

动态实时身份认证的方法研究

毛传武, 程阳, 余文明

(华中科技大学自动化学院, 湖北 武汉 430074)

摘要: 为减小身份伪装给企业和组织带来的内部威胁, 提出了一种全场景下动态实时身份认证方法。利用伪造的鼠标异常构建动态身份认证场景; 当出现连续的动态认证异常时, 系统引导用户进行了由记忆游戏构建的固定场景身份认证。对 15 位用户进行了身份认证实验, 得到了 3.0% 的拒真率、1.8% 的容错率以及 8.12 s 的认证时间。实验表明, 动态场景和固定场景身份认证的有效配合, 不仅克服了固定场景无法实时身份认证的弊端, 而且保证了用户的体验, 同时提高了身份认证的准确率。

关键词: 内部威胁; 身份认证; 身份识别; 支持向量机; 鼠标动力学

中图分类号: TP309

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2016.00040

Method of dynamic real-time authentication

MAO Chuan-wu, CHENG Yang, YU Wen-ming

(School of Automation, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: To solve the problem organization's insider threats which was brought by internal staff through identity spoofing. A method of dynamic real-time authentication was presented. By faking exceptions of mouse to build dynamic authentication, when continuously consecutive authentication were abnormal, user was forced into fixed scene certification which was building of memory game. An identity verification experiment, in which 15 participants were involved, showed that the performance of proposed method was encouraging with FAR of 1.8% and FRR of 3.0% and authentication time of 8.12 s for authentication. The experiment shows that the method is not only real-time but also has a better user experience and accuracy.

Key words: insider threat, identity authentication, identity monitoring, SVM, mouse dynamics

1 引言

内部攻击对组织信息安全的威胁一直都是最大的挑战^[1~3]。2015 年 1 月, Vormetric 发布的《内部威胁报告》显示, 93% 的美国组织易受内部威胁的影响。2015 年 12 月, NUIX 安全专家的一篇文章为《人的行为是企业安全的最大威胁》的报告指出, 员工几乎成为信息安全的最大软肋。

身份伪装攻击是内部威胁最常用的手段之一。身份伪装攻击是指内部员工将自己伪装成其

他合法用户来窃取信息^[4]。相比普通的外部攻击, 内部攻击给安全人员带来的挑战更大^[5]。一方面, 内部攻击者可以合法地访问内部网络, 常用的防护技术如防火墙等无法监控到其攻击行为; 另一方面, 攻击者对系统安全规则更加熟悉, 更容易绕过现有的安全防护技术^[5]。

常用的身份验证技术可分为 3 种: 记忆性的, 如密码等; 携带性的, 如 ID 卡等; 生理特征的, 如指纹等。这些方法虽然在一定程度上能够识别出身份伪装攻击, 但是都只能用于系统登录前的

一次性身份认证, 无法在计算机使用过程中持续地监控用户身份。

基于生物行为特征的身份认证是一种持续型的认证, 主要分为键盘动力学和鼠标动力学的身份认证^[4]。随着图形界面的普及, 键盘输入减少, 严重影响了利用击键动力学的认证效果。相比击键行为, 利用鼠标行为的身份认证更有优势。首先, 监控鼠标不需记录用户的敏感信息, 减小了用户的隐私威胁^[6]; 其次, 鼠标结构更简单, 不易受到尺寸和布局的影响^[7]。

2 相关工作

基于鼠标动力学的身份认证最早是由 Ahmed 在 2003 年提出的。Ahmed 等^[8]利用鼠标移动的速度、移动的距离实现了主机用户身份的认证。Pusara 等^[7]采集用户在浏览同一网页时的鼠标行为, 采用决策树算法对用户的 18 个行为习惯进行了研究。Gamboa 等^[9]邀请了 50 个志愿者参与玩一种测试记忆力的游戏, 提取鼠标移动轨迹的速度、角速度等特征, 采集 90 s 的鼠标数据, 得到的身份认证准确率是采用手写签名身份认证的 5 倍。

国内对于鼠标行为特征身份认证的研究比较晚。2008 年, Fang 等^[10]提出了一种利用鼠标动力学行为特征进行身份识别的方法, 利用神经网络分类算法, 得到了 0.48 % 的容错率 (FAR, false acceptance rate) 和 2.86 % 的拒真率 (FRR, false rejection rate)。2010 年, Shen 等^[11]从人机交互和生理行为层面对计算机用户的鼠标行为进行研究, 得到了 1.67% 的 FAR 和 3.68% 的 FRR。2013 年, Wang 等^[12]研究并实现了 3 种截获用户鼠标行为数据的方法, 最高达到了 94% 的认证准确率。2014 年, Wang 等^[13]结合鼠标行为和键盘行为进行身份认证, 得到了最高 96.6% 的认证准确率。

基于鼠标动力学的身份认证研究虽取得了一定的成果, 但这些方法有共同的限制。它们的认证时间平均需要几分钟甚至更长的时间, 认证需要收集足够多的鼠标行为数据。而在实际情况下, 内部攻击人员可能只需几秒钟 (如复制敏感信息, 植入病毒等) 即可完成攻击。

