台灣科技大學

資　　訊　　工　　程　　系

博　士　論　文

用於資料防護之多面向使用者行爲剖析與身份學習辨識

Multi-Vector User Behavior Profiling and Learning to Classify Identity for Data Protection

研　究　生：吳建興
指導教授：李漢銘博士
共同指導教授：李育杰博士

中　華　民　國　一　〇　六　年　一　月

D10015006

博士學位論文指導教授推薦書

本校　資訊工程系　　吳建興(WU, JAIN-SHING)　君

所提之論文：

用於資料防護之多面向使用者行為剖析與身份學習辨識

係由本人指導撰述，同意提付審查。

指導教授：李漢銘
共同指導教授：李育杰

指導教授 [illegible]

105 年 12 月 29 日

博士學位考試委員審定書

D10015006
指導教授：李漢銘
共同指導教授：李育杰

本校　資訊工程系　　吳建興　君

所提之論文：

用於資料防護之多面向使用者行為剖析與身份學習辨識

經本委員會審定通過，特此證明。

學校考試委員會

委　　　員：
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
黃彥男
林永松

召　集　人：黃彥男

學　程　主　任：

系主任（所長）：[illegible]

中華民國　　　年　　　月　　　日

用於資料防護之多面向使用者行爲剖析與身份學習辨識

Multi-Vector User Behavior Profiling and Learning to Classify Identity for Data Protection

研　究　生：吳建興　　Student: Jain-Shing Wu

指 導 教 授：李漢銘博士　　Advisor: Dr. Hahn-Ming Lee

共同指導教授：李育杰博士　　Co-advisor:Dr. Yuh-Jye Lee

台 灣 科 技 大 學

資　訊　工　程　系

博 士 論 文

A Thesis

Submitted to Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University of Science and Technology

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science and Information

January 2017

Taipei, Taiwan

中華民國一〇六年一月

用於資料防護之多面向使用者行爲剖析與身份學習辨識

研究生：吳建興　　　　　　　　　指導教授：李漢銘博士
共同指導教授：李育杰博士

台灣科技大學資訊工程系

摘　　要

近年來企業資料外洩事件頻繁且衝擊更爲嚴重，物聯網時代更多的電腦主機、行動裝置及雲端系統更容易遭致來自全球的駭客攻擊威脅，隱私及機敏資料外洩的疑慮倍增。過去敏感資料保護技術多需進行檔案內容掃描過濾，非常倚賴對各種檔案格式的解析能力，對於無法解析之機敏檔案常無法有效保護。此外，文件在終端被編輯建立時，常缺乏製作者有效識別，不易判別文件機敏程度而同步進行控管，萬一身分被冒用或惡意內部人員進行文件資料竊取活動，更是如大海撈針不易發覺，致使機密文件被竊事件不斷發生。

本論文提出了一種以使用者多重行爲面向進行資料安全識別與控管機制，主要包括：(a)終端裝置之操作物理行爲辨識、(b)系統環境之意圖行爲分析。物理行爲辨識係擷取使用者在電腦上個人獨特的操作特徵習慣，包括文字鍵盤敲擊速率、滑鼠移動方向速度、以及手機觸控螢幕之按壓位置、滑移速度、按壓面積與壓力等各種連續行爲特徵數值，以SVM分類學習方法產生個別行爲模型，並建立連續性身份辨識方法，有效偵測僞冒竄改與非法存取。此外，針對意圖行爲分析，則以應用層活動日誌進行分析，涵蓋AD(Active Directory) 及Proxy 日誌資料等，用於偵測使用者異常存取資料之活動，透過Markov鏈建立使用者正常行爲樣態，以此爲基準用以偵測異常，可及早發現惡意使用者之意圖。

依上述藉由物理及意圖行爲識別使用者身份，可突破DLP無法解析未知格式檔案之限制。於操作物理行爲辨識方面，針對鍵盤及滑鼠操作行爲分析實驗，結果顯示識別之正確率爲92.64%，另就手機觸控行爲分析實驗更達到98.60%正確率；此外於系統環境意圖行爲分析，採用AD日誌實驗之異常偵測率達85.66%，整體而言，以人因行爲分析之方法確能支援安全識別與管控，有效降低資料外洩風險。未來結合更多種日誌分析，可延伸偵測金融詐騙及基礎設施攻擊等。

Multi-Vector User Behavior Profiling and Learning to Classify Identity for Data Protection

Student: Jain-Shing Wu

Advisor: Dr. Hahn-Ming Lee
Co-advisor: Dr. Yuh-Jye Lee

ABSTRACT

In the recent years, data leakage issues have become increasingly serious for companies and organizations. As the Internet of Things continues to expand, more computer software, connected devices, and cloud systems are vulnerable to attacks from anywhere in the world. Any incident of data leakage can cause severe financial loss or damaged reputations for individuals and corporations. This issue is becoming more explicit as people increasingly depend on smartphones in daily life and business. To protect sensitive data, traditional solutions such as content scanning and file filtering are commonly used, but these depend on the capability of parsing various file formats. For unsupported file formats, this type of approach is ineffective and the risk of data breach persists. Further, these methods are unable to discover malicious tampering by unauthorized users during data creation and modification. Even for authorized users, potential risks exist. It often occurs that a privileged account may be compromised, or a database may be accessed by a malicious employee. Detecting this type of malicious behavior in normal privileged activities is like finding a needle in a haystack.

This thesis proposes an user behavior-based approach with multi-vector profiling, which can be regarded as a complimentary solution to the state-of-the-art, content-based data loss prevention (DLP) model. The proposed approach includes two types of behavior analysis to actively identify data creators and malicious activity. The first type is related to the physical behavior of the user on the endpoint device, such as keystrokes, mouse movement, or touch-screen gestures. Support vector machine (SVM) classification and machine learning methods generate a behavior model and identify the user. The second type is focused on analyzing user intent through user behavior in the system environment, including account access, privilege escalation, and web browsing. Using a Markov chain with AD (Active Directory) and proxy logs, this analysis can model user behavior patterns to detect malicious activity.

As mentioned above, by using physical and intention behavior analysis to recognize user identity, the proposed approach can resolve the limitation of parsing an unknown format. For analyzing physical behavior, the experiment on keystroke and mouse movement operations is 92.64% accurate, while the experiment on touch-screen gesture is 98.60% accurate. Additionally, for analyzing system intension behavior, the experiment with AD logs yields an anomaly detection rate of up to 85.66%. In summary, the conceived approach based on user behavior analysis can help protect data by effectively identifying data creators and malicious activity. Ultimately, it can also contribute to mitigating the risk of data leakage. In the future, the proposed approach can be widely used in financial fraud detection, infrastructure threat detection, and other areas of security vulnerability.

誌　　　　謝

首先我要感謝我的太太一路支持，讓我在繁忙工作之餘，能無後顧之憂地持續進行研究。其次感謝共同指導教授李育杰博士及指導教授李漢銘博士的多年協助，由於其專業的指導與方向指引，使得我在研究上持續精進，尤其感謝李育杰博士在Keystroke、Mouse及Touch Panel之行爲數據分析上，提供非常有價值的指導意見，研究工作得以順利進行並於國際學術期刊發表論文；此外，也非常感謝我工作上的同仁謝志宏、魏得恩，於企業內用户異常行爲分析上一起討論，以及林志達、鍾松剛等夥伴協助進行相關實驗，特此也感謝資策會各級長官的鼓勵與支持，使我得以於工作中同時進修成長，最後要感謝所有口試委員特地撥冗指導。

Table of Contents

1、 INTRODUCTION 1

1.1 Background and Motivation 1

1.2 Challenges 2

1.2.1 File Format Parsing 2

1.2.2 Data Creator Identification at an Endpoint Devicee 3

1.2.3 Stealthy Activity Discovery in a System Environment 5

1.3 The Proposed Approach 5

1.3.1 Physical Behavior Analysis on an Endpoint Device 6

1.3.2 Intention Behavior Analysis in System Environment 7

2、 RELATED WORK 9

2.1 Data Loss Prevention Techniques 9

2.2 Input Behavior Profiling of Touch Panel 10

2.3 Abnormal Behavior Detection Techniques 12

3、 METHODOLOGY 16

3.1 Overview 16

3.2 Keystroke Behavior Learning 18

3.2.1 Recording and Extracting 19

3.2.2 Training and Testing 19

3.3 Mouse Movement Behavior Learning 21

3.4 Smartdevice Touch-Panel Behavior Learning 24

3.5 Abnormal User Behavior Analysis 26

3.5.1 Windows Active Directory Domain Service 27

3.5.2 Proxy Service 27

3.5.3 The Real Dataset 28

3.5.4 The Proposed Method - *ChainSpot* 28

3.5.5 State Constitution of Used Markov Chain 29

3.5.6 Build a Markov Chain Given a Log Sequences Dataset 31

3.5.7 Deviation Estimation Given Different Markov Chains 32

4、 EXPERIMENTS 33

4.1 Framework Implementation 33

4.1.1 Secure Keystream Analyzer (SKA) 34

4.1.2 DLP System 35

4.2 Keystroke Behavior Experiments 36

4.3 Mouse Movement Behavior Experiments 38

4.4 Practical Keystroke and Mouse Behavior Experiments 40

4.4.1 Diverse Style Keystroke Experiment 40

4.4.2 Combining Keystroke and Mouse Movement Profiling Findings 42

4.5 Smartdevice Touch Panel Behavior Experiments 46

4.6 Discussion on Content-Based and Behavior-Based DLP Models 52

4.7 Abnormal User Behavior Experiments 53

4.7.1 Effectiveness of Measuring Behavior Deviation Using *ChainSpot* 54

4.7.2 Performance of Anomaly Detection Using *ChainSpot* 58

4.7.3 Representative Demonstration and Case Study of *ChainSpot* . 60

5、 CONCLUSIONS 65

REFERENCES 68

Appendix A: Example for Behavior Log File 75

List of Tables

3.1 Character type ID and characters 20

3.2 A data sample of character time cost for an instance 22

3.3 Data sample for the characteristic time-cost of an instance 25

3.4 Number of different SOC tickets 29

4.1 The information of experiment instances 37

4.2 A data sample of character time cost for an instance 37

4.3 The number of extracted EMMS of each user ID 40

4.4 Results for various user measures in the first SVM test 43

4.5 Additional user measure results in ten SVM tests 44

4.6 The comparison of various average accuracy results of the experiment 46

4.7 Number of extracted events from each user 47

4.8 Results from various user measures in the ninth SVM test 48

4.9 Further measurement results for families in the tenth SVM test of the unigram dataset 49

4.10 Various averages in terms of accuracy 50

4.11 Comparison between content-based and behavior-based DLP model . 52

4.12 Reconnaissance survey for effectiveness of *ChainSpot* 55

4.13 The major changes in quantities when applying probability ratio filtering strategy 56

4.14 The major changes on quantities when applying entropy filtering strategy 57
4.15 Performance of anomaly detection under different deviation thresholds 60
4.16 Performance comparison between *ChainSpot* and three baselines . . . 60

List of Figures

3.1 The concept of the proposed DLP model 17

4.1 Keystroke tracking 34

4.2 StreamB in log_newfileB.xml 35

4.3 The detailed results of CMS 36

4.4 The average testing results of ten runs for each user. 39

4.5 The average testing results of ten runs for each user in the mouse behavior experiment. 41

4.6 The average testing results of ten runs for each user 45

4.7 Average confusion results from the ten runs for each user 49

4.8 Average identification accuracy for each user 50

4.9 Average rejection ratio for each user 51

4.10 Effectiveness changing given different filtering thresholds 57

4.11 Cost curve describing distributions of normalized expected cost under different δ and probability costs 61

4.12 Patterns of user A_{case1} who participated in “Multiple accounts logon from single IP in short time” event 62

4.13 Access records extracted from abnormal-phase Markov chain of account A_{case2} who participated in “Host connects to malicious domain” event 64

4.14 Access records extracted from Markov chains of account A_{case2} who participated in “Host connects to malicious domain” event 64

Terminologies and Symbols

T	Switching time cost of character type.
$T_{i,j}$	The switching time cost for inputting characters from the i-th type to the jth type.
SVM	The support vector machine algorithm.
κ	The Gaussian radial basis function.
C	The penalty parameter (cost parameters) of SVM.
γ	The kernel parameters of the Gaussian radial basis function.
x, y	The horizontal and vertical position of the mouse on the screen.
t	The current universal time value.
P_n	Data of a mouse movement record.
$\overrightarrow{P_n}$	The mouse movement vector
θ	The direction angle of a movement vector.
d_n	The move distance of a $\overrightarrow{P_n}$, from recorded point P_{n-1} to P_n.
k	Number of consecutive data records (P_n).
R_n	Move Direction Identity $n, n = 1\ 8$.
EMMS	A subset of P that consists of k consecutive data records (P_n) with the same R_n, and that satisfies the conditions.
δ_1	The threshold for effective mouse move distance.
δ_2	The threshold of consecutive data records.
P_{start}	The start point of a **EMMS**.
P_{end}	The end point of a **EMMS**.

$trianglet_n$	The movement time cost from recorded point P_{n-1} to P_n.
v_n	The mouse moving velocity at P_n.
a_n	The mouse moving acceleration at P_n.
A	The smooth fit data by three-degree polynomial curve of a a_n set.
A_{max}	The maximum value of acceleration.
A_{min}	The minimum value of acceleration.
$T_{A=0}$	The zero acceleration time point from maximum to minimum.
j	The direction ID.
k	The degree of distance.
l	The degree level of a feature, $1 \leq l \leq 5$.
A	The TOUCH_MAJOR area.
p	The touch pressure at a certain point.
$f_{i,n}$	The feature value with respect to the maximum value $f_{i,max}$ of the i^{th} feature among all users.
$f_{i,max}$	The maximum value of the i^{th} feature among all users.
$r(l)$	The distribution ratio set of a feature, between 0 and 1, and the summation of $r(l)$ is 1.
N	The total number of data.
$z_n(x, y)$	The x, y position of touch panel point n.
$\vec{z_n}$	The moving vector from z_n to end.
v_n	The touch panel sliding moving velocity at z_n.
vp_n	The pressure-release velocity at z_n
TP	The number of true positives

FN	The number of false negatives
FP	The number of false positives
TN	The number of true negatives
A	Accuracy, $= (TP + TN)/(TP + FN + FP + TN)$
P	Precision (Sensitivity) , $= TP/(TP + FP), recall R = TP/(TP + FN)$
S	Specificity $= TN/(TN + FP)$
N	Negative predictive value ,$= TN/(TN + FN)$
F	F-measure ,$= 2PR/(P + R)$

Chapter 1 INTRODUCTION

1.1 Background and Motivation

Data leakage is a serious issue for organizations and individuals, since sensitive data disclosed to unauthorized personnel could cause serious damage. This issue is particularly severe for business organizations because a single data leakage incident can result in the loss of customer loyalty, unanticipated lawsuits, costs related to the compensation of affected parties, and other negative consequences [1,2].

Symantec reported that more than 232.4 million identities were stolen in 2011 [3]. A data breach investigation report by Verizon revealed that 174 million data records had been compromised through 855 data breach incidents in 2011 [4]. According to statistics released by DataLossDB [5], 1,646 data leakage incidents were reported worldwide in 2012, a far higher number than in the past. In 2013, 1,459 data leakage incidents occurred.

Furthermore, data loss risks have been increasing dramatically with the proliferation of mobile devices, removable devices, and ubiquitous Internet access [6]. Over the past decade, smartphones have become a crucial part of our daily life. They have also become more affordable and powerful, and they are usually equipped with 3G and 4G networking [7]. According to a market report, global sales of smartphones to end users totaled 1,423 million units in 2015, a 12.6 percent increase over the same period in 2014 [8]. Given the convenience and the portability of using smartphones, users are likely to store sensitive and private data (*e.g.*, passwords, credit-card numbers, and photos) on their smartphones, just as they will undoubtedly use their phones for some security-sensitive tasks (*e.g.*, sending classified documents and making authorized financial transactions). This makes smartphones an attractive target for hackers. For instance, if an attacker knows a user's PIN or screen lock code, the attacker can exploit the phone to steal sensitive data or conduct illegal activities.

To prevent data leakage in hosts and smartphones, most solutions are focused on sensitive data discovery in files. These are a form of content-aware solutions [22], which adopt a deep content inspection approach. In order to check if there is any sensitive data in file contents, these solutions need to parse the suspicious file and extract text streams for examination. The text streams can be temporarily stored in memory or saved as a file in storage. By matching text streams with predefined patterns or keywords, sensitive data in file contents can be identified. The file containing sensitive data can then be tagged, deleted, quarantined, encrypted, or moved to a safe place for further management.

1.2 Challenges

Although the content-based DLP model has been widely used for many years, it can be ineffective due to some technological limitations. In general, a DLP solution needs to understand various file formats so that it can extract and examine text streams in file content. However, the inability to parse unknown file formats will cause this solution to fail. It is also unable to discover malicious tampering against unauthorized users during data creation and modification. Even for authorized users, potential risks still exist if the privileged account is compromised or if the database is accessed by a malicious employee. To detect this type of malicious behavior in normal privileged activities is like finding a needle in a haystack. The details of the three major challenges are described in the following sections.

1.2.1 File Format Parsing

State-of-the-art DLP systems use regular expressions, statistical pattern matching, keyword comparison, and document fingerprinting to discover sensitive data [9]-[10]. To achieve data loss prevention (DLP) in hosts and smartphones, DLP systems need to be capable of understanding various file formats so that contents of files can be extracted and examined. The file decoding involved in this is significantly complicated because of various file structure designs (*e.g.*, sequential, inverted, or index-sequential) and different character encodings *etc.*) and different character encodings (*e.g.*, Unicode, ASCII, Big5, or GB2312). For example, parsing a Microsoft Excel document requires a precise understanding of how the file expresses and separates each data value. However, ac- cording to the binary file format published by Microsoft for Office [11], the structure of the Excel 97-2003 file format is described in a document with 1,183 pages. It is therefore too complicated to implement a corresponding parser for DLP systems.

