与实现[J]. 计算机技术与发展,2014,24(8):10-13.

[13] Wantanbe T, Ito S, Yokoi K. Co-occurrence histograms of oriented gradients for pedestrian detection [J]. LNCS, 2009, 52(14): 37-47.

[14] 孙宏国,李天然,蒲宝明,等. 复杂背景下人体检测算法[J]. 计算机系统应用,2013,22(4):134-138.

[15] Poppe R. A survey on vision-based human action recognition[J]. Image and Vision Computing, Elsevier B. V. 2010, 28(6): 976-990.

[16] Kratz L, Nishino K. Tracking pedestrians using local spatiotemporal motion patterns in extremely crowded scenes [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, 34(5): 987-1002.

[17] Li W T, Chang H S, Lien K C, et al. Exploring visual and motion saliency for automatic video object extraction [J]. IEEE Transactions on Image Processing, 2013, 22(7): 2600-2610.

[18] Dollar P, Wojek C, Schiele B, et al. Pedestrian detection: an evaluation of the state of the art [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, 34(7): 743-761.

(上接第 288 页)

[22] Li M, Wang J, Chen X, et al. A local average connectivity-based method for identifying essential proteins from the network level[J]. Computational Biology and Chemistry, 2011, 35(3): 143-150.

[23] Deshwar A G, Morris Q. PLIDA: cross-platform gene expression normalization using perturbed topic models[J]. Bioinformatics, 2014, 30(7):956-961.

[24] Mewes H W, Frishman D, Mayer K F X, et al. MIPS: analysis and annotation of proteins from whole genomes in 2005[J]. Nucleic Acids Research, 2006, 34(1):169-172.

[25] Cherry J M, Adler C, Ball C, et al. SGD: Saccharomyces genome database[J]. Nucleic Acids Research, 1998, 26(1):73-79.

[26] Zhang R, Lin Y. DEG 5.0: a database of essential genes in both prokaryotes and eukaryotes[J]. Nucleic Acids Research, 2009, 37(1): 455-458.

[27] Saccharomyces Genome Deletion Project [OL]. http://www-sequence.stanford. edu/group/.

(上接第 311 页)

[5] 张程,马兆丰,钮心忻,等. 一种 API 动态序列分析和 DAG-SVM 多类支持向量机的未知病毒检测方法[J]. 小型微型计算机系统,2012,12(12):2724-2728.

[6] Schultz M G, Eskin E, Zadok F, et al. Proceedings of the 2001 IEEE Symposium on Security and Privacy[C]//Washington: IEEE Computer Society, 2001:38-49.

[7] 谢燕江. 一种基于轻量级虚拟化的沙盒机制[D]. 湖南:湖南大学软件学院,2012.

[8] 张永超. 基于虚拟执行技术的恶意程序监测系统研究与实现[D]. 国防科学技术大学,2011.

[9] 甘早斌,吴平,路松峰,等. 基于扩展攻击树的信息系统安全风险评估[J]. 计算机应用研究,2007,24(11):153-160.

[10] 钟倩,方勇,刘亮,等. 基于攻击树的文件风险评估方法[J]. 通信技术,2011,5(44):77-79.

[11] Xia Jiang, Weiwei Qi. An Improved Attack Tree Algorithm Based on Android[J]. Journal of Software Engineering, 2014, 34(1):1-8.

[12] 乐洪州. 基于扩展攻击树的木马检测技术研究[D]. 大连:大连海事大学,2013.