文献[1,4,14]提出了基于介入式场景的身份

认证方法, 有效地解决了身份认证的实时性问题。文献[1]提出了一种基于行为跟踪的身份认证系统, 提取用户在光标隐藏和固定两种异常场景下的鼠标行为, 认证时间仅需 5 s, 得到了最好 2.86% 的 FRR 和 3.23% 的 FAR。文献[4]通过随机地注入鼠标光标隐藏和光标固定两种场景进行身份认证, 分别得到了 3.9% 的 FAR、2.56% 的 FRR 和 4% 的 FAR、2.86% 的 FRR, 认证时间最快也是 5 s。文献[14]提出了基于鼠标隐藏场景的实时身份认证方法, 得到了 2.6% 的 FAR 和 3.3% 的 FRR, 而消耗的验证时间也只有几秒。

可以看出, 介入式场景的身份认证仅需采集几秒内的用户行为数据即可完成认证, 大大地缩短了认证时间。然而, 人的短时行为容易受到情绪、身体健康状况、应用场景等因素的影响而表现出不确定性^[4], 势必会影响介入式场景身份认证的准确率。

本文在介入式身份认证的基础上提出了固定场景的身份认证, 通过介入式动态场景身份认证和固定场景身份认证的结合, 有效地提高了身份认证的实时性和准确率。

3 全场景身份认证系统

3.1 全场景身份认证系统介绍

为解决企业由于内部员工采用身份伪装方式对他人主机带来的威胁问题, 本文提出了一种全场景身份认证系统。系统的流程如图 1 所示。若是新用户, 系统先分别进行动态场景和固定场景的离线训练, 否则, 便进行实时身份认证。系统进行身份认证的主要步骤如下。

1) 等待合适的时机构建注入异常事件, 以减小主机资源的消耗、保证用户的体验。

2) 构建光标异常认证场景, 包括鼠标光标的跳变、隐藏和呆滞 3 种异常场景。

3) 采集用户在异常场景发生时的鼠标行为数据, 提取特征向量。

4) 利用 SVM 方法, 根据已知模板对当前的用户进行分类, 判断是否为正常用户。若为正常, 等待进入动态场景环境; 若为异常, 判断当前是否是连续第二次动态场景异常, 若不是则继续跳

转到 2), 构建光标异常场景, 继续进行动态认证, 否则进入 5)。

5) 利用记忆游戏构建固定场景身份认证, 若为正常, 存储该特征向量, 达到一定数量后更新模板库; 若为异常, 则发出报警, 进行相应的处理。

每一次提取的鼠标行为特征向量, 根据身份认证的结果, 存储该特征向量, 达到一定数量后更新模板库, 适时更新模板库。

3.2 动态场景身份认证

3.2.1 介入式身份认证介绍

介入式身份认证, 是指用户在使用电脑的过程中, 由某一后台程序为得到用户的操作行为, 有目的地注入一些特定的异常场景, 从而诱导用户进入一种短时认证场景^[4]。目前, 介入式场景

主要是模拟光标短时间内失控, 如光标突然静止、消失、跳变、移动速度变慢、移动轨迹发生较大偏移等。在异常突然发生时, 用户所表现出来的往往是下意识的行为, 具有较高的个体差异性和辨识度。实验中, 本文采集了 2 个不同用户在不同的光标异常场景下的鼠标行为轨迹, 如图 2 所示, 实线和虚线分别代表用户 1 和用户 2 的鼠标轨迹。图 2(a)是在光标图标突然闪跳状态下, 2 位用户的鼠标轨迹; 图 2(b)是在光标突然消失状态下, 2 位用户的鼠标轨迹; 图 2(c)是在光标移动速度突然变慢的状态下, 2 位用户的鼠标轨迹。