Because file formats can be proprietary (*e.g.*, Microsoft Office documents), open source (*e.g.*, HTML, Office Open XML, or CSS), or even unpublished, the implementation of parsers is challenging and burdensome. Sometimes, files need to be reverse engineered when the file format is unspecified [12]. To protect DaR (Data at Rest), current DLP systems focus on the number of file formats that they can decode. For instance, RSA announced that its DLP suite can parse over 300 file formats, whereas McAfee claimed that its DLP solution can parse more than 390 file formats. Solutions developed by Websense can decode more than 400 file formats. Clearly, all data security companies aim to cover as many file formats as they can, and thus suffer an ever-increasing burden of implementation.

The rapid growth of modern applications makes the situation worse. Emerging applications may use new file formats to target different domains for usage purposes, thus creating new challenges for DLP systems in file format decoding. As a result, new and unsupported file formats may cause data breaches. To avoid this persistent challenge, there is a need for a solution that determines the content inside files without parsing them.

1.2.2 Data Creator Identification at an Endpoint Devicee

In addition to file content inspection, another challenge for DLP systems is to recognize the identity of the data creators of a file. A file may contain data created by different users. Current solutions do not keep records of which individuals are involved in data creation. Certain types of data must be carefully handled, especially data created by senior officials in an organization, e.g., the supervisor, section manager, or general manager. Such files may contain data such as strategic organizational policies that are more sensitive than data created by ordinary employees. Organizations therefore need to know the level and the means of protecting such data. The problem of identifying the data creator can be solved using common plagiarism detection methods [13]- [14]. However, such methods focus on identifying the plagiarized content from a given source (e.g., research papers or books) or determining the intentional modification of words or sentence structure without changing the content [13]. These solutions involve file-level cross-reference identification and cannot identify the specific author of material in a file, especially when there are multiple authors of the same file. The ability to identify the data creator of a file has two main advantages: (1) the ability to investigate and assign responsibility, and (2) the capability of refining the sensitivity level of the file according to the identity of its data creators. Current DLP solutions lack data creator identification ability, *i.e.* they can only recognize the creator of a file through its metadata, but cannot identify an author who contributes data to the file (the data creator). Consequently, the level of confidentiality of a file cannot be determined with high granularity, and current DLP solutions can fail to prevent unauthorized access to critical data [15]. Therefore, the risk of malicious or accidental leakage of data increases. [16] proposed a general approach for user verification based on user trajectory inputs. It employed a Markov chain model with Gaussian distribution in its transitions to describe the behavior in the trajectory.

In the case of smartphones, most now have a touch sensor; 800 million mobile phones were expected to be touch-enabled by 2014 [17], [18]). Recently, considerable research has focused on this feature, such as touch analytics [19]. Thus, it is believed that building a continuous authentication mechanism based on Keystroke Dynamics will be pertinent to touchscreen smartphones. Some previous works have adopted continuous and passive authentication mechanisms based on a user's touch operations on the touchscreen. In the field of mobile technology, continuous and passive authentication is a popular scheme to secretly classify users' identities based on their peculiar touch motions, namely keystrokes and gestures. In addition, a user's touch motions on mobile screens can be utilized for continuous and passive authentication. This is a convenient scheme because it does not require additional hardware or extra user attention. However, some researchers claim that touch analytics should be combined with other modalities—for example, the user's location, accelerometer data, images from the front-facing camera, and application usage patterns. Indeed, these additional modalities improve the accuracy of the authentication. However, these approaches are insufficient for classifying users with singular touch motions.

1.2.3 Stealthy Activity Discovery in a System Environment

Detecting malicious access behavior of a file inside an enterprise is like finding a needle in a haystack. An advanced persistent threat (APT) is a set of stealthy and continuous computer hacking processes. APT attacks involve a high degree of covertness over a long period of time, using malware with sophisticated penetrating techniques to hack target systems. According to surveys of well-known security institutes and companies [20] [21], only 31% victims have the ability to discover an inside attack or data breach by malicious employees. Once APT malware has successfully compromised the system, it might use legal privilege to access or tamper with sensitive data, or stealthily send it outside the organization. There are many commercial solutions for preventing APT attacks on email servers or end-point hosts. However, they are usually ineffective for detecting malicious users who have privileges at a post-compromised stage. One approach to resolve this issue is to trace and analyze user access logs to determine if there is any existing abnormal activity.

1.3 The Proposed Approach

To overcome the challenges mentioned above, this thesis proposes a user behavior-based approach, which can be regarded as a complimentary solution to state-of-the-art content-based models (*e.g.*, DLP solutions of Websense, RSA, or

McAfee). The proposed model includes two types of behavior analysis for actively identifying the data creator and malicious activity. The first type is related to a user's physical behavior on the endpoint device, such as keystrokes, mouse movement, or touch-screen gestures. The second type is related to behavior showing user intent in a system environment, such as account access, privilege escalation, and web browsing. The details of this approach are described in the following sections.

1.3.1 Physical Behavior Analysis on an Endpoint Device

Tracking and analyzing a user's keystroke behavior while he or she types text into a file improves the content-based model, which uses this information to refine the actual content that the user has entered. To further identify content generated by different data creators, the model uses keystrokes, mouse movement, and touch-screen gesture behavior to learn and profile data creator behavior. A novel threat detection framework was proposed to build graphical patterns by summarizing a user's sequential behaviors while using application-layer services, and to discover deviations from a user's normal patterns. With the proposed model, the current content-based model will benefit from the following advantages:

1 Sensitive content can be discovered without decoding a file, hence eliminating the need to build a new file parser for each new format.

2 Data creators can be identified by analyzing their keystroke and mouse movement behavior, so that the sensitivity level of the file in question can be assessed according to the identities of its creators.

3 The level of confidentiality of a file can be determined immediately after it has been created, thus reducing the time that sensitive data is unattended.

This thesis also proposes a novel continuous authentication method. The method not only profiles behavioral biometrics using keystrokes and mouse gestures, it also acquires the specific characteristics of one-touch motions during the user's interaction with a smartphone. This focuses on the manner in which a user interacts with the touchscreen—that is, the specific location touched on the screen, the drift of the finger when moved in all directions, the area size that is touched, and the pressure applied to the screen. Together, these characteristics reflect the user's unique physical and behavioral biometrics. Furthermore, this thesis also defines the speed of the Gesture Segment (GS) in order to extract a meaningful velocity segment. Finally, this thesis includes experiments conducted to evaluate the proposed method and demonstrate the accuracy and effectiveness of authentication. The contributions are summarized as follows:

1. Propose a novel extraction algorithm for touch-screen operations that can be used to identify users.
2. Combine the extracted pointing and sliding features to build an SVM classifier. To evaluate the effectiveness, this thesis conducted an experiment comprising 150 instances from 10 users.
3. Get an average accuracy (*i.e.*, the average micro-precision) for the ten-run experiments of 98.6

1.3.2 Intention Behavior Analysis in System Environment

Ultimately, the cyber threat detection framework *ChainSpot* was proposed. It can discover unaware intrusions and anomalies based on a service log analysis. The novelty of this approach is its ability to build graphical patterns by summarizing a user's sequential behaviors while using application-layer services, and to discover deviations from a user's normal patterns. In addition to modeling the behavior, the issue of justifying the trade-off between feature explicity and computation complexity is properly addressed. The effectiveness and performance of the proposed method are evaluated by using a dataset collected in a real case. The experiment outcome shows that *ChainSpot* is an excellent solution for detecting unaware abnormal behaviors that can bypass IDS (Intrusion Detection System) easily. The detection results are highly correlated to expert-labeled ground truth, so *ChainSpot* is proven to be helpful in significantly saving forensic efforts. Moreover, case investigations demonstrate that the differences between benign and suspicious patterns can be further interpreted to reconstruct the attack scenarios. The findings may then be regarded as critical indicators for intrusion detection and as tracing points for security forensics.

The rest of this thesis is structured as follows. Chapter 2 briefly presents related works in areas such as data loss prevention, input behavior profiling of touch panels, and abnormal behavior detection. In Chapter 3, the proposed methodologies are described in detail. Section 3.1 focuses on the proposed model and the implementation of its framework. Sections 3.3 and 3.4 mainly present the keystroke and mouse movement behavior learning methods to identify data creators. Section 3.5 introduces a novel threat detection framework, *ChainSpot*, which focuses on both application-layer AD and proxy sequential log analysis. Section 4.1 demonstrates the practical experiments of the proposed model. Section 4.3 presents a comparison between the proposed model and other content-based DLP models. The experiments in section 4.7 show that *ChainSpot* can provide excellent support for discovering abnormal behaviors, which is the starting point for early threat detection. Finally, the conclusions of this paper are presented in Chapter 5.

Chapter 2 RELATED WORK

In this chapter, several related works on multi-vector input behavior profiling techniques will be mentioned and discussed. The multi-vector input behavior profiling includes three categories of devices: keyboard, mouse and touch panel. In the remaining sections, these three types of input behavior profiling will be discussed, as well as methods to learn techniques for classifying identity for data protection. File access behavior to identify abnormal user will also be discussed.

2.1 Data Loss Prevention Techniques

Data leakage is a serious security issue in which sensitive data is disclosed to unauthorized personnel, either maliciously or inadvertently. Several data loss prevention (DLP) systems have been developed to discover, monitor, and protect data through deep content inspection. As DLPs focus on discovering sensitive data in files, they are classified as content-aware DLP systems [22]. There are many commercially available content-aware DLP systems. For example, the data security suite released by Websense includes three modules – Data Security Gateway, Data Discover, and Data Endpoint – to analyze sensitive data using a variety of techniques. RSA has developed a DLP suite to protect data in data centers, on networks, and at endpoints [23]. McAfee's Total Protection for DLP contains several modules to ensure safe data handling, e.g., DLP Discover, DLP Monitor, and DLP Endpoint [24]. The company amXecure has developed a DLP tool called PrivacyID to identify sensitive content in files [25]. Palo Alto Networks has also announced a next generation firewall with DLP functionality to detect critical personally identifiable information (PII), such as social security numbers (SSNs) or credit card numbers [26].

The above content-based DLP model has two main issues. First, to determine whether a file is sensitive or not, the model needs to understand various file formats so that contents of files can be extracted and examined. Second, it can only identify the data creators (such as the author and latest modifier) who are recorded in metadata of a file. It cannot provide the information about the data creators of each sentence, because most metadata does not record such information. Thus, the model cannot determine or adjust the confidentiality of a file according to the identity of the data creators. These two issues are described in detail in the following sections.

2.2 Input Behavior Profiling of Touch Panel

Owing to the increasing popularity of touchscreen smartphones, considerable research focusing on the study of biometric authentication has been dedicated to this platform. Biometric authentication refers to the use of human characteristics to label and identify individuals, and it is widely used to authenticate smartphone users. In 2014, a survey [28] divided biometric user-authentication techniques for mobile phones into two categories: physiological and behavioral biometrics. The survey introduced 11 biometric authentication techniques and analyzed the feasibility of their deployment on touch-enabled mobile phones. Physiological biometrics are based on a person's relatively unchanging physical characteristics, such as fingerprints, facial features, the iris or retina, and the hand or palm. Behavioral biometrics, on the other hand, refer to a particular behavioral trait of an individual, such as the individual's voice, signature, gait, behavioral profile, or typing rhythm (also called keystroke dynamics).

Recent years have witnessed an increasing number of studies dealing with keystroke and touch dynamics on Android smartphones. Keystroke dynamics are a popular area of study, and many proposals have been presented for authenticating users on mobile devices using this approach. Given the popularity of touchscreens, touch dynamics have also become a popular area of research. The term touch dynamics refers to the collection of detailed information about individual touches, including duration, movement, multi-touch, scrolling, tapping, flicking, and rotation of the touchscreen. Both touch and keystroke dynamics are commonly used for continuous authentication on smartphones, and they do not require any special hardware [28].

Recent research has applied keystroke dynamics and touch dynamics on smartphones with physical keys. For example, Antal *et al.* [29] examined the performance of touchscreen-based features with keystroke dynamics. They collected 42 user actions on Android devices with touchscreens and used random forests, Bayesian nets, and SVMs to identify users. Their results showed that these additional features enhanced the accuracy of both processes. TouchLogger [30] is an Android application that extracts features from device-orientation data to infer which keys are being pressed. TouchLogger uses motion as a side channel to infer keystrokes on a smartphone's soft keyboard. An evaluation showed that TouchLogger could correctly predict more than 70% of the keystrokes on a numbers-only soft keyboard. Buschek *et al.* [31] studied mobile keystroke biometrics by comparing touch-specific features of three different hand postures and evaluation schemes. They analyzed 20,160 password entries from 28 participants over two weeks, and showed that the inclusion of spatial touch features reduced the implicit authentication equal-error rate (EER) by 26.4-36.8% relative to previously used temporal features. Frank *et al.* [19] selected a set of 30 behavioral touch features that could be extracted from raw touchscreen logs, and trained user profiles based on vertical and horizontal strokes, using a k-nearest neighbor classifier and a Gaussian RBF kernel SVM. The results showed that the classifier had an EER between 0% and 4%. Draffin *et al.* [32] defined the micro-behavior of mobile users' interaction with their devices' soft keyboards. This micro-behavior includes the drift of a finger moving up or down, the force of the touch, and the area being pressed. They demonstrated that these micro- behavioral features can identify a non-authorized user within 5 keypresses 67.7% of the time, with a False Acceptance Rate (FAR) of 32.3% and a False Rejection Rate (FRR) of only 4.6%. Moreover, the detection rate after 15 keypresses was 86% with a FAR of 14% and a FRR of only 2.2%. Xu *et al.* [33] also adopted a continuous and passive authentication mechanism based on the touch operations of a user on a touchscreen, pointing out that touch operations such as keystrokes, slides, pinches, and handwriting can be used to continuously authenticate users. Li *et al.* [34] proposed a biometric-based system for smartphones to re-authenticate the current user's identity based on their finger and touch movements, without interrupting the user's smartphone interactions. They selected eight features that combine sliding and tapping behavior, and verified their method in an experiment comprising 75 participants. The results indicated that their proposal could achieve an FAR of 4% and an FRR of 4%. Finally, Meng *et al.* [28] developed a lightweight touch-dynamics-based authentication scheme with eight touch-gesture-related features. They also designed an adaptive mechanism [35] that alternates between classifiers to maintain accuracy when authenticating users.

Our work also aims at continuous user authentication. Our proposed method relies on both keystroke and touch dynamics. Moreover, we propose Motion Gesture (GS) for extracting a meaningful velocity segment to improve the effectiveness of the identification.

2.3 Abnormal Behavior Detection Techniques

Detecting cyber threats inside an enterprise is like finding a needle in a haystack. An advanced persistent threat (APT) is a set of stealthy and continuous computer hacking processes. APT attacks involve a high degree of covertness over a long period of time, using malware with sophisticated techniques to exploit vulnerabilities in systems and continuously monitor or extract confidential data from specific targets. According to surveys from well-known security institutes and companies [20] [21], only 31% of victims have the ability to discover by themselves that they have been compromised or that they have experienced a data breach internally. It takes an average of 205 days for threat groups present on a victim's network to be detected, and in the worst case, this can even take about eight years. For such emergent and sizeable demand, companies and industries devote significant time and resources to products that prevent users from being victims of APTs, such as firewall rules or malicious website blacklists. However, because of various tricks or camouflaging methods such as packing, obfuscated shellcode, or virtual private network (VPN) use, signature-based malware detection techniques like traditional intrusion detection systems (IDS) are becoming less useful, especially in identifying APTs at a post-compromised stage.

To mitigate the blind spot of traditional detection methods and alleviate the load upon human forensics, novel APT-monitoring approaches, such as mining attacks based on a huge amount of security device logs, have emerged in recent years [36] [37] [38] [39] [40]. T.F. Yen et al. first used principle component analysis to find significant intrusion signatures in 2013. This analysis was based on large-scale logs of security information and event management (SIEM) systems to detect suspicious activities on hosts in an enterprise [36]. Leveraging T.F. Yen's research from 2013, the same team extended the targeted scale to proxy logs, VPN logs, employee databases, and client-side anti-virus reports. This was done to determine the risk of hosts in an enterprise where the possibility of hosts being dynamically logged by multiple accounts was ruled out, according to Windows active directory authentication logs [37]. The research of A. Opera et al. [38], considered information recorded by a domain name system (DNS) and proxy, and used known malicious domain names or infected hosts as seeds. The research then adopted a belief propagation algorithm to detect other potentially compromised hosts and malicious domains. L. Invernizzi et al. further investigated full HTTP requests from networks owned by internet service providers (ISP), and tried to detect malware distribution and infrastructure with a global connection graph of malwares, hosts, accounts, or other entities involved in malware distribution [39]. In addition to focusing on signature-based statistics measurements, H.K. Pao et al. presented research considering causal relationships of IDS sequential alerts to detect intrusion [40].

However, to the best of our knowledge, existing malware or infected host detection mechanisms mostly belong to the following two categories: 1) traditional methods used for intrusion detection that are only focused on network-layer anomaly discovery, such as firewalls and IDS/IPS consistently tracing whether any pre-defined rule is triggered or not; 2) research on log analysis, usually estimating statistical measurements from various logs as signature features, without taking a user's sequential behavioral anomaly into consideration. When facing insider threats and APT attacks, detection methods with the above shortcomings might be easily evaded. Alternatively, users' intentions, habits or tendencies are often expressed when they are using application-layer services. For this reason, instead of using expert rules with only a few pre-defined signatures or network-layer recorded information, this thesis emphasizes highlighting the deviations in user behavior from the viewpoint of accessing application-layer services. To achieve this goal, the proposed method *ChainSpot* analyzes user's sequential behaviors and consists of the following essentials. 1) *ChainSpot* takes Windows active directory (AD) logs and proxy logs as time-series input data to provide chronological evidences. Both input logs are from application-layer services. The Windows AD domain controller of an organization monitors all related information when any intranet account tries to log on or acquire various services, while a proxy server acts as an intermediary between the user's computer and the Internet, allowing client computers to make indirect network connections to other network services. 2) Probabilistic graphical models are used to summarize patterns describing a user's sequential application-layer behaviors. In modeling a user's behavioral pattern, there is a trade-off between selecting appropriate features and keeping the computation simple. This issue was resolved in this thesis based on domain experience and a knowledge survey. 3) The last key point of *ChainSpot* is to adopt appropriate criteria for monitoring anomalies when there is a large deviation between a user's newly observed model and his or her historically benign one.

Datasets and ground truths for evaluation were collected from real environments comprising thousands of people. Experiments show that 1) the modeling method of *ChainSpot* has significant effectiveness in measuring a user's behavioral deviation when data from both normal and abnormal stages were supplied. 2) Deploying *ChainSpot*for anomaly monitoring results in very good support for threat detection, in which the detection results were highly correlated to expert-labeled ground truths. Proper settings for control parameters were also discussed and suggested using an efficient evaluating method [41]. 3) Representative case investigations demonstrate how the mechanism of *ChainSpot* works, and how the structural differences between benign and suspicious patterns, which effect *ChainSpot*'s predictions, could be further interpreted to reconstruct the cyber attack scenarios. As a result, these analytic findings might introduce novel domain knowledge and threat detection clues for digital forensics.

Chapter 3 METHODOLOGY

To analyze a user's keystroke and mouse movement behavior, the proposed model needs to track keyboard strokes and mouse movement without influencing the user's work. Although such tracking can assume various forms – e.g., software, hardware, or even external monitoring (such as acoustic analysis or electromagnetic emissions) – the model is implemented as a software agent that resides on the user's desktop. An organization can mandate its employees to install this agent and require them to run the software in the background of their operating system every time they use a computer. Accordingly, the agent can record and analyze the user's keystroke and mouse movement behavior. To obtain user training data, the organization can also mandate its employees to provide their keystroke and mouse behavior data within a fixed time period. Such data can then be used to create that user's behavior model for identification. Moreover, if an employee uses intentional delays or increases in speed to interrupt the keystroke and mouse movement behavior collection, causing the training data to be useless for data creator identification later, the company can still discover this abnormal behavior and conduct internal investigations. In the following sections, we first provide an overview of our proposed DLP model, and then show how to implement the framework using existing technologies.

3.1 Overview

Since a parser will extract a text stream – say streamA – from a file that is usually generated from user input, it is reasonable to attempt to identify the data creator by tracking and analyzing the user's keystroke behavior. This is done so that the newly extracted text stream – say streamB – has a very high probability of being identical to streamA. StreamB can then be verified using any of the aforementioned data matching techniques to determine its sensitivity. In order to obtain streamB and determine the identity of its creators, the central premise of our proposed DLP model is to create a Secure Keystream Analyzer (SKA). This can provide active file content analysis as follows (also see Fig. 3.1):

1. When a user (*e.g.*, Bob) starts an application (e.g., Microsoft Office Excel), the SKA will connect to the keyboard and mouse application programming interfaces (APIs).

2. When Bob types text in the application, the SKA records his keystroke and mouse movement behavior. Once the file (for example, newfileB) is saved, the SKA will start to analyze the record as follows.

3. As Bob's keystrokes (*e.g.*, typingB) may include a number of typing errors and useless keystrokes, the SKA will analyze and eliminate system keys (*e.g.*, Escape, Menu, Pause/Break, and PrintScreen/SysRq) and function keys (*e.g.*, F1, F2) in order to extract a refined streamB. The SKA also uses machine learning methods to identify the data creators by verifying keystroke and mouse movement behavior.

4. Subsequently, a file named log_newfileB containing the analyzed results, streamB, and the identity of the data creators is sent to a DLP system for sensitive data analysis.

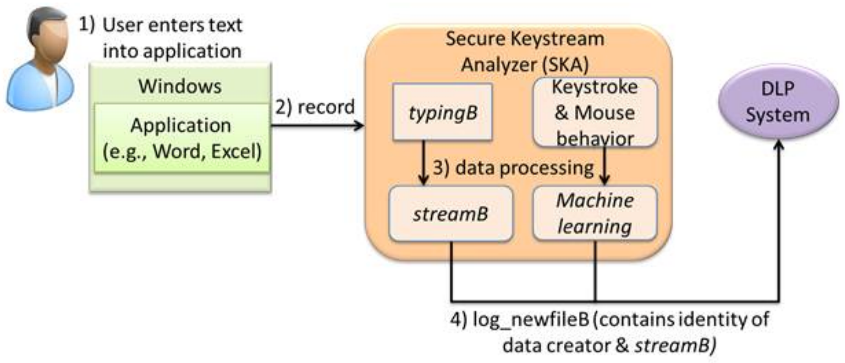

1) User enters text into application
Windows
Application (e.g., Word, Excel)
2) record
Secure Keystream Analyzer (SKA)
typingB
Keystroke & Mouse behavior
3) data processing
streamB
Machine learning
DLP System
4) log_newfileB (contains identity of data creator & streamB)

Figure 3.1: The concept of the proposed DLP model

The proposed SKA is equivalent to a text analyzer for files and is independent of but compatible with existing DLP systems. Therefore, once a DLP system determines that streamB contains sensitive data, the DLP can (i) tag newfileB as either "clean" or "sensitive" according to the receiving path, (ii) classify the sensitive data into different types, such as name, address, SSN, or passport number, (iii) refine the sensitivity level of the file according to the identity of its data creators, and (iv) calculate and send the fingerprint (e.g., the MD5 value) of newfileB to a security gateway, which will use the fingerprint to prevent the sensitive file from leaving the organization's network.

3.2 Keystroke Behavior Learning

Because keystrokes contain many interesting user typing behaviors, and the typing characteristics of each user (e.g., keyboard typing frequency or typing habits) are different, keystrokes can be used to verify user identity [42]- [44]. The sensitivity of the file can then be automatically fine-tuned using the identity of the data creator, and thus its security can be enhanced. For instance, if a manager types sensitive data, it is reasonable to assume that the sensitivity of such data is higher than data typed by general staff. Accordingly, a DLP system can monitor such data carefully. To achieve this goal, the SKA described in the previous section needs to collect the time cost of character typing for a user. Then, a machine learning algorithm is used to create the typing model for that user. An outline of the experiment is provided through the following basic steps:

- Recording and Extracting. The following corpus was typed during experiments:

  - "Min-Hwa Law; 0954125789; A239481567; mhlaw@gmail.com; 4234-4800-5437-2283"

  - "Mr. Law will arrive in 80 min; his phone number is 0954-234-437; email address is Min-Hwa948@gmail.com. Please contact him."

  - "Mr. Wang will arrive tomorrow with flight No. MH480; his phone number is 09371-25283; and the email address is wangming4289@gmail.com. Please contact him."

The time logs were collected in terms of characters and computer system time. Each character was categorized into one of 11 character location area types, and the switching time costs were extracted in milliseconds from the movement between 11 character types.

- Training and Testing: Machine learning techniques are applied for classifying users. The goal is to find the classifier and its parameters to achieve optimal accuracy.

3.2.1 Recording and Extracting

Because user behavior in keyboard operation and data input skills might vary between users, the results for the time cost of keying sensitive characters, symbols, and numbers, and for toggling character types, are different. By recording the time cost of switching between character types, the typing frequency of each user is obtained and can be used to create the user's typing model via machine learning algorithms. Subsequently, the model can be used to classify different data creators based on their typing frequency. All possible input characters were categorized into 11 types. Table 3.1 lists the character type ID and its corresponding characters.

The switching time cost of character type T was extracted from the character time cost, where $T_{i,j}$ is the switching time cost for inputting characters from the i-th type to the jth type. A total of 121 features of the switching time cost were extracted for each instance. In general, 60 characters were collected in one instance. The mean switching time cost was calculated from these features.

3.2.2 Training and Testing

The character time cost data introduced in the previous section can be applied

Table 3.1: Character type ID and characters

Character Type ID	Characters
1	qazwsxedc
2	rfvtgbyhn
3	ujmikolp
4	[] \ ’;,./
5	1234567890
6	QAZWSXEDC
7	RFVTGBYHN
8	UJMIKOLP
9	{}—:”?
10	!@#$%&*()_+‘-=
11	space

in various learning algorithms [45]. Support vector machines (SVMs) [46] were originally designed for binary classification. C. Hsu constructed a multi-class classifier by combining several binary classifiers [47]. Teh et al. [48] surveyed the research of keystroke dynamics biometrics. As a classification method, machine learning is widely used in the pattern recognition domain. A SVM generates the smallest possible region that encircles the majority of feature data related to a particular class. A SVM maps the input vector into a high-dimensional feature space via the kernel function. As a result, the separating function is able to create more complex boundaries and better determine to which side of a feature space a new pattern belongs. SVMs claim to have better performance than neural networks while being less computationally intense [49]. The effectiveness of an SVM depends on the kernel selection, the kernel parameters, and the soft margin parameter C. The Gaussian radial basis function (RBF) $\kappa(x_i, x_j) = exp(-\gamma \parallel x_i - x_j \parallel^2)$ is used for maximal margin hyperplanes. The kernel parameters γ and the cost parameter C are needed to estimate the best prediction. The LIBSVM tool [50], which is a method well established for SVMs, was included in the test environment. The tuning of γ and

C are selected by a grid search with exponentially growing sequences.

3.3 Mouse Movement Behavior Learning

Because mouse operation and movement behavior can vary per user, the results for the flying time costs and mouse movement acceleration are different. By recording the flying position and time of a mouse movement, the flying time costs of the moving distance in a certain direction and its acceleration profile are obtained for each user, and they can be used to create the user's typing model via machine learning algorithms. Subsequently, the model can be used to classify different data creators based on their typing frequency.

Mouse movement record datasets were collected over an extended period (*e.g.*, 4 hours) of regular mouse operation. A movement record dataset was collected as follows:

$$P = P_n(x, y, t) : x, y, t \in R, n > 0 \tag{3.1}$$

where x and y are the horizontal and vertical positions of the mouse in the screen, and t is the current universal time value. The recording time interval was 0.08 s. Users might randomly move the mouse at times, but most actions have an end target. For each movement between sequential points, the movement vector $\overrightarrow{P_n} = P_n(x, y) - P_{n-1}(x, y)$, and the direction angle θ of the movement vector can be calculated from Eqs. 3.9 and 3.10.

$$\Delta y = P_n(y) - P_{n-1}(y), \Delta x = P_n(x) - P_{n-1}(x) \tag{3.2}$$

$$\theta = \arctan(\Delta y / \Delta x) \tag{3.3}$$

Based on the values of $\theta, \Delta y, and \Delta x$, eight movement direction identities are defined in Table 3.2

Table 3.2: A data sample of character time cost for an instance

Move Direction Identity (R)	θ	$\Delta x, and \Delta y$
1	$-\pi/8 \leq \theta < \pi/8$	$\Delta x \geq 0$
2	$\pi/8 \leq \theta < 3\pi/8$	$\Delta x \geq 0$
3	$\theta \geq 3\pi/8 \parallel \theta < -3\pi/8$	$\Delta y \geq 0$
4	$-3\pi/8 \leq \theta < -\pi/8$	$\Delta y \geq 0$
5	$-\pi/8 \leq \theta < \pi/8$	$\Delta x < 0$
6	$\pi/8 \leq \theta < 3\pi/8$	$\Delta x < 0$
7	$\theta \geq 3\pi/8 \parallel \theta < -3\pi/8$	$\Delta y < 0$
8	$-3\pi/8 \leq \theta < -\pi/8$	$\Delta y < 0$

The move distance of $\overrightarrow{P_n}$ is defined as

$$d_n = \Delta\sqrt{(\Delta x)^2 + (\Delta y)^2}. \tag{3.4}$$

The Effectiveness of the Mouse Movement Segment (**EMMS**) is defined to extract a meaningful move segment. **EMMS** =a subset of P that consists of k consecutive data records (P_n) with the same R_n, and that satisfies the conditions

$$\forall \qquad d_n \geq \delta_1 \qquad and \qquad k \geq \delta_2. \tag{3.5}$$

where δ_1 is the threshold for effective mouse move distance, *e.g.*, four pixels, below which tiny movement can be ignored, and δ_2 is the threshold of consecutive data records. The overall movement distance D and the flying time cost Δt of **EMMS** can be derived from the start point P_{start} to the end point P_{end} of **EMMS**.A distance value of 10 degrees was used to represent the range of movement. The maximum degree value is 10, which means that the movement distance is more than 90% of the diagonal distance of the screen resolution. Moreover, acceleration can be the forward-moving force of user behavior. The acceleration information was used as an additional feature to represent the behavior of each **EMMS**,

$$a_n = \frac{v_n - v_{n-1}}{\triangle t_n} = \frac{d_n/\triangle t_n - d_{n-1}/\triangle t_{n-1}}{\triangle t_n}, n > 2 \tag{3.6}$$

where d_n is the move distance from recorded point P_{n-1} to P_n, $\triangle t_n$ is the movement time cost from recorded point P_{n-1} to P_n, and v_n is the moving velocity at P_n. A three-degree polynomial curve that fits A was used to smooth the distribution.

$$\Lambda = f(a_n), n = 3 \sim \textit{the data number of } \mathbf{EMMS} \tag{3.7}$$

Further, the features Λ_{max}, Λ_{min}, and $T_{\Lambda=0}$ can be decided from Λ, where Λ_{max} is the maximum value of acceleration, Λ_{min} is the minimum value of acceleration, and $T_{\Lambda=0}$ is the zero acceleration time point from maximum to minimum. A value interval of 5 degrees was used to describe the user acceleration profile. The Λ_{max} and Λ_{min} degrees are relative to the global maximum and minimum acceleration value of Λ, and $T_{\Lambda=0}$ is relative to the overall flying time cost $\triangle t$ of the **EMMS**. An instance might consist of multiple **EMMS** (*e.g.*, 100) to adequately describe mouse behavior. The mouse behaviors were collected and combined into a total of 95 features as follows:

- Average flying time cost for jth direction and kth degree of distance, where j = 1–8 and k = 1–10.
- The ratio $r\Lambda_{max}(l)$ for the degree of acceleration Λ_{max} profile, where $r\Lambda_{max}(l)$ is the count of l degree over total **EMMS** count and l = 1–5.
- The ratio $r\Lambda_{min}(l)$ for the degree of acceleration Λ_{min} profile, where $r\Lambda_{min}(l)$ is the count of l degree over total **EMMS** count and l = 1–5.
- The ratio $rT_{\Lambda=0}(l)$ for the degree of $T_{\Lambda=0}$ profile, where $\Lambda_{rT_{\Lambda=0}}(l)$ is the count of l degree over total **EMMS** count and l = 1–5.

3.4 Smartdevice Touch-Panel Behavior Learning

To improve the effectiveness of continuous authentication methods, we considered our previous work—a method for identifying users using traditional computers [51]. Because user behavior with touch-panel operations can vary between users, the results relating to point pressure, sliding velocity, and acceleration are different. By recording the pressure, position, and area of a touch-panel operation, the flying costs of the moving distance in a certain direction and its acceleration profile are obtained from each user, and these can be used to create a typing model unique to each user through machine-learning algorithms. Subsequently, the model can be used to classify different data creators based on their behavior.

Pointing and sliding datasets are collected over an extended period (e.g., 1 hour) of regular touch-panel operations. An operational dataset is collected as follows:

$$P = \{P_n(A, p, x, y, t) : A, p, x, y, t \ \in R, n > 0\} \tag{3.8}$$

where A is the TOUCH_MAJOR area, p denotes the touch pressure at a certain point, x and y are the horizontal and vertical positions of the mouse on the screen, and t is the current universal time value. The recording time interval was approximately $0.01s$.

The degree value $l = 1 \sim 5$ is used to represent the feature value as small, small-medium, medium, medium-large, or large. For a feature value $f_{i,n}$ with respect to the maximum value $f_{i,max}$ of the i^{th} feature among all users, the degree value of l can be described as follows:

$$l = \left[\frac{f_{i,n}}{f_{i,max}} * 5 \right] + 1, \ 1 \leq l \leq 5 \tag{3.9}$$

For all l of the i^{th} feature of the user, a distribution ratio set $r(l)$ was extracted to describe the weight of each degree for the i^{th} feature of the user.

$$r(l) = \frac{number\ of\ l\ degree}{N},\ l = 1, 5 \tag{3.10}$$

Here, $r(l)$ is a value between 0 and 1, and the summation of $r(l)$ is 1.

For pointing operations, 15 features were extracted from A and p as follows:

- The distribution ratio set $rA(l)$ for the degree of the A_{max} feature, $l \in \mathbb{N}, l \leq 5$.
- The distribution ratio set $rp(l)$ for the degree of the p_{max} feature, $l \in \mathbb{N}, l \leq 5$.
- The distribution ratio set $rpA(l)$ for the degree of the $\left(\frac{p}{A}\right)_{max}$ feature, $l \in \mathbb{N}, l \leq 5$.

Here, A_{max}, p_{max}, and $\left(\frac{p}{A}\right)_{max}$ are the maximum values for A, p, and $\left(\frac{p}{A}\right)$, respectively, in a pointing operation.

While users can slide to a position randomly, most actions have an end target. For each movement between sequential points $z_n(x, y)$ and $z_{n-1}(x, y)$, the movement vector $\vec{z_n} = z_n - z_{n-1}$, x, y are the coordinates for Z_n. The directional angle θ for the movement vector can be calculated from Eqs. (3.11) and (3.12).

$$\Delta y = z_n(y) - z_{n-1}(y),\ \Delta x = z_n(x) - z_{n-1}(x) \tag{3.11}$$

$$\theta = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right) \tag{3.12}$$

Based on the values θ, Δy, and Δx, two movement-direction identities are defined in Table 3.3.

Table 3.3: Data sample for the characteristic time-cost of an instance

Moving Direction Identity(R)	θ
1	$\frac{-3\pi}{8} \leq \theta < \frac{\pi}{8}$
2	$\theta \geq \frac{\pi}{8} \parallel \theta < \frac{-3\pi}{8}$

The moving distance of $\vec{z_n}$ is defined as

$$d_n = \sqrt{\Delta x^2 + \Delta y^2} \tag{3.13}$$

Similarly, the velocity can be calculated as a feature of the user's behavior. The velocity information v_n and pressure-release velocity are used as additional features to represent the behavior of each sliding event:

$$v_n = \frac{d_n - d_{n-1}}{\Delta t_n}, \;\; vp_n = \frac{p_n - p_{n-1}}{\Delta t_n}, \;\; n > 1 \tag{3.14}$$

where d_n is the moving distance from a recorded point z_{n-1} to z_n, Δt_n is the movement duration from a recorded point z_{n-1} to z_n, v_n is the moving velocity at z_n, and vp_n is the pressure-release velocity at z_n.