3.2.2 动态场景身份认证

为了解决身份认证的实时性问题, 本文通过介入式的手段构建了动态身份认证场景。动态身份认证, 是指无需引导用户进入特定的应用, 直

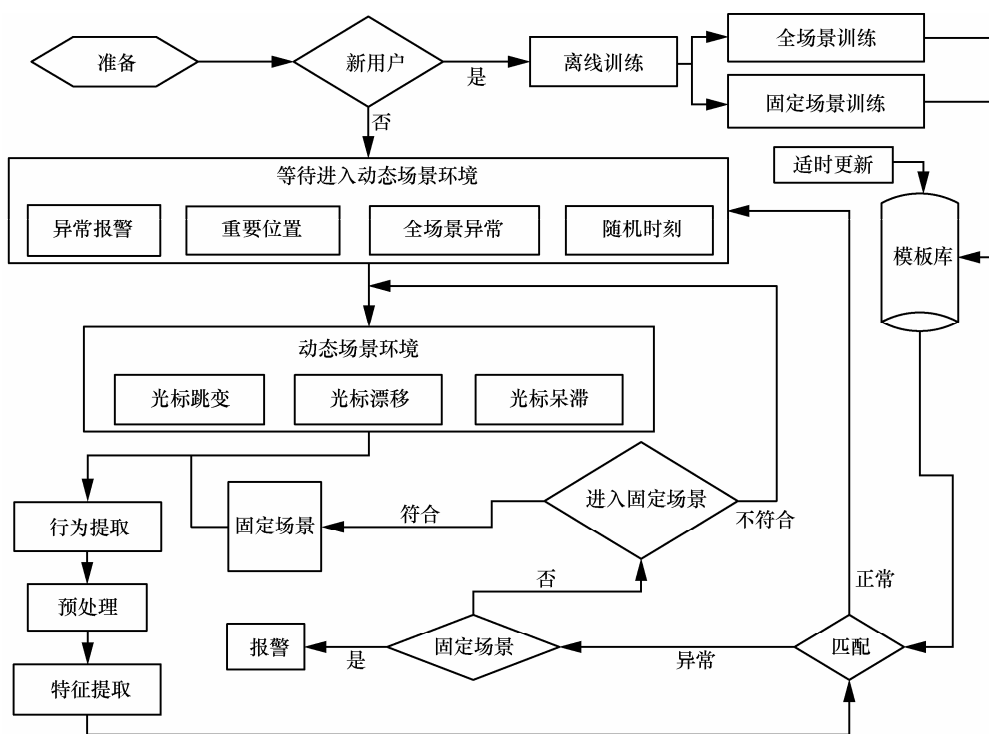


图1 全场景身份认证系统

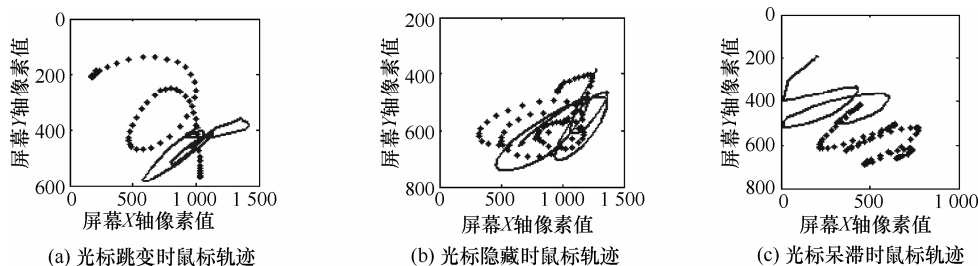


图2 不同用户在不同光标异常场景下的鼠标轨迹

接在用户使用主机的整个过程中,在适当的时候注入介入式事件,采集用户在介入式场景下的行为,进行实时身份认证。为了节省系统的资源,减少对用户体验的影响,构建异常场景时机包括4种,具体如下。

1) 异常报警,当主机的安全软件检测到可疑信息时,诱导用户进入异常场景。

2) 重要位置,如开启某些软件(如远程登录、邮件、文件传输等)。

3) 动态场景异常,由于动态认证采集的是短时行为,正常用户的某一次鼠标行为可能出现较大异常。若不是连续的动态认证异常,继续让用户进行另一种异常场景的身份认证。

4) 随机时刻,若当前身份认证正常,又不满足1)和2)的情况,则系统在一定的时间内随机注入异常场景,进行身份认证。本文后续的实验中选定时间为30 s。

动态场景的身份认证流程如下。

1) 当动态场景构建完成后,采集5 s内的用户鼠标行为数据,若当前用户没有操作鼠标,则继续构建动态异常场景。

2) 当获取到鼠标数据以后,对获取到的数据进行预处理,提取行为特征向量。

3) 采用SVM分类方法,根据已知的模板库对当前的特征向量进行模式分类。若为正常行为,继续等待注入动态异常场景的时机,否则,判断当前异常是否是连续第二次发生:若是第一次发生则进入动态场景构建,构建另一种异常行为,跳转到1)继续执行;若是连续第二次发生,则引导用户进入固定场景身份认证,同时保存当前的特征向量,判断如果保存的特征向量达到一定数量,则执行4)。

4) 对当前保存的特征向量进行建模,更新模板库。

3.3 固定场景身份认证

为了解决主客观因素对动态身份认证准确性的影响,本文提出了固定场景的身份认证系统。当连续两次动态身份认证均异常时,系统引导用户进行固定场景身份认证。固定场景身份认证,是指在用户使用主机的过程中,引导用户进入固定的应用,采集用户的行为数据进行身份认证。

本文设计了一款记忆性的拼图游戏作为固定场景的应用。与动态场景的身份认证相比,固定场景的身份认证应用固定,游戏带有记忆性,初始条件固定,因此,正常用户相比其他用户的行为会存在明显的差异。固定场景的身份认证流程如下。

1) 采集用户进行拼图游戏过程中的鼠标行为、时间戳等数据,直到拼图游戏完成。

2) 对采集到的数据进行适当的预处理,提取鼠标行为特征向量。

3) 数据SVM分类方法,利用已知的模板对提取的特征向量进行分类,若为异常,则进行身份报警。

4) 保存当前的特征向量,当达到一定数量时,生成相应的模板,更新模板库。

4 系统实现

4.1 场景的构建

4.1.1 动态场景的构建

动态场景的构建是通过Windows用户层API创建一个和当前屏幕大小相同的非模态透明窗口,然后设置该层的鼠标光标的属性。在这种情况下,用户移动鼠标,仍然可以通过钩子函数对鼠标行为数据进行抓取,但是用户却看不到鼠标的移动。动态异常场景分为3种,具体如下。

1) 光标隐藏:光标的隐藏是通过后台程序产生一个和当前屏幕大小相同的透明置顶窗口,设置该窗口的鼠标光标显示属性为FALSE。用户无法看到鼠标光标,但后台程序仍在记录鼠标的行为。若当前用户无鼠标行为,则保持光标隐藏;否则,退出光标隐藏场景。