For a sliding operation, 40 features were extracted from v_n, vp_n, and the moving direction R as follows:

- The distribution ratio set $rv_m\left(l\right)$ for the degree of the feature of velocity v_{max}, profiled in two directions, $l \in \mathbb{N}, l \leq 5$.
- The distribution ratio set $rv_a\left(l\right)$ for the degree of the feature of velocity v_{avg}, profiled in two directions, $l \in \mathbb{N}, l \leq 5$.
- The distribution ratio set $rvp_m\left(l\right)$ for the degree of the feature of velocity vp_{min}, profiled in two directions, $l \in \mathbb{N}, l \leq 5$.
- The distribution ratio set $rvp_a\left(l\right)$ for the degree of the feature of velocity vp_{avg}, profiled in two directions, $l \in \mathbb{N}, l \leq 5$.

Here, v_{max} denotes the maximum value of v_n, vp_{min} is the minimum decreasing value of vp_n, and v_{avg} and vp_{avg} are the average values of v and vp, respectively.

3.5 Abnormal User Behavior Analysis

3.5.1 Windows Active Directory Domain Service

The Active Directory (AD) domain controller provides a directory service that Microsoft developed for Windows domain networks [52] [53]. An AD domain controller authenticates and authorizes all users in a Windows domain-type network. For example, when a user logs into a computer that is part of a Windows domain, Active Directory checks the submitted password and determines whether the specified account is a legal user in this domain even a system administrator [54]. The AD domain controller of an organization also monitors all related information when any accounts in the domain try to allocate or acquire various resources and services, in addition to helping to set security policies for all computers and install software updates. Details about how the AD controller works can be found in [55].

3.5.2 Proxy Service

In computer networks, the major role played by a proxy is as a web proxy, facilitating access to content on the World Wide Web and providing anonymity [56]. A proxy server acts as an intermediary between intranet clients and other servers on the internet to serve requests sent by clients seeking resources from other servers. The operation procedure of a proxy is: 1) an intranet client that needs some service on the internet first connects to the proxy server and tells the server its request. The requested service may be one of various types, such a file, connection, or web page. 2) The proxy server then parses the client's request and verifies whether or not this request obeys pre-defined policies, based on what is being requested or which website holds the requested service. 3) Once this request is accepted by the proxy, the client will build a connection to the server on the internet for subsequent service requesting. If the request is not accepted, this connection will be denied by proxy.

3.5.3 The Real Dataset

The materials used for performance evaluations of the proposed framework are introduced in this section. In this thesis, a certain government organization consisting of approximately 1,000 employees in Taiwan was selected as the deployment environment for the proposed method. Materials collected from this organization comprise three types of data. Two of them, AD and proxy logs, are service logs used for user behavior modeling and anomaly detection. The other type of data is security operation center (SOC) event tickets, labeled with various security anomaly events by forensics experts who provide ground truths that specify which IPs related to certain accounts behave abnormally. The descriptions and statistics about the real dataset used are as follows. The Active Directory domain service of Windows Server 2008 R2 is mounted on this organization's domain network and monitors all service requests and resource allocations raised from intranet accounts. In this case, a total of 27,902,857 logs were collected during one month, from 2016/08/01 to 2016/08/31. The proxy logs over the same time period contain 78,044,332 logs. All log data supplying information about user behaviors will be dispatched to each account's profile based on the recorded field, "account name." The full size of the collected data (AD and proxy logs) is 152.299 gigabytes. On the other hand, a total of 99 SOC event tickets across six types were generated in August of 2016. These event tickets were used as a source of ground truth because SOC tickets indicate which accounts are attacked or are acting strangely at a specific time or over a time duration. The number of abnormal accounts correlated by SOC tickets is 1,089. Because of confidentiality concerns, please contact the corresponding author for any requests about this real data. Table 3.4 lists all numbers of SOC-tickets collected from the target organization.

3.5.4 The Proposed Method - *ChainSpot*

This section and following subsections define what components constitute the adopted Markov chain and how to build it given the dataset consisting of logs from an employee. In probability theory, a Markov chain is a stochastic approach

Table 3.4: Number of different SOC tickets

Index	Type of SOC ticket	#tickets
1st	Single account failed too many times when logon	4
2nd	Multiple accounts logon from single IP in short time	5
3rd	Host connects to malicious domain	11
4th	Buffer overflow attack from outside	19
5th	DoS attack from outside	59
6th	Try to logon in non-working hours	1

to model randomly changing systems where it is assumed that future states depend only on the present state and not on the sequence of events that preceded it (that is, it assumes the Markov property) [57]. Generally, the reason for taking this assumption is that it enables subsequent reasoning and computation regarding the model that would otherwise be intractable. The simplest Markov model is the Markov chain. It models the state of a system with a random variable that changes through time. In this context, the tendency of every transition from one state to another is described by a probability. The Markov property suggests that the distribution of this probability depends only on the distribution of the previous state. The advantage of being successful in describing consequent state changes means that there are many applications, such as speech recognition [58], hand-written text recognition [59], gesture recognition [60], and cyber security intrusion detection [40], that are based on the Markov chain itself or its popular variant, the hidden Markov model.

3.5.5 State Constitution of Used Markov Chain

The first subsection explains which information collected from AD and proxy logs is used to construct the states of the Markov chain. The goal of *ChainSpot* is to model a user's behaviors, which can express their intention or anomalies if something unexpected occurs. However, too much information that cannot be interpreted may simply worsen computational complexity. For this reason, a feature selection strategy in the data pre-processing of *ChainSpot* will decide which type of information should be included in building the Markov chain. The major principle consists of two parts. 1) Logs driven spontaneously by a machine to acknowledge protocol or exchange information do not directly reflect user intent, so these type of logs (for instance, AD event codes "4624" and "4634") will be ruled out and will not be considered. 2) Data fields that describe users' intentions and habits (such as "Accessed Domain Name" in proxy logs) will be retained to form Markov states.

For the final state constitution used in AD logs:

1) 1) All active directory event codes, except event codes "4624" and "4634," were included [61].

2) Event code "4771" which means "Kerberos Authentication Failed" will be considered when accompanied with a corresponding field "reply code", where "0x12" means that "this account has been locked out" and "0x18" stands for "wrong password" [62].

The following fields are included in the Markov state for proxy logs modeling:

1) "Adopted HTTP Method" represents the value of "GET" or "POST" indicating a user's query action.

2) "Download or Upload" represents the packet flow direction of the access connection.

3) "Accessed Domain Name" stores the second-level domain name that this user visited.

4) "Access Result" records the value of "allowed" or "denied", which is the final judgment on whether or not pre-set proxy policy allows this user's connection.

Consequently, the Markov chain based on AD logs is a sequence of all of the produced Windows AD event codes, where event codes "4624" and "4634" were excluded and event code "4771" was accompanied with the additional field, "reply code". On the other hand, the state constitution for a proxy Markov chain is a sequence of integrated field information, such as "try to *GET* and *Download* on *microsoft.com* then this action was *denied*".

3.5.6 Build a Markov Chain Given a Log Sequences Dataset

Given the dataset D_i containing log sequences of the i^{th} accounts, $i = 1, ..., E$, the resulting Markov chain based on the dataset should include the following components:

1) A finite state set $S = \{st_1, st_2, ..., st_{ns}\}$ that contains all possible states of Markov chains defined in the previous subsection and derived from D_i. Note that ns is the total number of derived states in D_i;

2) An $ns \times ns$ transition probability matrix (TPM), as follows:

$$TPM = \begin{bmatrix} tp_{1,1} & \dots & tp_{1,j} & \dots & tp_{1,ns} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ tp_{i,1} & \dots & tp_{i,j} & \dots & tp_{i,ns} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ tp_{ns,1} & \dots & tp_{ns,j} & \dots & tp_{ns,ns} \end{bmatrix}$$

where for each i and j, $tp_{i,j}$ represents the transition probability from i^{th} state to j^{th} state, with constraints that $\sum_{j=1}^{ns} tp_{i,j} = 1$, and $i = 1, ..., ns$.

$$\forall\ i, j = 1, ..., ns,$$
$$tp_{i,j} = \frac{\#\text{transitions from } st_i \text{ to } st_j \text{ in } D_i}{\#\text{transitions starting from } st_i \text{ in } D_i}$$

It should be noted that for a general setting, there is one additional parameter describing a Markov chain, a $ns \times 1$ initial probability vector. The real-valued elements in the initial probability vector represent the probability of any state being the initial state of an event chain. However, due to the reality in a practical environment that it is almost impossible to precisely partition a user's sequential behaviors into several independent behavior segments, *ChainSpot* omits the need to use an initial probability vector by taking all behaviors in one user's profile as a single but lengthy log sequence to simplify data preprocessing.

3.5.7 Deviation Estimation Given Different Markov Chains

The anomaly detection mechanism of *ChainSpot* relies on appropriately estimating the deviation from a user's new profile to his benign one. In this paper, user profiles are summarized as patterns represented by Markov chains, and as a result, deviation measurements can be performed using graph edit distance (GED) to quantify similarity (or dissimilarity) between different Markov chain models. The formal graph edit distance between two graphs G_1 and G_2, written as $GED(G_1, G_2)$ can be defined as follows:

$$GED(G_1, G_2) = \min_{(e_1,\ldots,e_k)\in P(G_1,G_2)} \sum_{i=1}^{k} cost(e_i), \tag{3.15}$$

where $P(G_1, G_2)$ denotes the universal set of editing paths isomorphically transforming G_1 into G_2, and $cost(e_i)$ is the cost of each graph editing operation, e_i.

With respect to Markov chains in *ChainSpot*, the calculation of GED on two Markov chains can then be implemented by following equation (3.16):

$$GED(TPM^1, TPM^2) = \sum_{i=1}^{ns}\sum_{j=1}^{ns} \left|tp^1_{i,j} - tp^2_{i,j}\right|, \tag{3.16}$$

where TPM^1 and TPM^2 are transition probability matrices of two Markov chains, and $tp^1_{i,j}$ and $tp^2_{i,j}$ are the corresponding transition probabilities in TPM^1 and TPM^2, respectively. The ns is the number of states in the Markov chain.

Chapter 4 EXPERIMENTS

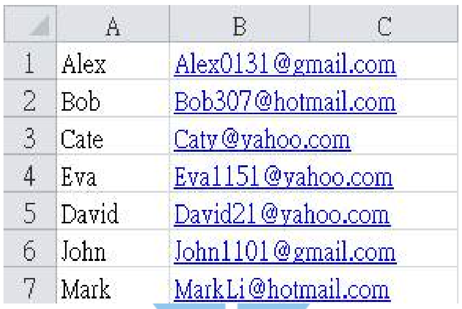
This chapter will verify the use of the established model of machine learning using keystrokes, mouse movement, and touch panel usage behavior to classify identification and to test the effectiveness and performance of abnormal behavior detection. The user identification research involved 10 volunteers and a variety of user behaviors in multiple input contexts, which were tested by collecting user input behavior records including time stamp, coordinate position, action patterns, and miscellaneous system information. After extracting the data features, the SVM classifier was established and the results were tested. The correctness of the diverse assessments was then compared. Cross validation was used to identify good parameters so that the classifier can accurately predict unknown data and prevent the overfitting problem [63].In v-fold cross-validation, the training set was divided into v subsets of equal size. One subset was tested sequentially using the classifier trained on the remaining v-1 subsets. Leave-p-out cross-validation is a method to increase the proportion of a test set, p subsets were used to test. Our experiment was based on the standard method (50% training set and 50% test set). To mitigate the over-fitting issue, a (10%) portion of the training set was used for a parameter tuning pretest. The parameter was tuned before using cross-validation to evaluate the test set. Ten-fold data subsets were generated randomly. We left 5-fold subsets out for cross-validation testing, 4-fold subsets were used to build the SVM classifier, and a 1-fold subset was used to pretest for parameter tuning. The 10-fold data subsets were randomly recombined for training, tuning, and testing on a rotation estimation of ten runs.

4.1 Framework Implementation

The following sub-section describes how to implement the SKA and how the SKA can collaborate with a DLP system.

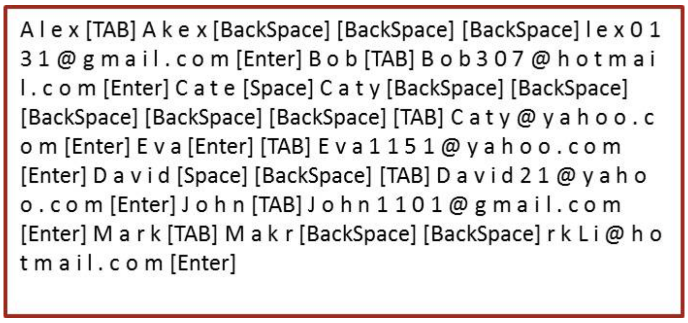
4.1.1 Secure Keystream Analyzer (SKA)

During implementation, the SKA hooks the keyboard APIs to track and profile user behavior. 4.1(a) shows Bob entering some contact information into Microsoft Excel, whereas 4.1(b) shows the raw input data, typingB, recorded by the SKA. To track Bob's mouse movement behavior as he types typingB, the SKA also installs a mouse hook to record mouse movement behavior on the screen, pixel by pixel. Since mouse movement behavior contains valuable patterns that can be used to identify the user [64], the SKA combines such information with the keystroke behavior pattern to identify data creators.



	A	B	C
1	Alex	Alex0131@gmail.com	
2	Bob	Bob307@hotmail.com	
3	Cate	Caty@yahoo.com	
4	Eva	Eva1151@yahoo.com	
5	David	David21@yahoo.com	
6	John	John1101@gmail.com	
7	Mark	MarkLi@hotmail.com	

(a) Bob's input.



A l e x [TAB] A k e x [BackSpace] [BackSpace] [BackSpace] l e x 0 1 3 1 @ g m a i l . c o m [Enter] B o b [TAB] B o b 3 0 7 @ h o t m a i l . c o m [Enter] C a t e [Space] C a t y [BackSpace] [BackSpace] [BackSpace] [BackSpace] [BackSpace] [TAB] C a t y @ y a h o o . c o m [Enter] E v a [Enter] [TAB] E v a 1 1 5 1 @ y a h o o . c o m [Enter] D a v i d [Space] [BackSpace] [TAB] D a v i d 2 1 @ y a h o o . c o m [Enter] J o h n [TAB] J o h n 1 1 0 1 @ g m a i l . c o m [Enter] M a r k [TAB] M a k r [BackSpace] [BackSpace] r k L i @ h o t m a i l . c o m [Enter]

(b) The *typingB* recorded by SKA.

Figure 4.1: Keystroke tracking

Once the SKA detects that Bob has saved his typing into a file called newfileB.xls (through the CreateFile and WriteFile APIs), typingB is analyzed accordingly. Since there are a number of control keys (e.g., [TAB], [BackSpace], [Space], and [Enter]) in typingB, the SKA preserves all alphabetical, numeric, and punctuation keys, and translates [TAB], [Space], and [Enter] into a space between words. It ignores function keys and system keys because these are useless in refining streamB.

Finally, the SKA counts the number of [BackSpace] (or [Delete]) strokes following a word to infer the output of Bob's typing, and stores the analysis results of streamB in an XML format file called log_newfileB in XML format (see Fig. 4.2). It is easy to see from Fig. 4.2 that log_newfileB.xml contains the same text that Bob initially entered in Microsoft Excel. As a result, a DLP system with an XML parser can discover that newfileB.xls contains a certain amount of sensitive data, such as personnel names and email addresses, through the analysis of log_newfileB.xml.

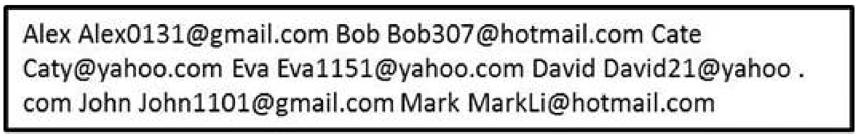
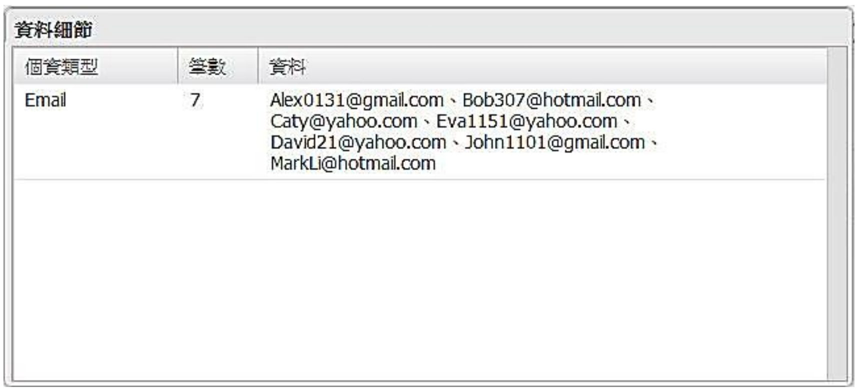

Alex Alex0131@gmail.com Bob Bob307@hotmail.com Cate
Caty@yahoo.com Eva Eva1151@yahoo.com David David21@yahoo .
com John John1101@gmail.com Mark MarkLi@hotmail.com

Figure 4.2: StreamB in log_newfileB.xml

4.1.2 DLP System

We use the PrivacyID tool developed by amXecure [25] in our proposed model. PrivacyID is an effective DLP tool that is implemented as an agent and is deployed on the client's computer. Our model also includes a Central Management Server (CMS). It supports 23 file formats (XML, XLS(X), DOC(X), PPT(X), EPUB, ODS, et cetera) and seven sensitive data types (Chinese name, email, credit card number, phone number, et cetera), and it performs well when detecting sensitive content in files. Once the agent discovers sensitive content, it sends the content to the CMS. In this way, the system administrator can keep track of the distribution of sensitive files throughout the company. To show that the result of the analysis of log_newfileB.xml is equivalent to the original newfileB.xls, the PrivacyID agent is triggered. The agent performs sensitive content scanning after authenticating the user. When we log into the CMS following the completion of the process, the details of log_newFileB.xmlcan be found. It is clear that the email addresses found in log_newFileB.xml are the same as those entered by Bob in the original newfileB.xls newfileB.xls (see Fig. 4.3).


資料細節

個資類型	筆數	資料
Email	7	Alex0131@gmail.com、Bob307@hotmail.com、Caty@yahoo.com、Eva1151@yahoo.com、David21@yahoo.com、John1101@gmail.com、MarkLi@hotmail.com

Figure 4.3: The detailed results of CMS

4.2 Keystroke Behavior Experiments

A total of 589 instances of ten users were tested and recorded in three experiments; the information for the experiment instances is listed in Table 4.1. Three different character sets were input and tested in three runs of the experiments:

1. 73 characters were input, and three instances were extracted from the record data of the 1st to 60th, 11st to 70th, and 21st to 73rd character sets.

2. 119 characters were input, and 10 instances were extracted from the record data of the 1st to 60th, 11st to 70th, 21st to 80th, 31st to 90th, 41st to 100th, 51st to 110th, and 61st to 119th character sets.

3. 149 characters were input, and 10 instances were extracted from the record data of the 1st to 60th, 11st to 70th, 21st to 80th, 31st to 90th, 41st to 100th, 51st to 110th, 61st to 120th, 71st to 130th, 81st to 140th, and 91st to 149th character sets.

Table 4.2 provides a data sample of characters and switching time costs for an instance between different types. For example, for the serial characters "Min-Hwa Law; 0954125789;..." in the typing process, the time cost for each character would be recorded in milliseconds as 300, 190, 872, 711, 1121 ..., and the type for each character would be encoded to 8, 3, 2, 10, 7. The switch time cost between "M" and "i" character was marked as $T_{8,3}$, and its time cost value was 190 ms.

A confusion matrix is a specific table layout that allows performance visualiza-

Table 4.1: The information of experiment instances

Experiment	Char. #	Instance #	Test Times #	Test User #	Total Instance #
1	73	3	10	10	300
2	119	7	3	9	189
3	149	10	1	10	100

Table 4.2: A data sample of character time cost for an instance

Character time cost
300; 190; 872; 711; 1121; 311; 460; 581; 611; 340; 481; 271; 640; 411; ...
Character encode type
8; 3; 2; 10; 7; 1; 1; 11; 8; 1; 1; 4; 11; 5; 5; 5; 5; 5; 5; 5; 5; 5; 5; 4; ...
Instance features
$T_{8,3}$=190; $T_{3,2}$=872; $T_{2,10}$=711; $T_{10,7}$=1121; $T_{7,1}$=311; $T_{1,1}$=460; ...

tion of a classification system. The confusion matrix table was generated and the accuracy was evaluated for every estimate subset and testing subset.

Fig. 4.4 shows the confusion matrix results of the 10-run SVM test and the average identification accuracy for each user. In the left figure, the matrix diagonal corresponds to correct classification assignments. Outside of the matrix diagonal, each raw value represents the ratio of false negatives to other users, and each column value represents the ratio of false positives to the user. The right figure shows the error bar of standard deviation for each user in the 10-run test. As a proof of concept, keystroke profiling can not only inspect sensitive content without decoding the file format, but it can also effectively determine the identity of the data creator. It is clear that the sensitivity level of such data can be automatically fine-tuned with the significance of the corresponding identity, and, accordingly, can enhance the inspection capability of the DLP system. Because a user might use a mouse as an auxiliary input device when creating data, the accuracy of the data creator identification can be further improved by integrating mouse movement behavior. Section 4.2 demonstrates the learning method for mouse movement behavior.

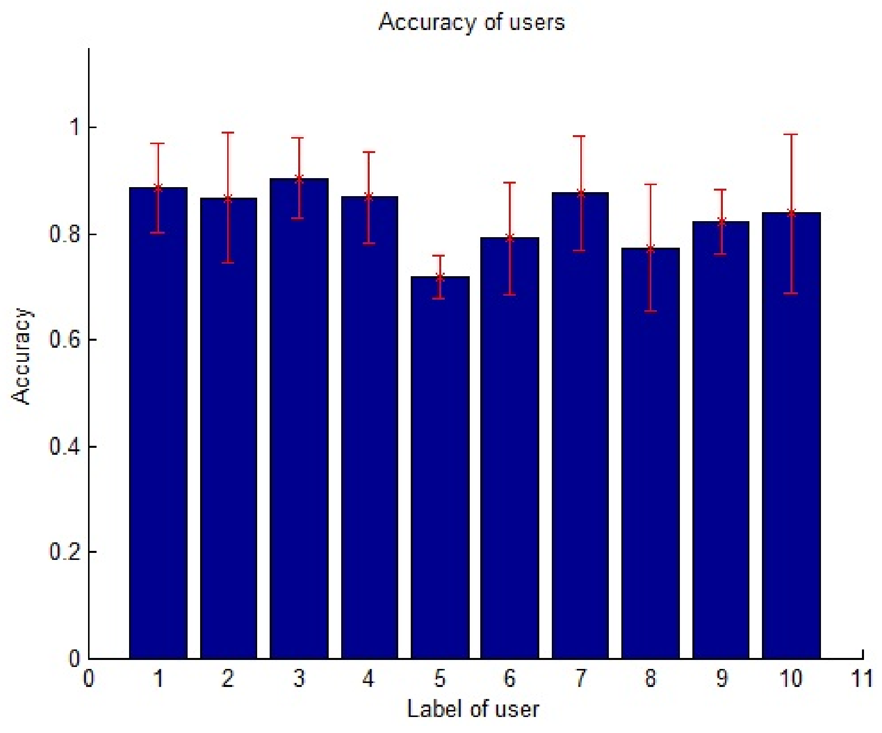
4.3 Mouse Movement Behavior Experiments

A total of ten users participated in the mouse movement behavior experiment with an individual computer. Mouse operation behavior was recorded approximately every $0.08s$ over the course of 4 h, *i.e.*, $180,000$ data points were logged. **EMMS** were extracted using the method described in Section 3.4. The number of extracted **EMMS** for each user is listed in Table 4.3.

The **EMMS** dataset was assigned randomly to the 40 instances. A dataset with 95 features for each instance was generated. The methods described in Section 3.2 were used for learning and classification. The Leave-5-Out 10-fold cross-validation method was used for evaluation. A total of 40% of the instances were trained, 10% of the instances were used to tune the parameters, and the remaining 50% of the

	1	2	3	4	5	6	7	8	9	A
1	0.89	0.05	0.01	0.03	0.01	0.01	0.00	0.02	0.00	0.00
2	0.03	0.86	0.00	0.01	0.01	0.05	0.01	0.01	0.02	0.00
3	0.01	0.00	0.91	0.03	0.01	0.03	0.00	0.01	0.00	0.01
4	0.02	0.00	0.02	0.87	0.04	0.00	0.00	0.02	0.00	0.03
5	0.03	0.04	0.03	0.08	0.72	0.05	0.03	0.01	0.00	0.01
6	0.05	0.09	0.03	0.01	0.00	0.79	0.01	0.00	0.02	0.00
7	0.00	0.07	0.00	0.01	0.02	0.01	0.87	0.00	0.01	0.01
8	0.06	0.03	0.02	0.02	0.04	0.03	0.00	0.77	0.00	0.03
9	0.00	0.11	0.00	0.00	0.00	0.05	0.01	0.01	0.82	0.00
A	0.01	0.00	0.05	0.04	0.00	0.00	0.04	0.05	0.00	0.82

(a) The average confusion results.



(b) The average identification accuracy

Figure 4.4: The average testing results of ten runs for each user. [$C = 46.6, \gamma = 0.000000146.$]

Table 4.3: The number of extracted EMMS of each user ID

User ID	EMMS #
1	1481
2	2264
3	4345
4	4490
5	6281
6	2521
7	2528
8	3855
9	678
10	2375

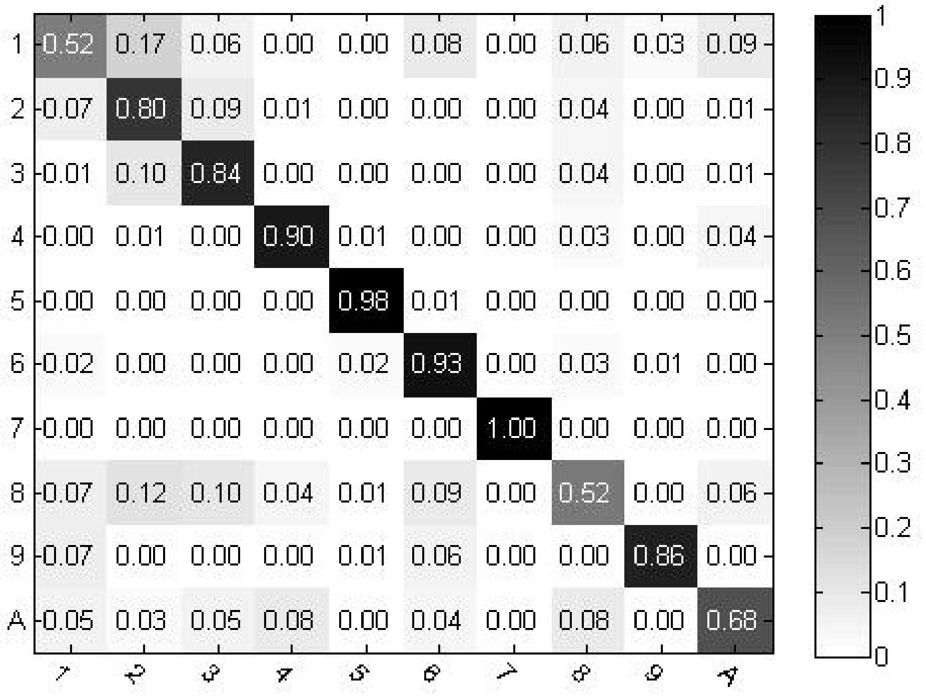
instances were tested. The overall test accuracy is 80.25% with ten-run experiments. Fig. 4.5 shows the confusion matrix results of the ten SVM tests and the average identification accuracy for each user. The matrix diagonal corresponds to correct classification assignments, and the total accuracy was calculated by dividing the sum of the matrix diagonal values by the testing sample number.

4.4 Practical Keystroke and Mouse Behavior Experiments

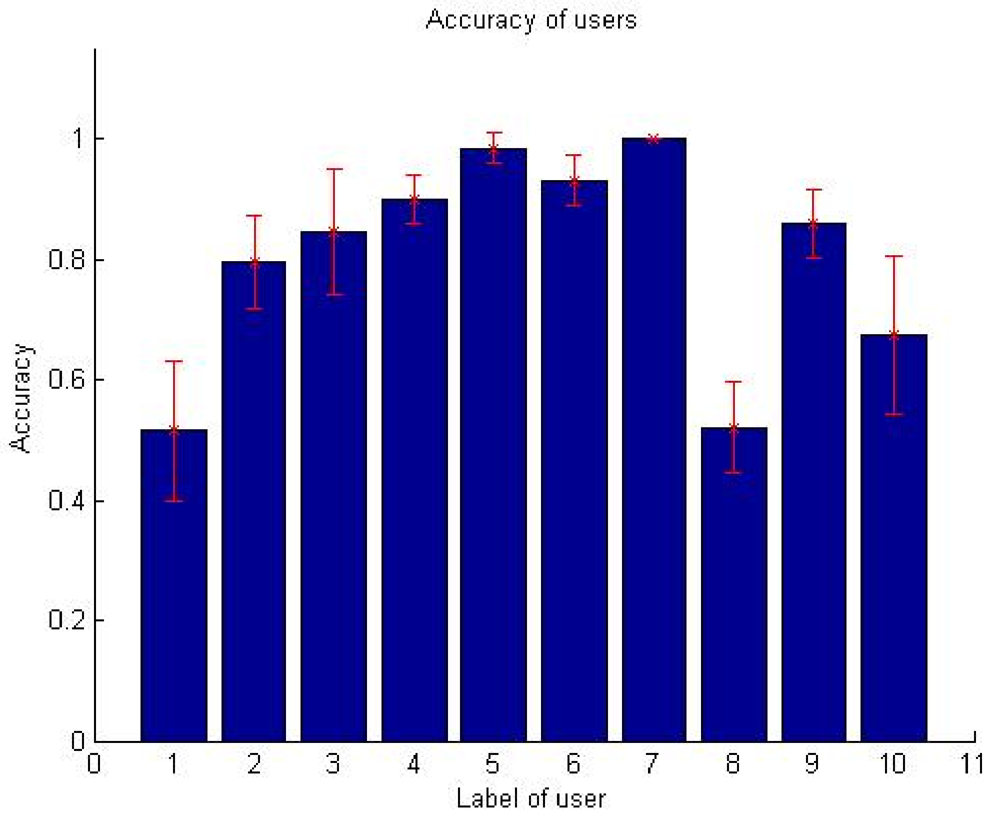
4.4.1 Diverse Style Keystroke Experiment

To extend the coverage of diverse styles, the text from ten additional articles were included in the keystroke experiment [Sep. 2014, Yahoo.com]:

- New phablets from Apple, Samsung could revive mobile market.



(a) The average confusion results.



(b) The average identification accuracy.

Figure 4.5: The average testing results of ten runs for each user in the mouse

- The Microsoft Engine That Nailed The World Cup Is Predicting Every NFL Game.
- The AL East-leading Baltimore Orioles have hit the most home runs in baseball, and it's not even close.
- This Baby's Reaction to Hearing for the First Time Is Guaranteed to Make You Smile.
- Tomorrow - Partly cloudy with afternoon showers or thunderstorms.
- Taiwan's first budget airline said Monday it was scheduled to launch its maiden flight later this month.
- Google Is Working on a Chip That Lets Machines Think Like Humans.
- Hackers break into server for Obamacare website: U.S. officials.
- Ebola Could Reach the U.S. By the End of This Month.
- Photos from the celebrities were stolen individually, the company said.

A total of 100 instances of five users were retested and recorded in two keystroke experiments. Fifty percent of the data was joined with the previous training set, the classifier was rebuilt using the method described in Section 3.3, and the remaining 50% of the data was tested. The average accuracy of the new test set was 84.2%.

4.4.2 Combining Keystroke and Mouse Movement Profiling Findings

Using the keystroke feature data set with 610 instances from ten users, the mouse movement **EMMS** data set was randomly assigned to 61 instances and then combined to form a new user behavior feature data set. The mouse behavior feature value required normalization to meet the keystroke behavior feature value range, i.e., the mouse behavior features value had to be multiplied by $1,000$. The methods described in Section 3.3 were used for learning and classification, and the Leave-5-Out 10-fold cross-validation was used for evaluation. Various methods were used to evaluate our retrieval system. Table 4.4 lists the results of the first SVM test for various user measures, where $TP\ \# =$ the number of true positives, $FN\ \# =$ the number of false negatives, $FP\ \# =$ the number of false positives, $TN\ \# =$ the number of true negatives, accuracy $A = (TP{+}TN)/(TP{+}FN{+}FP{+}TN)$, precision $P = TP/(TP{+}FP)$, recall $R = TP/(TP{+}FN)$, and F-measure $F = 2PR/(P{+}R)$.

Table 4.4: Results for various user measures in the first SVM test
[A: Accuracy, P: Precision, R:Recall, F:F-measure]

ID	TP #	FN #	FP #	TN #	A	P	R	F
1	30	1	5	267	0.98	0.86	0.97	0.91
2	28	2	2	271	0.99	0.93	0.93	0.93
3	30	0	6	267	0.98	0.83	1.00	0.91
4	29	1	1	272	0.99	0.97	0.97	0.97
5	28	2	0	273	0.99	1.00	0.93	0.97
6	28	3	2	270	0.98	0.93	0.90	0.92
7	30	0	0	273	1.00	1.00	1.00	1.00
8	27	3	2	271	0.98	0.93	0.90	0.92
9	29	2	1	271	0.99	0.97	0.94	0.95
10	25	5	0	273	0.98	1.00	0.83	0.91

Table 4.5 lists the additional results for user measures from the ten SVM tests conducted, where

$$Micro\ Precision = \sum_{i=1}^{10} TP_i / (\sum_{i=1}^{10} TP_i + \sum_{i=1}^{10} FP_i),$$

$$Micro\ Recall = \sum_{i=1}^{10} TPi / (\sum_{i=1}^{10} TP_i + \sum_{i=1}^{10} FN_i),$$

$$Micro\ Specificity = \sum_{i=1}^{10} TNi / (\sum_{i=1}^{10} TN_i + \sum_{i=1}^{10} FP_i),$$

Table 4.5: Additional user measure results in ten SVM tests

n^{th}Run	Micro Precision	Micro Reall	Macro Precision	Macro Recall	Macro F
1	0.937	0.937	0.942	0.937	0.938
2	0.925	0.925	0.928	0.925	0.923
3	0.931	0.931	0.937	0.932	0.931
4	0.931	0.931	0.937	0.931	0.931
5	0.938	0.938	0.942	0.938	0.938
6	0.895	0.895	0.911	0.895	0.896
7	0.947	0.947	0.950	0.947	0.947
8	0.914	0.914	0.926	0.914	0.914
9	0.905	0.905	0.908	0.905	0.904
10	0.941	0.941	0.944	0.941	0.941
average	0.926	0.926	0.933	0.927	0.926

$$Micro\ Negativepredictive = \sum_{i=1}^{10} TNi/(\sum_{i=1}^{10} TN_i + \sum_{i=1}^{10} FN_i),$$

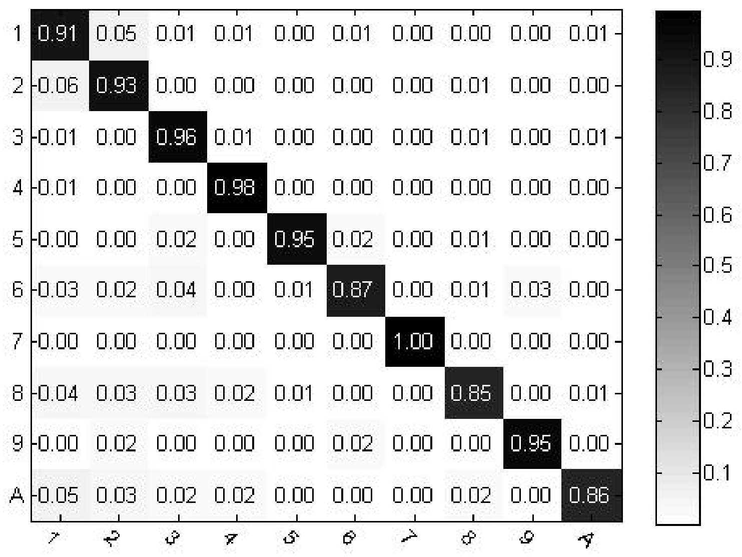
$$Macro\ Precision = \sum_{i=1}^{10} P_i/10,$$

$$Macro\ Recall = \sum_{i=1}^{10} R_i/10,$$

$$Macro\ F = \sum_{i=1}^{10} F_i/10.$$

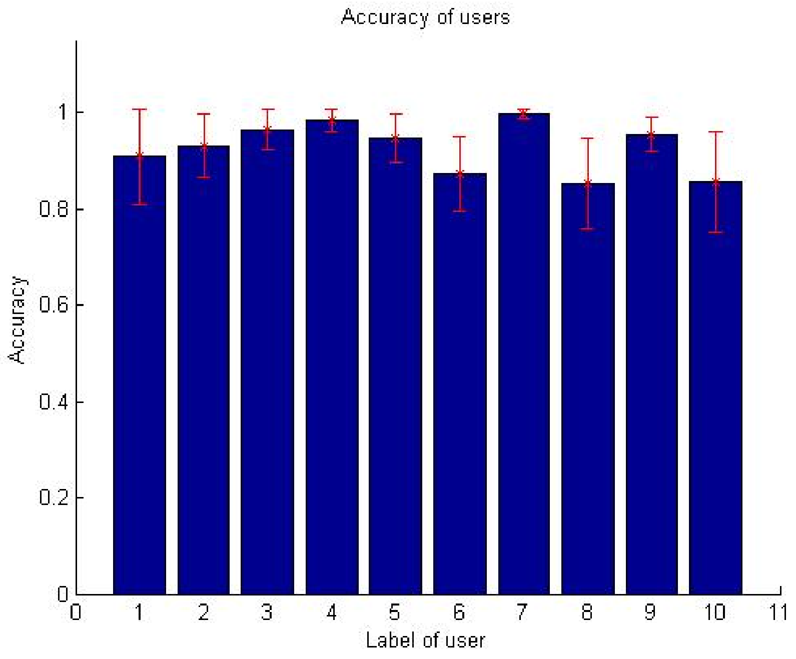
The average accuracy of the ten experiments conducted is 92.64%. Fig. 4.6 shows the confusion matrix results of the ten SVM tests and the average identification accuracy for each user. The matrix diagonal corresponds to correct classification assignments, and the total accuracy was calculated by dividing the sum of the matrix diagonal values by the testing sample number.