光标隐藏场景的核心代码如下。

```
void cursorHide(){
    quitException=true;//光标异常场景保持标志,
    当用户产生鼠标反应时,退出异常场景
```

```
    HWND m_hwnd; //全屏后的窗口句柄
```

```
    LPCTSTR m_lpszClassName=AfxRegisterWnd
    Class(CS_HREDRAW|CS_VREDRAW, LoadCur-
    sorA(NULL,MAKEINTRESOURCE(IDC_CROSS)),
    (HBRUSH)::GetStockObject(0),NULL); //注册子
    窗口
```

```

int screen_x = GetSystemMetrics(SM_
CXSCREEN); //获取屏幕的宽度
int screen_y = GetSystemMetrics(SM_
CYSCREEN); //获取屏幕的高度
CRect rect(0,0,screen_x,screen_y); //创建
屏幕大小的矩形
m_hwnd=::CreateWindowEx(WS_EX_TOPM
OST, m_lpszClassName, T(""), WS_VISIBLE|WS_
POPUP, 0,0,rect.Width(),rect.Height(), NULL,0,NUL
L,NULL); //创建置顶子窗口
::SetWindowLong( hwndF, GWL_EXSTYLE,
GetWindowLong( hwndF, GWL_EXSTYLE) |
WS_EX_LAYERED); //设置子窗口的扩展属性
::SetLayeredWindowAttributes(m_hwnd,
0,1,2); //设置子窗口透明, 第 3 个参数即为透明度
while(quitExpection){
    ShowCursor(FALSE); //设置光标隐
藏属性
}
}

```

2) 光标跳变: 光标跳变场景是指让用户感觉鼠标的光标在不停地闪烁, 可以通过光标交替显示即可构建。核心代码同光标隐藏场景核心代码类似, 只需将 while 循环进行如下修改。

```

while(quitExpection){
    SYSTEMTIME sys;
    GetLocalTime( &sys ); //获取系统当前
时间
    long preTime= timeToMilliseconds(sys);
    //将当前时间转化为毫秒数
    long curTime= preTime;
    ShowCursor(FALSE); //设置光标隐藏
    while(curTime<preTime+35){ //光标隐藏
保持 35 ms
        GetLocalTime( &sys );
        curTime= timeToMilliseconds(sys);
    }
    ShowCursor(TRUE); //设置光标可见
    while(curTime<preTime+60){ //光标可见
保持 25 ms
        GetLocalTime( &sys );

```

```

curTime= timeToMilliseconds(sys);
}
}

```

3) 光标呆滞: 光标呆滞的场景是指用户在移动鼠标的过程中, 随机地让光标在当前位置稍作停留, 并将鼠标的光标形状改为缓冲的形状。调用的 API 函数如下。

```

HCURSOR hCur = LoadCursor(NULL, IDC_
WAIT); //设置新的光标, 参数 IDC_WAIT 表示光
标形状为沙漏形, 表示在缓冲

```

```

::SetCursor(hCur); //显示新的光标

```

```

SetCursorPos(positionX,positionY); // 将
光标定位到(positionX,positionY)

```

4.1.2 固定场景的构建

固定场景的构建主要是通过设定记忆性的拼图游戏, 如图 3 所示。进入游戏后, 通过 SetCursorPos(x, y) 函数将鼠标光标初始化到一个固定位置, 调用 ClipCursor(&rect) 函数将鼠标限制在拼图游戏的界面内。游戏中的图片由正常用户自己选取, 选定图片以后, 第一次开始训练时, 系统随机地将图片分成 8 小块, 以后每次进入游戏时, 各个小块的初始形状、位置和内容都保持不变, 直到更换新的图片进行新的训练。这样正常用户便能够熟练快速地完成认证。

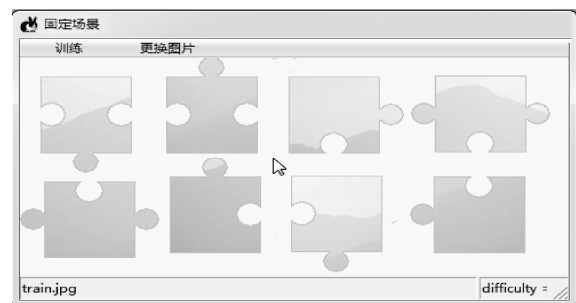


图 3 固定场景身份认证游戏界面

其中, 新用户可以通过软件左上角的“训练”按钮进行训练, 可以通过“更换图片”按钮选择自己熟悉的图片进行训练。对于动态认证异常的用户, 进入固定场景认证到完成认证期间, 这 2 个按钮均不可单击。待认证为合法用户以后, 才允许用户进行新的训练。

4.2 数据的获取

利用 Windows 系统钩子 WH_MOUSE_LL

来截获介入式场景发生时的鼠标行为数据，如图 4 所示，包括鼠标的事件类型参数；鼠标动作类型，其中，MOUSEMOVE 表示鼠标移动、LBTDN 表示左键按下、LBTUP 表示左键抬起、MOUSEWHELL 表示滚轮滚动、RBTDN 表示右键按下、RBTUP 表示右键抬起；像素坐标，其中，第一维是屏幕横向像素坐标，第二维是屏幕纵向像素坐标，系统采用的屏幕分辨率为 1366×768；鼠标所在窗口以及当前系统时间。

0x200	0x16F984	MOUSEMOVE	(1126,290)	AVPageView	57617941
0x201	0x16F984	LBDN	(1126,290)	AVPageView	57618393
0x202	0x16F984	LBTUP	(1126,290)	AVPageView	57618518
0x20A	0x16F984	MOUSEWHEEL	(1126,290)	AVPageView	57619907
0x20A	0x16F984	MOUSEWHEEL	(1126,290)	AVPageView	57620967
0x204	0x16F984	RBDN	(1126,290)	AVPageView	57621654
0x205	0x16F984	RBTUP	(1126,290)	middleKey	57621825