Table 4.6 lists the average accuracy results for all of the experiments. The results show that the combination of keystroke and mouse movement algorithms



	1	2	3	4	5	6	7	8	9	A
1	0.91	0.05	0.01	0.01	0.00	0.01	0.00	0.00	0.00	0.01
2	0.06	0.93	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
3	0.01	0.00	0.96	0.01	0.00	0.00	0.00	0.01	0.00	0.01
4	0.01	0.00	0.00	0.98	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.02	0.00	0.95	0.02	0.00	0.01	0.00	0.00
6	0.03	0.02	0.04	0.00	0.01	0.87	0.00	0.01	0.03	0.00
7	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
8	0.04	0.03	0.03	0.02	0.01	0.00	0.00	0.85	0.00	0.01
9	0.00	0.02	0.00	0.00	0.00	0.02	0.00	0.00	0.95	0.00
A	0.05	0.03	0.02	0.02	0.00	0.00	0.00	0.02	0.00	0.86

(a) The average confusion results.



(b) The average identification accuracy.

Figure 4.6: The average testing results of ten runs for each user $[C = 17, \gamma = 0.0000000945]$

provides significant progress in keystroke and mouse movement behavior identification. The proposed approach offers a competitive keystroke and mouse behavior identification solution. In the experiment, it required approximate 60 keystroke events to get 83.13% of accuracy rate. In general, it takes about 30 to 60 seconds for user to complete 60 character inputs. For mouse behavior analysis, it required approximate 40 effective mouse actions to get 80.25% of accuracy rate. The time of completing 40 effective actions is highly associated with the frequency of mouse operation, and might take 1 to 3 minutes.

Table 4.6: The comparison of various average accuracy results of the experiment

Exp. ID	Average Accuracy Train Set (%)	Average Accuracy Tuning Set (%)	Average Accuracy Test Set (%)	Feature Data Set Description
1	100	86.6	83.13	Keystroke features
2	100	78.5	80.25	95 mouse movement features
3	100	91.29	92.64	Combination keystroke and mouse movement features

4.5 Smartdevice Touch Panel Behavior Experiments

A total of ten users participated in the touch panel behavior experiment, and each was asked to interact with a Nexus Pad. A total of one hour of operational behavior was recorded, at intervals of approximately 0.01s. The number of pointing and sliding events were extracted for each user, and this information is listed in Table 4.7.

Table 4.7: Number of extracted events from each user

User ID	Point Events #	Sliding Events #
1	1228	155
2	1250	369
3	1115	418
4	1050	370
5	1456	583
6	2633	236
7	1749	362
8	1215	369
9	1106	342
10	1110	443

The extracted dataset was assigned randomly to the 15 instances for each user. The values from pointing features were most common. The values from sliding features were assigned a weight ω (in this experiment, $\omega = 0.25$). We combined these 15 pointing features with 40 weighted sliding features from each instance to form a dataset with 55 features. This method was used for learning and classification. A 10-fold cross-validation was performed to evaluate the proposed method. This thesis used 40% of the instances for training, 10% for tuning the parameters, and the remaining 50% for testing.

Various methods were used to evaluate our retrieval system. Table 4.8 lists the results from the users in the first SVM test. In this table "TP # " refers to the number of true positives, "FN # " refers to the number of false negatives, "FP # " refers to the number of false positives, and "TN # " refers to the number of true negatives. Additionally, accuracy A = (TP + TN) = (TP + FN + FP + TN), precision P = TP/(TP + FP), recall R = TP/(TP + FN), and F-measure F = 2PR/(P + R). Table 4.9 lists the results from the tenth SVM test of the unigram dataset. For FP, it represents the rate of misclassifying the incorrect user as the valid identity. In contrast, FN presents the rate of misclassifying the correct user as other identity. In practice, the severity of FP is relatively high due to allowing invalid user to get privilege of accessing sensitive data. Whereas, the affect of FN is just causing unnecessary inconvenience for user. By using secondary authentication approach, it can double check the user's identity and recover the access privilege.

Table 4.8: Results from various user measures in the ninth SVM test [(A: Accuracy, P: Precision, R: Recall, F: F-measure)]

User ID	TP #	FN #	FP #	TN #	A	P	R	F
1	7	0	0	72	1.00	1.00	1.00	1.00
2	9	0	1	69	0.99	0.90	1.00	0.95
3	5	1	0	73	0.99	1.00	0.83	0.91
4	9	0	0	70	1.00	1.00	1.00	1.00
5	8	0	0	71	1.00	1.00	1.00	1.00
6	8	0	0	71	1.00	1.00	1.00	1.00
7	9	0	1	69	0.99	0.90	1.00	0.95
8	6	0	0	73	1.00	1.00	1.00	1.00
9	7	1	0	71	0.99	1.00	0.88	0.93
10	9	0	0	70	1.00	1.00	1.00	1.00

The average accuracy (i.e., the average micro-precision) of the ten experiments was 98.6%. Figs. 4.7 and 4.8 show the confusion-matrix results from the ten SVM tests conducted, and the average identification accuracy for each user. The matrix diagonal corresponds to correct classification assignments, and the total accuracy was calculated by summing the values in this matrix diagonal, then dividing that sum by the number of testing samples. Figure 4.9 shows the average rejection ratio for each user. The rejection ratio is derived by dividing "TN no." by the total number of negatives. Table 4.10 lists the average accuracy results from the various experiments.

The results show that the combination of pointing features and weighted sliding features significantly improves pad-behavior identification. The proposed approach offers a competitive solution for user pad-behavior identification.

Table 4.9: Further measurement results for families in the tenth SVM test of the unigram dataset

n^{th} Run	Micro-precision	Micro-recall	Macro-precision	Macro-recall	Macro-F
1	1.000	1.000	1.000	1.000	1.000
2	1.000	1.000	1.000	1.000	1.000
3	0.987	0.987	0.986	0.986	0.985
4	0.959	0.959	0.961	0.965	0.962
5	0.986	0.986	0.986	0.989	0.986
6	1.000	1.000	1.000	1.000	1.000
7	0.987	0.987	0.989	0.990	0.989
8	0.963	0.963	0.966	0.965	0.964
9	0.975	0.975	0.980	0.971	0.974
10	1.000	1.000	1.000	1.000	1.000
Average	0.986	0.986	0.987	0.987	0.986

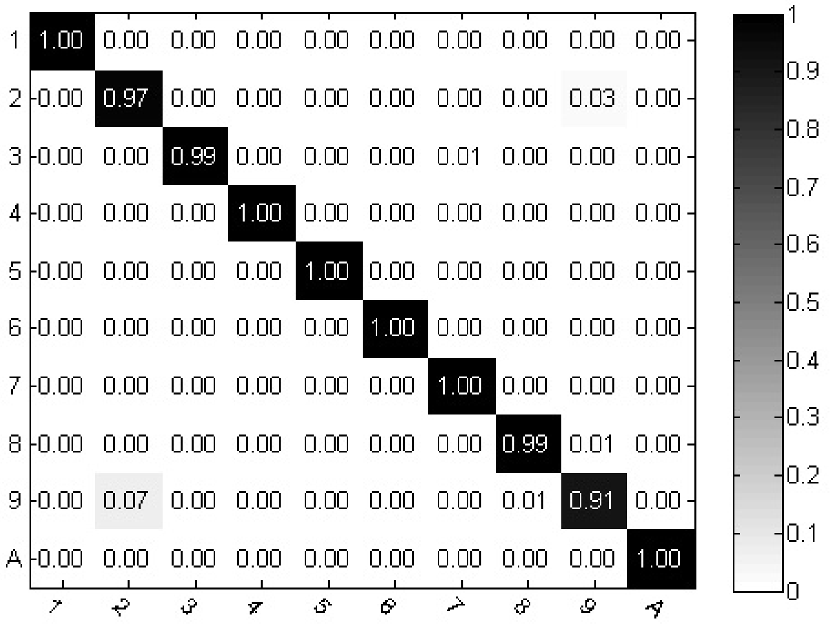


Figure 4.7: Average confusion results from the ten runs for each user [C = 11, γ = 2.793025]

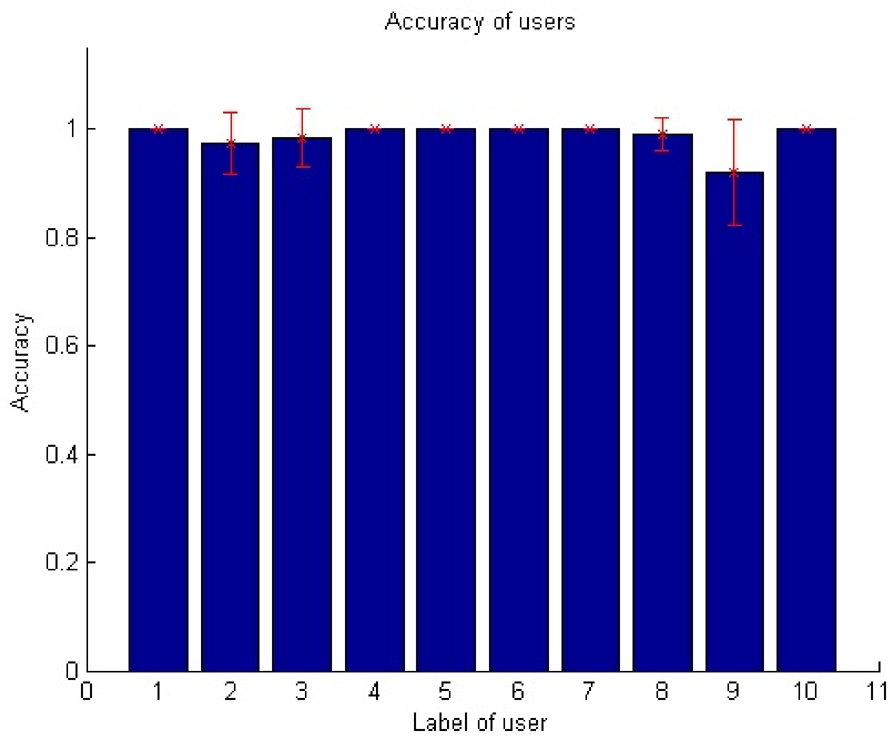


Figure 4.8: C = 11, γ = 2.793025

Table 4.10: Various averages in terms of accuracy

Exp.,ID	Average Accuracy of the Training Set (%)	Average Accuracy of the Tuning Set (%)	Average Accuracy of the Test Set (%)	Feature, Dataset Description
1	96.21	87.00	86.62	15 pointing events features
2	100	97.98	98.58	Combined 15 pointing features and 40 weighted sliding features

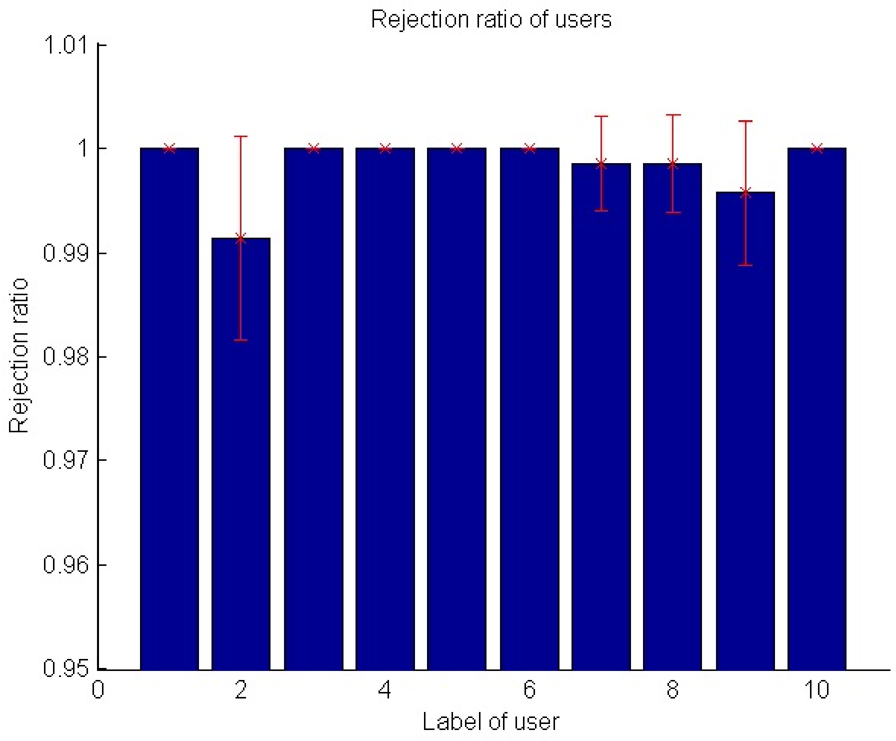


Figure 4.9: Average rejection ratio for each user

[C = 11, γ = 2.793025]

4.6 Discussion on Content-Based and Behavior-Based DLP Models

Table 4.11 illustrates the comparison between content-based and behavior-based DLP models. It is clear that the proposed behavior-based DLP model is able to obtain text streams from a file without parsing it. The central concept is to record a user's keystrokes online and then convert them to text streams. Therefore, more resources are required to perform the online analysis. On the contrary, the content-based DLP model requires that the file format is implemented manually. Since this method supports offline processing, it can be used to perform content inspection when a computer is idle. As it does not hook the APIs and analysis keystrokes timely, the required computing resources are less than that of the behavior-based DLP model.

Table 4.11: Comparison between content-based and behavior-based DLP model

Compared Items	Content-based DLP model	Behavior-based DLP model
File format parsing	Yes, required to implement file parser manually	No, the model is able to obtain text stream without file format parsing
Content inspection	Offline processing, can be applied when computer is idle	Online processing, more computing resources are required
Data creator identification	Can only identify one data creator via metadata	Can identify multiple data creators of a file via machine learning
Confidentiality detection	Determine only when content inspection starts	Timely; determine immediately after a file is created
APIs hook	No	Yes

On the other hand, current content-based DLP models can only determine the data creator via metadata. As metadata does not record every data creator identity, the content-based DLP model cannot provide more data beyond what is included. By using machine learning methods to analyze user keystroke and mouse movement behavior, the proposed behavior-based model solves this problem. It can identify multiple data creators of a file without relying on metadata. Compared with content-based DLP models, the proposed behavior-based DLP model can determine the confidentiality of a file (via the converted text streams and the identity of the file creator) shortly after the file is created. Therefore, when cooperating with a security gateway, this method can prevent the sensitive file from leaving the organization's network. To summarize, although the behavior-based DLP model requires more computing resources than the content-based DLP model, it does not require a file parser and is able to identify multiple data creators. Moreover, the behavior-based DLP model can provide robust data protection because it can determine the confidentiality of a file in a timely manner.

4.7 Abnormal User Behavior Experiments

The main objective of this section is to demonstrate the effectiveness and performance of *ChainSpot*. The first sub-section is a reconnaissance survey on the effectiveness of using *ChainSpot* to detect deviation from a normal case to an abnormal case for each account in each SOC ticket. The second experiment is a general performance evaluation of prediction ability when treating *ChainSpot*as an anomaly detection framework in terms of recall and prediction, compared to alternative baselines. The last sub-section explains what causes lead *ChainSpot* to successfully differentiate abnormal cases, and how these findings will be used as indicators of compromised data to benefit digital forensics teams.

4.7.1 Effectiveness of Measuring Behavior Deviation Using *ChainSpot*

This experiment is a reconnaissance survey to check if each Markov model of an abnormal account results in a larger deviation, compared to that account's benign model and normal cases. For each account that is ever labeled as "abnormal" by SOC event tickets, three phases of log data with different significances and usages can be generated:

1) "Abnormal testing phase": this log data phase consists of the three days before the date that the SOC ticket was signed. A corresponding abnormal Markov chain, named $M^i_{abnor.}$, is used to represent the i^{th} account's abnormal behaviors.