图 4 系统钩子提取的鼠标数据描述

4.3 特征的获取

4.3.1 动态场景特征选择

从图 2 中可以看出，在动态认证场景下，不同的用户在介入式场景发生时，鼠标移动的轨迹明显不同，因此，本文选择从鼠标的轨迹特征入手，提取不同鼠标轨迹的有效特征。

所选取的特征如表 1 所示。主要提取了轨迹

的移动方向、弯曲程度、移动距离以及移动速度 4 种特征，构成一个共 29 维的特征向量。其中，第 1、2 维描述移动鼠标的开始和结束方向；第 3~10 维是 8 个方向的移动次数统计，从 0~2 π ，每隔 $\frac{\pi}{4}$ 为一个方向区域；第 11、12 维描述轨迹拐角的曲率均值以及拐角的个数；文献[14]通过实验证明，用户移动鼠标时，鼠标在各个方向上的移动速度以及速度随着方向的变化趋势均存有差异，因此，第 13~29 维用以描述轨迹沿着 X 轴和 Y 轴移动距离和速度的均值、方差以及欧式距离。

4.3.2 轨迹拐点曲率计算

本文采用抛物线模型表征鼠标轨迹在拐角处的弯曲程度。该抛物线由拐角处的点作为顶点以及顶点前后的几个点拟合而成，模型如图 5 所示。曲线 $L_1: ax+by+c=0$ 表示抛物线的对称轴。根据 a 、 b 的值，抛物线可以分为两类，具体如下。

1) $ab \neq 0$ ，即抛物线的对称轴既不平行于标准的 X 轴也不平行于标准的 Y 轴。这种抛物线无法直接用标准方程表示，可采用基坐标系变化，如图 5(a)所示，变换后的基坐标系以抛物线的顶点为坐标原点，对称轴为 Y' 轴，开口方向为 Y' 轴正方向。抛物线在变换后的基坐标系中的方程为：

表 1 动态场景鼠标行为特征向量参数说明		
参数类型	参数描述	特征维数编号
鼠标移动方向	起始方向，结束方向，中间途经方向统计	1, 2, 3~10
鼠标轨迹曲率	轨迹转弯处的曲率均值，转弯个数	11, 12
鼠标移动距离	X、Y 方向均值，X、Y 方向方差，起止点距离	13、14、15、16、17
鼠标移动速度	X 方向速度均值、方差、极大值、极小值，Y 方向速度均值、方差、极大值、极小值，极值次数	18~21, 22~25, 26~29

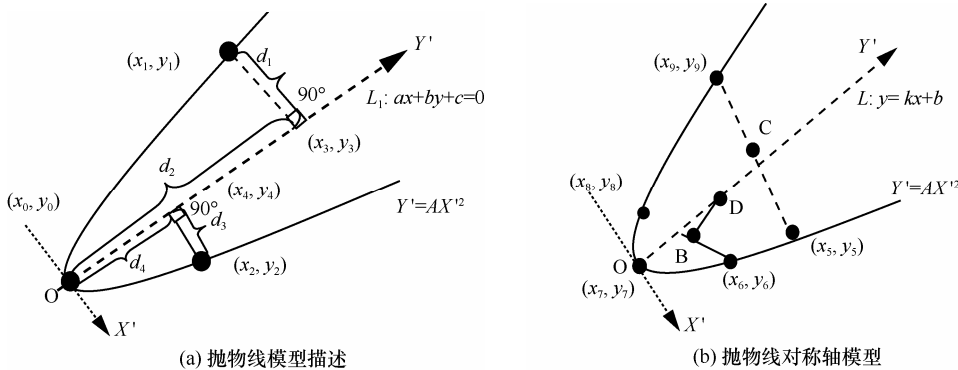


图 5 鼠标轨迹拐角曲率模型

$$y'=Ax^2.$$

由 (x_0, y_0) , (x_1, y_1) , (x_2, y_2) 构成的抛物线可以得到式(1)和式(2)

$$\begin{cases} d_1 = \frac{|ax_1 + by_1 + c|}{\sqrt{a^2 + b^2}} \\ d_2 = \sqrt{(x_3 - x_0)^2 + (y_3 - y_0)^2} \\ d_2 = Ad_1^2 \end{cases} \quad (1)$$

$$\begin{cases} d_3 = \frac{|ax_2 + by_2 + c|}{\sqrt{a^2 + b^2}} \\ d_4 = \sqrt{(x_4 - x_0)^2 + (y_4 - y_0)^2} \\ d_4 = Ad_3^2 \end{cases} \quad (2)$$

设 (x_1, y_1) , (x_3, y_3) 组成的直线方程为 $ex + fy + g = 0$, 则可得到式(3)

$$\begin{cases} \frac{ea}{fb} = -1 \\ ex_1 + fy_1 + g = 0 \\ ex_3 + fy_3 + g = 0 \end{cases} \quad (3)$$

同理设 (x_2, y_2) , (x_4, y_4) 组成的直线方程为 $hx + iy + j = 0$, 则可得到式(4)

$$\begin{cases} \frac{ha}{ib} = -1 \\ hx_2 + iy_2 + j = 0 \\ hx_4 + iy_4 + j = 0 \end{cases} \quad (4)$$

用 k 表示抛物线对称轴在原坐标系中的斜率, 即 $k = -\frac{a}{b}$, 综合式(1)~式(4)可得

$$\begin{cases} [(x_1 - x_0) + k(y_1 - y_0)]^2(1 + k^2) \\ = A^2[k(x_1 - x_0) - (y_1 - y_0)]^4 \\ [(x_2 - x_0) + k(y_2 - y_0)]^2(1 + k^2) \\ = A^2[k(x_2 - x_0) - (y_2 - y_0)]^4 \end{cases} \quad (5)$$

由式(5)可以得到

$$A = \frac{\sqrt{[(x_1 - x_0) + k(y_1 - y_0)]^2(1 + k^2)}}{[k(x_1 - x_0) - (y_1 - y_0)]^2} \quad (6)$$

从式(5)可以看出, 2个方程一共有2个未知

数 A 和 k , 求解 k 需要复杂的高阶运算, 消耗了系统资源。通过大量实验发现, 用户移动鼠标的轨迹在拐角处方向发生较大的变化, 拐点附近的轨迹像素点比较密集, 拐点前后的几对轨迹像素的中点分布在对称轴附近。因此, 为了简化 k 的计算, 可以通过接近抛物线顶点的几组像素点的中点规划出对称轴。

如图5(b)所示, 轨迹的拐点, 即抛物线的顶点为 O 点, B 点为点 (x_6, y_6) 和点 (x_8, y_8) 的中点, C 点为点 (x_5, y_5) 和点 (x_9, y_9) 的中点。 D 点为 B 点和 C 点的中点。抛物线的对称轴近似为直线 L_{OD} 。由此可知, 对称轴的斜率

$$k = \frac{y_5 + y_6 + y_8 + y_9 - 4y_7}{x_5 + x_6 + x_8 + x_9 - 4x_7} \quad (7)$$

将式(7)代入式(6)即可得到抛物线 $Y=AX^2$ 的 A 值, 由标准抛物线的 $y^2=2px$ 的曲率为 $\frac{1}{p}$ 可知, 对于规整化后的抛物线 $y'=Ax^2$ 的曲率即为 $\frac{A}{2}$ 。

2) $ab=0$, 即抛物线的对称轴平行于标准的 X 轴或 Y 轴。该类抛物线只需简单地平移即可转化为标准类型的抛物线, 很容易求出曲率。

4.3.3 固定场景特征选择

固定场景认证的实现如4.1.2节所示。可以看出, 动态场景认证时, 主要关注的是鼠标的移动行为。对于固定场景, 鼠标的单击动作比较频繁, 而且鼠标的移动范围很有限, 初始条件固定, 因此, 提取的特征可以更多, 顽健性更好。本文对于固定场景的数据特征向量在动态特征向量的基础上添加了20维, 如表2所示。

5 实验结果

5.1 特征分析

5.1.1 动态场景特征分析

从4.3.1节可以看出, 动态场景主要是提取用户移动鼠标的轨迹特征, 包括移动速度、移动方

表2

固定场景鼠标行为特征向量新增参数说明

参数类型	参数描述	特征维数编号
鼠标左键单击	单击次数, 单击到抬起的时间均值、方差	30,31~32
鼠标移动时间	完成整个拼图的时间	33
鼠标移动距离	图片每块的拖动时间及距离(每张图分为8块)	34~41,42~49

向、移动距离 3 方面的特征。本文随机选取了正常用户和其他用户近 50 次动态异常场景下的鼠标移动轨迹, 提取了其中的 4 维特征, 如图 6 所示。不带结点的折线表示正常用户的特征, 带圆形实结点的折线表示其他用户的特征。图 6(a)展示了动态场景下单次移动鼠标的欧式距离, 图 6(b)展示了动态场景下单次移动鼠标 X 方向上的平均距离, 图 6(c)表示了动态场景下单次移动鼠标的平均移动速度, 图 6(d)展示了动态场景下单次移动鼠标的轨迹在拐点处的曲率。

从图 6 可以看出, 正常用户移动鼠标提取的特征基本稳定, 而其他用户移动鼠标提取的特征个体间存在较大的差异。实验表明, 系统选取的特征能够在一定程度上区分不同的用户。