2) "Training phase": this log data phase for one account is the former half of the whole log, excluding the logs of the "abnormal testing phase". Note that behaviors from this phase are all assumed to be benign, and notation $M^i_{tr.}$, represents the Markov chain for this phase of the i^{th} account.

3) "Normal testing phase": this phase consists of all logs, excluding those of the first two phases, and these behaviors are also assumed to be benign. Markov chain $M^i_{nor.}$ is used to summarize the i^{th} account's behaviors in this phase.

The successful condition in following reconnaissance survey over total E accounts labeled by SOC tickets is regarded as follows:

$$\begin{aligned}&\forall\ i = 1, ..., E,\ i^{th} \text{ account is regarded as:}\\&Successful,\ \text{If } GED(M^i_{abnor.}, M^i_{tr.}) > GED(M^i_{nor.}, M^i_{tr.}).\\&Failed,\ \text{otherwise.}\end{aligned} \tag{4.1}$$

Table 4.12 lists all accounts labeled by SOC tickets in terms of different event names, as well as corresponding success rates. In the real dataset, each SOC ticket index indicates a different anomaly or attack and leads to a various number of abnormal accounts. Additionally, different kinds of logs will link to different types of attacks. For instance, AD logs link tothe second type of SOC tickets ("Multiple accounts logon from single IP in short time") and result in 865 accounts, while proxy logs mainly contribute 255 accounts of the third type of SOC tickets ("Host connects to malicious domain") . Table 4.12 shows that *ChainSpot* is significantly more effective compared to the uniform baseline (about 50.00%), except in cases of the first type of tickets ("Single account failed too many times when logging on") tickets (50.00%). Measuring general effectiveness results in 85.66% and 87.17% success rates for averaging various types and averaging different accounts, respectively.

Table 4.12: Reconnaissance survey for effectiveness of *ChainSpot*

AD logs related			
Ticket index	#Related Accounts	#Succ. Accounts	Succ. Rate
1^{st}	2	1	50.00%
2^{nd}	865	791	91.45%
5^{th}	3	3	100.00%
6^{th}	2	2	100.00%
Proxy logs related			
Ticket index	#Related Accounts	#Succ. Accounts	Succ. Rate
3^{rd}	255	185	72.50%
4^{th}	3	3	100.00%

(The explanation for each SOC ticket index can be found in table 3.4.)

Another useful observation from the organization where ChainSpot was deployed can be reasonably leveraged to narrow the set of accounts being detected and to improve the current effectiveness of *ChainSpot*. The dynamic host configuration protocol (DHCP) used in the experiment organization allows users to dynamically login from different intranet source IPs. Thus, using the "source IP" field of SOC tickets to correlate suspicious accounts might unintentionally include some innocent accounts that happened to be logging in on the same IP as the abnormal accounts. To rule out these mistakes, a threshold "adhesion degree" is applied to measure the extent that a user routinely uses the same IP to log on or browse the web. Accounts that do not meet the pre-set "adhesion degree" requirement will be filtered out. Table 4.13 and Table 4.14 list the major changes to the quantities of accounts related to some SOC tickets, the filtering results, and the corresponding successful accounts using different "adhesion degree" criteria. Table 4.13 incorporates the probability ratio of users' most used IPs, while Table 4.14 calculates entropy based on all IPs that a user has ever used, measuring "adhesion degree" . Generally speaking, applying "adhesion degree"thresholds improves the success rate in spite of the fact that different "adhesion degree" calculations mainly contribute to different SOC ticket types. Using the "adhesion degree" filtering strategy is essentially another trade-off situation between recall and precision. Accounts with a lower adhesion degree will be ruled out as the threshold increases. However, this will lead to a higher level of confidence in the judgement of *ChainSpot* and the success rate will be improved. Figure 4.10 focuses on security events containing most accounts, and further depicts performance changes when applying different threshold settings. Users of *ChainSpot* can understand the trend and convergence of performance trade-offs according to Figure 4.10, and can decide the appropriate "adhesion degree" threshold value based on their preference between precision and recall.

Table 4.13: The major changes in quantities when applying probability ratio filtering strategy

AD logs related			
	Threshold = 0.0	Threshold = 0.7	Threshold = 0.9
Ticket index	Succ. Rate	Succ. Rate	Succ. Rate
1st	1/2	1/2	1/2
2nd	791/865	552/588	212/220
5th	3/3	3/3	3/3
6th	2/2	1/1	1/1

Proxy logs related			
	Threshold = 0.0	Threshold = 0.7	Threshold = 0.9
Ticket index	Succ. Rate	Succ. Rate	Succ. Rate
3rd	185/255	108/130	48/51
4th	3/3	1/1	1/1

Table 4.14: The major changes on quantities when applying entropy filtering strategy

AD logs related			
	Threshold = 1.0	Threshold = 0.4	Threshold = 0.2
Ticket index	Succ. Rate	Succ. Rate	Succ. Rate
1st	1/2	1/2	1/2
2nd	791/865	656/700	377/392
5th	3/3	3/3	3/3
6th	2/2	2/2	2/2

Proxy logs related			
	Threshold = 1.0	Threshold = 0.4	Threshold = 0.2
Ticket index	Succ. Rate	Succ. Rate	Succ. Rate
3rd	185/255	168/237	46/58
4th	3/3	1/1	1/1

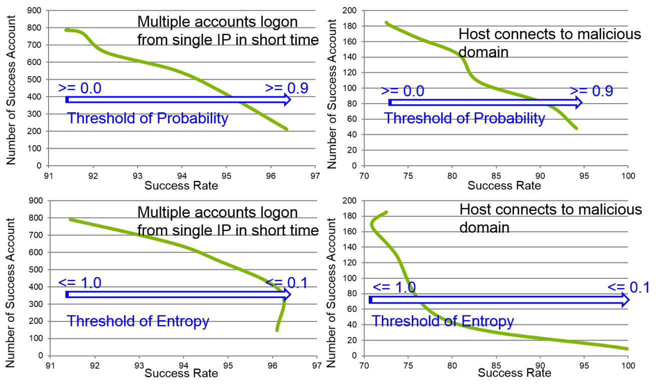


Figure 4.10: Effectiveness changing given different filtering thresholds

4.7.2 Performance of Anomaly Detection Using *ChainSpot*

The ultimate goals of *ChainSpot*are to build the behavioral model for each account and to detect anomalies when accounts do not act like themselves, compared to their historical daily routine. For this purpose, we formalize the following experiment as a training-detection scenario. In total, the collected dataset comprises 1,089 accounts, and three Markov chains were built to be instances of different usages for each account's profile. As mentioned in subsection IV, Part A, each account's profile logs in the "training phase" lead to one benign Markov chain equivalent to one training instance, while Markov chains from the "abnormal testing phase" and "normal testing phase" result in two testing instances, one for the positive (abnormal) case and the other for the negative (normal) case. It should be noted that the usage of *ChainSpot* in this scenario is different than its usage in previous experiments. In the training-detection framework, the label ("abnormal"(negative) or "normal") of each testing instance is obviously by default, such that equation (4.1) is no longer appropriate. The following equation is therefore defined to help *ChainSpot* make predictions when unknown instances arrive. $\forall i = 1, ..., E$, given an unseen Markov chain M^i_{unseen} which is known to belong to the i^{th} account, the predicted label of M^i_{unseen} is classified as:

$$\begin{aligned} &Abnormal, \text{ once } GED(M^i_{tr.}, M^i_{unseen}) \geq \delta, \\ &Normal, \text{ otherwise,} \end{aligned} \tag{4.2}$$

where $M^i_{tr.}$ is the Markov chain representing the i^{th} account's historically benign model, $GED(\cdot)$ is the graphical edit distance defined in equation (3.16), and δ is a user-defined threshold. The idea of using equation (4.2) as an anomaly detection mechanism is relatively intuitive. Once the graph edit distance between $M^i_{tr.}$ and M^i_{unseen} is larger than δ, it means that the unseen behaviors of the i^{th} account deviate from the same account's historical pattern to an unallowable extent. In this condition, an anomaly alert should be triggered.

In this anomaly detection experiment, there are a total of 1,089 abnormal (positive) instances, and 1,089 normal (negative) instances. Table 4.15 shows the performance results of *ChainSpot* under different deviation thresholds δ of 4, 6, and 8. Taking $\delta = 6$ seems to result in a generally well-balanced prediction of 0.71 *Precision* and 0.81 *Recall* rates. Based on this, Table 4.16 then compares *ChainSpot* where $\delta = 6$ with three alternative baselines, including predictors that always output "positive", always output "negative", and output randomly. It can be clearly observed that the performance of *ChainSpot* significantly outperforms performances of these simple baselines.

It should be noted that different settings of δ will gives different preferences between *Precison* and *Recall*, and "cost curve" is an efficient technique for evaluating different prediction methods in a cost-sensitive manner [41]. Figure 4.11 depicts a cost curve that gives a suggestion for how to select an appropriate δ value when *ChainSpot* deploys. For a binary prediction problem like this experiment, the cost curve considers the performances of each candidate predictor (δ = 4, 6, and 8) on the following two dimensions:

x-**axis** : "Probability cost" ($PC(+)$) is defined by combining the penalty costs of a wrong prediction and data distribution for each class:

$$PC(+) = \frac{p(+)cost(-|+)}{p(+)cost(-|+) + (1 - p(+))cost(+|-)}, \tag{4.3}$$

where $p(+)$ is the probability of positive samples, $cost(-|+)$ and $cost(+|-)$ are penalty costs of false negative and false positive cases.

y-**axis** : "Normalized expected cost" (NEC), defined as following, indicates expected predictor performance:

$$NEC = Rate_{FN} * PC(+) + Rate_{FP} * (1 - PC(+)), \tag{4.4}$$

where $Rate_{FN}$ and $Rate_{FP}$ are the ratios belonging to $[0, 1]$ where false negative and false positive errors occur. It should be noted that the smaller the value of NEC is, the better performance the candidate predictor can provide.

When deploying *ChainSpot* to a certain organization, the members of the security team should consider the value of $PC(+)$ for their own environment based on the frequencies of abnormal security events, as well as the cost when a security event occurs. For instance, according to Figure 4.11, a suggestion of δ value is that:

$$\begin{cases} \delta = 4, \text{ if } PC(+) \leq 0.33, \\ \delta = 6, \text{ if } 0.33 < PC(+) \leq 0.67, \\ \delta = 8, \text{ if } 0.67 < PC(+), \end{cases} \tag{4.5}$$

where this strategy provides an expected performance of $NEC \leq 0.28$

Table 4.15: Performance of anomaly detection under different deviation thresholds

Measurements	$\delta = 4$	$\delta = 6$	$\delta = 8$
$\#TP$	1050	882	645
$\#FP$	674	362	218
$\#TN$	415	727	871
$\#FN$	39	207	444
$Precision$	0.61	0.71	0.75
$Recall$	0.96	0.81	0.59

Table 4.16: Performance comparison between *ChainSpot* and three baselines

Measurements	*ChainSpot* with $\delta = 6$	always "Pos."	always "Neg."	Randomly output
$\#TP$	882	1089	0	546
$\#FP$	362	1089	0	541
$\#TN$	727	0	1089	548
$\#FN$	207	0	1089	543
$Precision$	0.71	0.50	Null	0.50
$Recall$	0.81	1.0	0	0.50

4.7.3 Representative Demonstration and Case Study of *ChainSpot*

The final part of the experiment consists of demonstrative case studies that

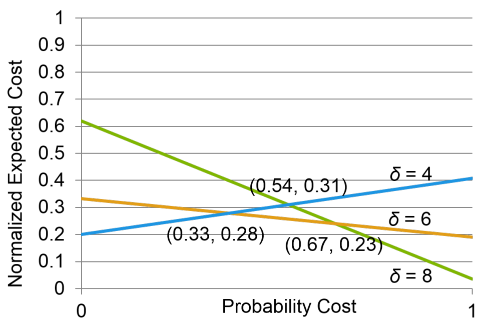


Figure 4.11: Cost curve describing distributions of normalized expected cost under different δ and probability costs

show which essentials *ChainSpot* will catch to differentiate normal and abnormal cases, and how these essentials found by *ChainSpot* will benefit domain experts as new sources of forensics knowledge. The following demonstrations on AD and proxy logs also illustrate how the mechanism of *ChainSpot* works, and that the mechanism is quite different from common existing methods. When deploying in real circumstances, *ChainSpot* can be a complementary threat-detecting technique and can be used concurrently with other solutions.

Case I - Detecting "Multiple accounts logon from single IP in short time" on AD logs

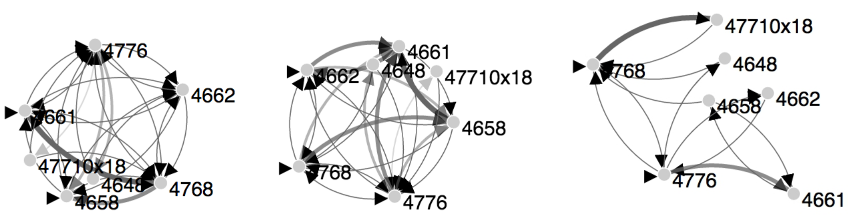
The 2^{nd} event - "Multiple accounts logon from single IP in short time" indicates a policy violation that occurs when too many accounts log on from an outside IP within a short duration, and it is usually prognostic and followed by account compromise. The first discussion is on the representative account, A_{case1} of this abnormality. Figure 4.12 depicts the three corresponding patterns summarized from sequential behaviors recorded in the AD logs. Each Markov chain is dedicated to user's behaviors in one phase. Vertices represent event codes included in the Windows Active directory protocol [61], while directional edges indicate the transition probability among different events. The wider and darker an edge is depicted, the stronger probability of this transition. In general, by comparing all patterns (Markov chains) in Figure 4.12, it can be clearly seen that the structural difference between the training phase (Figure 4.12(a)) and abnormal instance (figure 4.12(c))is much larger than that between the training phase and normal instance (Figure 4.12(b)).Further, after deep investigation into the main differences leading to *ChainsSpot*'s predictions in terms of causalities in vertices and edges, behaviors that constitute an abnormal pattern have the following characteristics that differ from the training and normal testing phases:

1) Without "4661" to "4662": the event "4661" means "a session handler to an object was requested", while "4662" represents "an operation was performed on a requested object".

2) Without "4662" to "4658": the event "4658" means "a session handler to a requested object was released".

3) With loops from "4768" to "4771 0x18": the event "4768" means "a Kerberos authentication ticket was requested", and the "4771 0x18" occurs when authentication fails because of entering a wrong password.

Combining the original meaning of the original event, the above example shows that attackers try to log on using many intranet accounts, one by one, in order to gain access permission to a certain object or service in this organization. This kind of intelligence can help cyber security team members recommend that all intranet users review their password complexity, as well as enhance protection on certain assets that are targeted by attackers with additional information about what resources attract attackers' interest.



(a) (b) (c)

Figure 4.12: Patterns of user A_{case1} who participated in "Multiple accounts logon from single IP in short time" event

Case II - Detecting "Host connects to malicious domain" on proxy logs

Another illustration is about *ChainSpot* detecting an account A_{case2} engaging in the event of "Host connects to malicious domain" based on behavioral patterns summarized from proxy logs. Again, a Markov chains representing the data of A_{case2}'s training, normal testing, and abnormal testing phases was built by *ChianSpot*. According to subsection III.Part A, the states and transitions in proxy Markov chains of *ChainSpot* take the following form: **From "Get and Download from google.com then failed" to "Get and Upload from facebook.com then failed"**. By means of investigating vertices and transitions from three-phase Markov chains, Figure 4.13 shows the accumulated counts of domains that A_{case2} accessed in the abnormal phase, but never during the other two phases. The x-axis of Figure 4.13 lists these domains that A_{case2} accessed during the abnormal stage, but never in the other two phases. The y-axis shows the accumulated counts of access to corresponding domains. It should be noted that domain "moneydj.com" has been proven as a malicious website, and never occurs in A_{case2}'s training and normal profiles. Further, implicit behavioral anomalies can also be detected by comparing the state constitutions of different A_{case2}'s profiles. Figures 4.14(a) and 4.14(b) show the accumulated visit count of common domains that A_{case2} accessed during both the training and normal testing stages. It can clearly be noted that the five most frequently visited domains are 1) "amazon.com", 2) appledaily.com.tw, 3) chartbeat.net, 4) edgesuit.net and 5) facebook.com. However, none of these well-known social platforms, news sites, or common websites were being accessed during tA_{case2}'s abnormal phase. This is a highly suspicious situation and the security team should carefully investigate this anomaly to check if A_{case2} was already compromised.

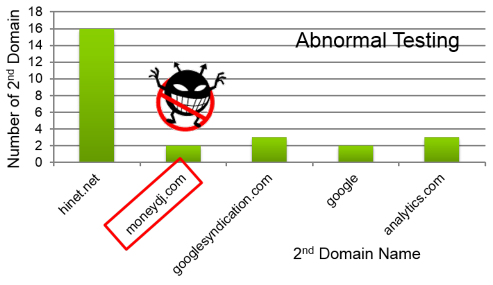
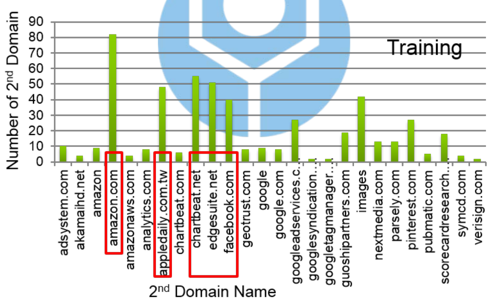
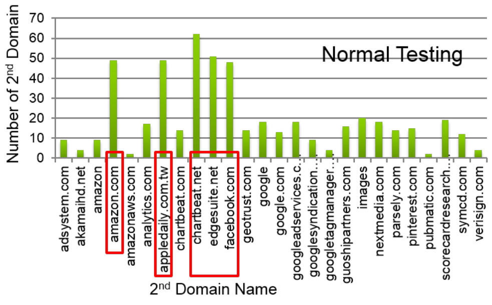


Figure 4.13: Access records extracted from abnormal-phase Markov chain of account A_{case2} who participated in "Host connects to malicious domain" event



(a)



(b)

Figure 4.14: Access records extracted from Markov chains of account A_{case2} who participated in "Host connects to malicious domain" event

Chapter 5 CONCLUSIONS

Based on keystroke and mouse movement profiling, this thesis proposed an active behavior-based DLP model to avoid the capability limitation of parsing various formats to detect sensitive data in file contents. This model makes it possible to identify the data creator by recording and analyzing his/her keystroke and mouse movement behavior. This combination approach provides high data visibility, helps determine the identity of the data creators, and is compatible with current DLP systems. Using the proposed approach to classify the user's identity while operating any document in the device, it is possible to (1) check the user's right to protect sensitive files from being illegitimately or surreptitiously sent out, (2) verify ownership to prevent critical contents from being arbitrarily modified or intentionally altered, and (3) lock the device to avoid covert usage by a fake user.