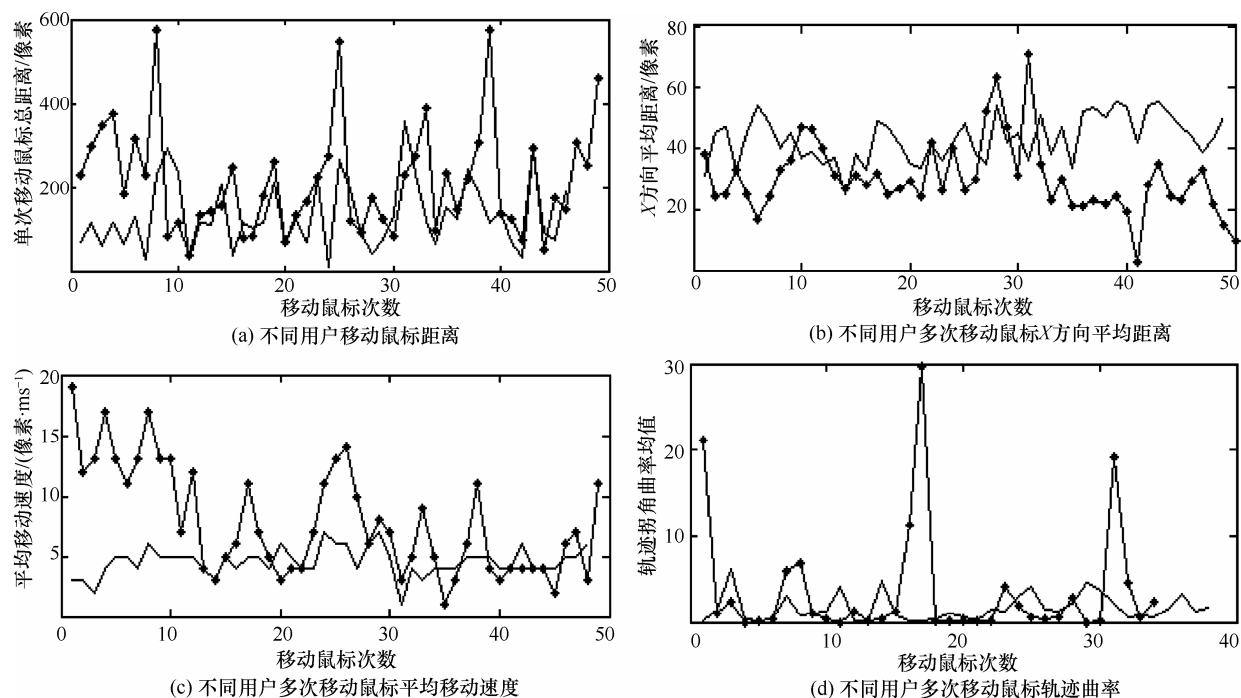


图6 动态身份认证场景提取特征对比

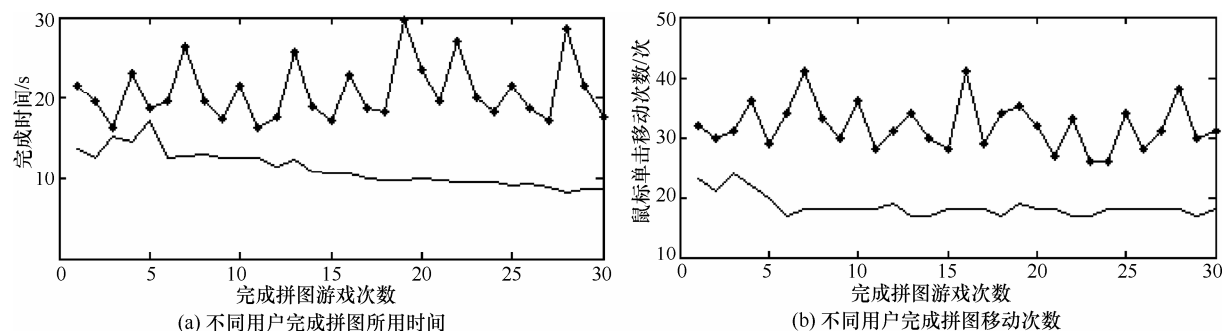


图7 固定场景身份认证不同用户鼠标行为特征

5.1.2 固定场景特征分析

固定场景身份认证主要提取的是鼠标移动拼图的特征。与动态场景相比, 固定场景身份认证使用的是正常用户所熟悉的拼图游戏, 如图 7 所示, 正常用户和其他用户的特征基本上没有交集, 由此可以很容易地区分正常用户与其他用户。无结点折线表示正常用户的特征, 带圆形结点的折线表示其他用户的行为特征。图 7(a)展示的是完成拼图游戏的总时间, 图 7(b)展示的是完成拼图鼠标单击图片总的移动次数。可以看出, 正常用户基本稳定在 10 s 以内即可完成, 平均移动次数基本稳定在 20 次以内。这里考虑到实际应用环境, 选取了其他 10 位用户, 共完成 30 次拼图, 每连续的 3 次是同一位用户完成的。可以看出,

表 3 全场景实时身份认证结果

用户	动态场景		固定场景		全场景		平均认证时间/s
	<i>FAR</i>	<i>FRR</i>	<i>FAR</i>	<i>FRR</i>	<i>FAR</i>	<i>FRR</i>	
用户 1	3.3%	4.6%	0.8%	1.7%	1.6%	2.3%	8.74
用户 2	3.0%	4.4%	1.1%	2.1%	2.3%	3.1%	7.23
用户 3	2.5%	3.9%	0.7%	2.3%	1.7%	2.7%	8.26
用户 4	2.6%	3.7%	0.7%	1.7%	1.5%	2.9%	7.67
用户 5	3.0%	3.9%	1.5%	2.7%	1.9%	3.1%	7.70
15 位用户平均	3.1%	4.2%	1.2%	2.4%	1.8%	3.0%	8.12

其他用户连续 3 次完成拼图游戏所用时间和移动次数都呈减小趋势，总体上比正常用户完成速度慢、移动次数更多。

5.2 用户身份认证结果

本文采集 15 位用户两周内的介入式场景下的鼠标数据，提取一共 436 730 条特征向量。每位用户训练数据将其他 14 位用户的鼠标行为作为异常数据。利用 LIBSVM 模式分类算法对用户进行分类。LIBSVM 是林智仁教授设计的开源软件包。LIBSVM 使用的数据格式为

<label><index1>:<value1><index2>:<value2>...

其中，<label> 是训练数据集的类别标识，在本文中有“+1”和“-1”两种取值，其中，“+1”是正常用户的标签，“-1”是异常用户的标签；<index> 的值是以 1 开始的整数，一般可看作是特征向量的维度；<value>值就是特征向量的特征值。

其中，5 位用户的认证结果以及这 15 位用户的平均认证结果如表 3 所示。可以看出，通过全场景的认证，平均的 *FRR* 为 3.0%，*FAR* 为 1.8%，而平均的认证时间仅为 8.12 s。

6 结束语

本文构建动态的介入式场景进行身份认证，不仅节省了系统的资源，而且大大地减少了身份认证的时间。当连续两次不同的动态场景身份认证出现异常时，利用记忆游戏构建固定场景进行身份认证，有效地克服了动态场景身份认证的短时行为容易受到主客观因素影响的弊端，增强了系统的顽健性。通过动态场景身份认证和固定场景身份认证的有效配合，既

保证了身份认证的实时性，又提高了认证结果的准确率。

参考文献：

[1] CHEN X J, SHI J Q, XU R, et al. PAITS: detect masquerader via short-lived interventional mouse dynamics[M]. Berlin: Springer, 2014: 231-242.

[2] CHEN X J, XU F, XU R, et al. A practical real-time authentication system with identity tracking based on mouse dynamics[C]// IEEE INFOCOM. c2014: 121-122.

[3] CHEN X J, FANG B X, TAN Q F, et al. Inferring attack intent of malicious insider based on probabilistic attack graph model[J]. Chinese Journal of Computers, 2014, 37(1): 62-72.

[4] 魏梓丞. 面向身份伪装的主机异常行为检测[D]. 北京: 北京邮电大学, 2014.

WEI Z C. Abnormal behavior detection of host computer for identity based intrusion detection[D]. Beijing: Beijing University of Posts and Telecommunications, 2014.

[5] 陈小军, 时金桥, 徐菲, 等. 面向内部威胁的最优安全策略算法研究[J]. 计算机研究与进展, 2014, 51(7):1565-1577.

CHEN X J, SHI J Q, XU F, et al. Research on the optimal security policy algorithm for internal threats[C]//Computer Research and Development, 2014, 51 (7): 1565-1577.

[6] ZHENG N, PALOSKI A, WANG H. An efficient user verification system via mouse movements[C]//ACM Conference on Computer & Communications Security. c2011: 139-150.

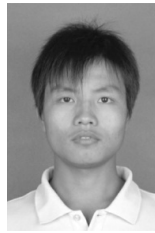
[7] PUSARA M, BRODLEY C E. User re-authentication via mouse movements[C]//Workshop on Visualization & Data Mining for Computer Security.c2004: 1-8.

[8] AHMED A, TRAORE I. Detecting computer Intrusions using behavioral biometrics[C]// The third Annual Conference on Privacy, Security and Trust. c2005: 91-98.

[9] GAMBOA H, FRED A. A behavioral biometric system based on human-computer interaction[C]// The International Society for Optical Engineering. c2004: 381-392.

- [10] 房超, 蔡忠闽, 沈超, 等. 基于鼠标动力学模型的用户身份认证与监控[J]. 西安交通大学学报, 2008, 10(40): 1235-1239.
FANG C, CAI Z M, SHEN C, et al. Authentication and monitoring of user identities based on mouse dynamics[J]. Journal of Xi'an Jiaotong University, 2008, 10(40): 1235-1239.
- [11] 沈超, 蔡忠闽, 管晓宏, 等. 基于鼠标行为特征的用户身份认证与监控[J]. 通信学报, 2010, 31(7): 68-75.
SHEN C, CAI Z M, GUAN X H, et al. User identity authentication and monitoring based on the behavior of the mouse[J]. Journal of Communication, 2010, 31(7): 68-75.
- [12] 王森, 蔡忠闽, 沈超等. 行为截获技术对鼠标动力学身份认证的影响[J]. 微电子学与计算机, 2013, 30(4): 14-21.
WANG S, CAI Z M, SHEN C, et al. The influence of the technology on the dynamic identity authentication of the mouse[J]. Microelectronics and Computer, 2013, 30(4): 14-21.
- [13] 王鑫, 王梦莹, 梁金. 基于鼠标键盘行为的身份认证[J]. 中国教育技术装备, 2014(22): 50-53.
WANG X, WANG M Y, LIANG J. Identity authentication based on the behavior of the mouse and keyboard[J]. China Educational Technology and Equipment, 2014(22): 50-53.
- [14] WEI Z, CHEN X, PU Y, et al. A real-time authentication method based on cursor-hidden scene[J]. International Workshop on Cloud Computing & Information Security, 2013, 52(1391): 346-350.

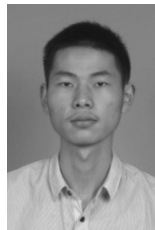
作者简介:



毛传武(1991-), 男, 湖北武汉人, 华中科技大学硕士生, 主要研究方向为网络信息安全。



程阳(1986-), 男, 湖北武汉人, 华中科技大学博士生, 主要研究方向为人工智能与模式识别。



余文明(1991-), 男, 湖北黄冈人, 华中科技大学硕士生, 主要研究方向为网络信息安全。