The framework implementation and experiment results show that the pro- posed model performs well with prevalent technologies. To the best of our knowledge, this is a novel method to lessen the burden on DLP systems to develop new file format parsers. At the same time, it provides existing DLP systems with the capability to determine the identity of data creators with an accuracy of 92.64%. The following issues still exist and need to be resolved: (1) a user's behavior may vary depending on different keyboard and mouse hardware, so incremental learning should takes place whenever new behavior instances emerge; (2) the SKA cannot detect sensitive text that is not typed linearly, e.g., text copied from elsewhere or generated from a form through the auto-fill feature [65]. The resolution of these issues requires further research. Nevertheless, we believe that our proposed DLP model and behavior identification are important innovations in the DLP domain.

Based on user keystrokes and gesture profiling, this thesis proposed a novel extraction model for pointing events and sliding operations by users, and combined this model with a SVM algorithm to identify users through behavioral biometrics. Our model makes it possible to identify the data creator by recording and analyzing his or her pointing and sliding behavior on a touch device. This combined approach provides high data visibility, an improved ability to identify data creators, and compatibility with current data leakage prevention (DLP) systems. In case of remote VPN connection, Secure Keystream Analyzer (SKA) could be downloaded into client host through AD dispatch. Thus, the proposed approach can still work no matter how its connection.

To mitigate the blind spot in relying on human forensics or traditional techniques, this thesis first proposes a cyber threat detection framework, *ChainSpot*, focusing on both application-layer AD and proxy sequential log analysis. The novelty of *ChainSpot* mainly comes from building graphical patterns by summarizing a user's application-layer sequential behaviors as Markov chains, and from monitoring deviations from one's normal behavior. All practical issues with respect to implementing *ChainSpot*, such as trade-offs between choosing enough features and keeping computation simple, were carefully addressed with appropriate domain knowledge. Experiment results show that *ChainSpot* is able to alleviate a large amount of human effort in monitoring or forensics. Furthermore, case demonstrations show that the differences between benign and suspicious patterns could be organized as threat intelligence and used for describing attack scenarios.

Future work of this thesis is manifested as follows:

1. Combine and align heterogeneous logs, such as active directory, proxy, and firewall, into a super timeline, and to try to model the benign and malicious patterns.

2. Extract additional content-features, such as histogram of using word or frequent information, for cross-validating the identity.

3. Analyze and select the major features of switch time cost that can gain high effectiveness.

4. Propose a continuous training mechanism for model updating.

5. Leverage physical recognition mechanism, such as biometrics, passive face recognition etc., for double check.

6. Align more heterogeneous logs into super timeline to classify in-depth
7. Combine both of the user behavioral profiles in client and server sides to extend the visibility
8. Apply to various application domains, such as user authentication and data protection on mobile device for insurance broker.

REFERENCES

[1] A. Shabtai, Y. Elovici, and L. Rokach, *A Survey of Data Leakage Detection and Prevention Solutions,* Springer, Berlin, 2012.

[2] V. Stamati-Koromina, C. Ilioudis, R. E. Overill, C. K. Georgiadis, and D. Stamatis, *Insider threats in corporate environments: a case study for data leakage prevention,* in Proceedings of the 5th Balkan Conference in Informatics, 2012, pp. 271-274.

[3] Symantec, Inc., *Internet security threat report, 2011 trends,* http://www.symantec. com/threatreport/, 2012.

[4] Verizon Communications, *2012 data breach investigations report,* http://www.verizonenterprise. com/DBIR/2012/, 2012.

[5] DataLossDB, Open Security Foundation, *Data loss statistics,* http://datalossdb.org/statistics, 2013

[6] Websense, *Unified data loss prevention for gateways, endpoints and discovery,* http://www.websense.com/assets/datasheets/datasheet-data-security-suite-en.pdf, 2013.

[7] eMarketer, *Smartphone users worldwide will total 1.75 billion in 2014,* http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536, January 2014.

[8] Gartner, *Gartner says smartphone sales surpassed one billion units in 2014,* http://www.gartner.com/newsroom/id/2996817, March 2015.

[9] S. Schleimer, D. S. Wilkerson, and A. Aiken, *Winnowing: Local algorithms for document fingerprinting,* in Proceedings of ACM SIGMOD International Conference on Management of Data, 2003, pp. 76-85.

[10] H. Takabi, J. B. D. Joshi, and G. J. Ahn, *Security and privacy challenges in cloud computing environments,* IEEE Security and Privacy, Vol. 8, 2010, pp. 24-31.

[11] M. Cooperation, *Office binary file formats (for Word, Excel and PowerPoint),* http://download.microsoft.com/download/2/4/8/24862317-78F0-4C4B-B355-C7B2C1D997DB/OfficeFileFormatsProtocols.zip, 2008.

[12] Wikipedia, *File format,* http://en.wikipedia.org/wiki/File_format, 2013.

[13] M. S. Pera and Y. K. Ng, *Simpad: A word-similarity sentence-based plagiarism detection tool on web documents,* Web Intelligence and Agent Systems, Vol. 9, 2011, pp. 27-41.

[14] S. Hariharan, *Automatic plagiarism detection using similarity analysis,* The International Arab Journal of Information Technology, Vol. 9, 2012, pp. 322-326.

[15] C. C. Lin, C. C. Chang, and D. Liang, *A new non-intrusive authentication approach for data protection based on mouse dynamics,* in Proceedings of International Symposium on Biometrics and Security Technologies, 2012, pp. 9-14.

[16] H.K. Pao, H.Y. Lin, K.T. Chen, and J. Fadlil, *Trajectory based behavior analysis for user verification,* in proceeding of IDEAL"10 Proceedings of the 11th international conference on Intelligent data engineering and automated learning, pp. 316-323, 2010.

[17] Y. Z. Hui Xu and M. R. Lyu. *Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones,* In Symposium On Usable Privacy and Security (SOUPS 2014), 2014.

[18] Y. Zhang, P. Xia, J. Luo, Z. Ling, B. Liu, and X. Fu, *Fingerprint attack against touch-enabled devices,* In Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, 2012.

[19] M. Frank, R. Biedert and D. Song, *Touchalytics: On the applicability of touch-screen input as a behavioral biometric for continuous authentication,* In IEEE Transactions on Information Forensics and Security, vol. 8, no. 1, pages 136–148, 2013.

[20] Trend Micro Inc., *M-Trends 2015: A VIEW FROM THE FRONT LINES,* 2015.

[21] Trend Micro Inc., *Trend micro white paper on advanced persistent threat(apt),* 2013.

[22] E. Ouellet, *Magic quadrant for content-aware data loss prevention,* Technique Report No. G00224160, Gartner, Inc., 2013.

[23] RSA, The Security Division of EMC Corporation, *RSA data loss prevention suite,* http://www.rsa.com/products/DLP/sb/9104n DLPSTn SBn 0311.pdf, 2010.

[24] McAfee, *Mcafee total protection for data loss prevention,* Technical Report, http://www.mcafee.com/au/resources/solution-briefs/sb-total-protection-for-dlp.pdf, McAfee,Inc., 2012.

[25] amXecure, *RSA data loss prevention suite,* http://www.amxecure.com/index.php/zh/siem/453-privacyid, 2013.

[26] Paloalto Networks, *Preventing data leaks at the firewall,* http://www.paloaltonetworks.com/literature/whitepapers/, 2008.

[27] W. Meng, D. S. Wong and J. Zhou, *Surveying the development of biometric user authentication on mobile phones,* In Communications Surveys & Tutorials, IEEE, vol. 17, no. 3, pages 1268–1293, 2014.

[28] Y. Meng and L.-F. Kwok, *Design of touch dynamics based user authentication with an adaptive mechanism on mobile phones,* In Proceeding SAC '14 Proceedings of the 29th Annual ACM Symposium on Applied Computing, pages 1680–1687, 2014.

[29] M. Antal and L. szl'o Zsolt Szab'o, *An evaluation of one-class and two-class classification algorithms for keystroke dynamics authentication on mobile devices,* http://www.ms.sapientia.ro/ manyi/research/43.pdf, 2015.

[30] L. Cai and H. Chen, *Touchlogger: Inferring keystrokes on touch screen from smartphone motion,* In Proceedings of the 6th USENIX conference on Hot topics in security, 2011.

[31] D. Buschek and F. Alt, *Improving accuracy, applicability and usability of keystroke biometrics on mobile touchscreen devices,* In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2015.

[32] B. Draffin and J. Zhang, *Keysens: Passive user authentication through micro-behavior modeling of soft keyboard interaction,* In 5th International Conference, MobiCASE 2013, 2013.

[33] H. Xu and M. R. Lyu, *Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones,* In Symposium On Usable Privacy and Security (SOUPS 2014), 2014.

[34] L. Li and G. Xue, *Unobservable re-authentication for smartphones,* In NDSS, The Internet Society, 2013.

[35] W. Meng and L.-F. Kwok, *The effect of adaptive mechanism on behavioural biometric based mobile phone authentication,* In Information Management & Computer Security, vol. 22, no. 2, pages 155–166, 2014.

[36] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda, *Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks,* in Proceedings of the 29th Annual Computer Security Applications Conference. ACM, 2013, pp. 199–208.

[37] T.-F. Yen, V. Heorhiadi, A. Oprea, M. K. Reiter, and A. Juels, *An epidemiological study of malware encounters in a large enterprise,* in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014, pp. 1117–1130.

[38] A. Oprea, Z. Li, T.-F. Yen, S. H. Chin, and S. Alrwais, *Detection of early-stage enterprise infection by mining large-scale log data,* in Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on. IEEE, 2015, pp. 45–56.

[39] L. Invernizzi, S. Miskovic, R. Torres, C. Kruegel, S. Saha, G. Vigna, S.-J. Lee, and M. Mellia, *Nazca: Detecting malware distribution in large-scale networks.* in NDSS, vol. 14, 2014, pp. 23–26.

[40] H.-K. Pao, C.-H. Mao, H.-M. Lee, C.-D. Chen, and C. Faloutsos, *An intrinsic graphical signature based on alert correlation analysis for intrusion detection,* in Technologies and Applications of Artificial Intelligence (TAAI), 2010 International Conference on. IEEE, 2010, pp. 102–109.

[41] R. C. Holte and C. Drummond, *Cost-sensitive classifier evaluation using cost curves,* in Advances in Knowledge Discovery and Data Mining. Springer, 2008, pp. 26–29.

[42] K. Revett, F. Gorunescu, M. Gorunescu, M. Ene, S. T. de Magalhaaes, and H. M. D. Santos, *A machine learning approach to keystroke dynamics based user authentication,* International Journal of Electronic Security and Digital Forensics, Vol. 1, 2007, pp. 55-70.

[43] Y. Zhao, *Learning user keystroke patterns for authentication,* World Academy of Science, Engineering and Technology, Vol. 14, 2008, pp. 739-744.

[44] I. Traore, I. Woungang, S. Obaidat, Y. Nakkabi, and I. Lai, *Combining mouse and keystroke dynamics biometrics for risk-based authentication in web environments,* in Proceedings of International Conference on Digital Human, 2012, pp. 138-145.

[45] S. P. Banerjee and D. L. Woodard, *Biometric authentication and identification using keystroke dynamics: A survey,* Journal of Pattern Recognition Research, 2012, pp. 116-139.

[46] C. Cortes and V. Vapnik, *Support-vector network,* Machine Learning, Vol. 20, 1995, pp. 273-297.

[47] C. W. Hsu and C. J. Lin, *A comparison of methods for multiclass support vector machines,* IEEE Transactions on Neural Networks, Vol. 13, 2002, pp. 415-425.

[48] P. S. Teh, A. B. J. Teoh, and S. Yue, *A survey of keystroke dynamics biometrics,* The Scientific World Journal, Vol. 2013, Article ID 408280, 2013.

[49] E. Yu and S. Cho, *Keystroke dynamics identity verification its problems and practical solutions,* Computers and Security, Vol. 23, 2004, pp. 428-440.

[50] C. C. Chang, *LIBSVM, a library for support vector machines,* http://www.csie.ntu.edu.tw/cjlin/libsvm, 2012.

[51] J. Wu, C. Lin, S. Chong and Y. Lee, *Keystroke and mouse movement profiling for data loss prevention,* In Journal of Information Science and Engineering, pages 23–42, 2015.

[52] Microsoft, MSDN Library, *Directory system agent,* 2014, [Online; accessed: 6-May-2014]. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms675902(v=vs.85).aspx

[53] M. E. Russinovich and D. A. Solomon, *Microsoft Windows Internals: Microsoft Windows Server (TM) 2003, Windows XP, and Windows 2000 (Pro-Developer).* Microsoft Press, 2004.

[54] Microsoft, *Active directory collection: Active directory on a windows server 2003 network,* Microsoft, TechNet Library, Tech. Rep., 2015, [Online; accessed: 6-May-2015]. [Online]. Available: https://technet.microsoft.com/en-us/library/cc780036(WS.10).aspx

[55] Microsoft, *How the kerberos version 5 authentication protocol works,* Microsoft, TechNet Library, Tech. Rep., 2015, [Online; accessed: 6-May-2015]. [Online]. Available: https://technet.microsoft.com/en-us/library/cc772815(v=ws.10).aspx

[56] M. Shapiro, *Structure and encapsulation in distributed systems: the proxy principle,* in icdcs, 1986, pp. 198–204.

[57] J. R. Norris, *Markov chains.,* Cambridge university press, 1998, no. 2.

[58] L. Rabiner, *A tutorial on hidden markov models and selected applications in speech recognition,* Proceedings of the IEEE, vol. 77, no. 2, pp. 257–286, 1989.

[59] M.-Y. Chen, A. Kundu, and J. Zhou, *Off-line handwritten word recognition using a hidden markov model type stochastic network,* Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 16, no. 5, pp. 481–496, 1994.

[60] A. D. Wilson and A. F. Bobick, "Parametric hidden markov models for gesture recognition," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 21, no. 9, pp. 884–900, 1999.

[61] Monterey Technology Group, Inc., *Windows security log events,* 17-April-2016. [Online]. Available: https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/Default.aspx

[62] Monterey Technology Group, Inc., *Event code 4771: Kerberos pre-authentication failed,* 17-April-2016. [Online]. Available: https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=4771

[63] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval,* McGraw-Hill, 1986.

[64] N. Zheng, A. Paloski, and H. N. Wang, *An efficient user verification system via mouse movements,* in Proceedings of the 18th ACM Conference on Computer and Communications Security, 2011, pp. 139-150.

[65] T. Wuchner and A. Pretschner, *Data loss prevention based on data-driven usage control,* in Proceedings of IEEE 23rd International Symposium on Software Reliability Engineering, 2012, pp. 151-160.

[66] P. A. Networks, *Preventing data leaks at the firewall,* http://www.paloaltonetworks.com/literature/whitepapers/, 2008.

Appendix A: Example for Behavior Log File

列表 C.1: An example for a keystroke log file

```
062942960 M 062943260 r 062943611 . 062943921 # 062944492 L 062944823
    a 062945103 w 062945313 # 062945874 w 062946154 i 062946515 l
   062946725 l 062946986 # 062947416 a 062947637 r 062947797 r
   06294877 i 062948247 v 062948478 e 062948628 # 062948848 i
   06294919 n 062949229 # 062949780 8 0629500 0 062950220 #
   062951362 M 062951612 i 062951783 n 062952203 , 062952884 #
   062953565 h 062953735 i 062953886 s 06295466 # 062954466 p
   062954657 h 062954857 o 06295587 n 062955318 e 062955518 #
   06295659 n 062956329 u 062956609 m 062956860 b 062957150 e
   062957411 r 062957591 # 062957811 i 062957931 s 062958132 #
   062958712 0 0629593 9 062959464 5 062959714 4 063001486 -
   063001707 2 063001947 3 063002147 4 063002428 - 063003650 4
   063003850 3 063004220 7 063004951 , 063005182 # 063005452 e
   063005703 m 063006273 a 063006484 i 063006694 l 063007125 #
   063007335 i 063007505 s 063007715 # 063008346 M 063008657 i
   063008827 n 063009398 H 063010249 w 063010539 a 063011491 9
   063011891 4 063012202 8 063012723 @ 063013664 g 063014125 m
   063014285 a 063014415 i 063014625 l 063014846 . 063014986 c
   063015126 o 063015296 m 063016128 , 063016308 # 063016538 p
   063016809 l 0630179 e 063017229 a 063017449 s 063017680 e
   06301870 # 063018431 c 063018571 o 063018761 n 063019482 t
   063019853 a 063020143 c 063020474 t 06302175 # 063021475 w
   063021605 i 063021756 t 063021896 h 063022126 # 063022467 h
   063022937 i 063023198 m 063023458 . 063023929
```

列表 C.2: An example for a mouse movement event log file

```
SCREEN Resolution: 1366 * 768
X:404,Y=129,Wheel:0 [635356812944212037]
X:404,Y=134,Wheel:0 [635356812946302037]
X:416,Y=143,Wheel:0 [635356812946462037]
X:449,Y=157,Wheel:0 [635356812946702037]
X:498,Y=185,Wheel:0 [635356812946862037]
X:583,Y=210,Wheel:0 [635356812947102037]
X:647,Y=222,Wheel:0 [635356812947252037]
X:667,Y=233,Wheel:0 [635356812947422037]
X:673,Y=234,Wheel:0 [635356812947652037]
X:673,Y=236,Wheel:0 [635356812947822037]
X:674,Y=237,Wheel:0 [635356812948062037]
X:674,Y=238,Wheel:0 [635356812948222037]
X:675,Y=240,Wheel:0 [635356812948452037]
X:673,Y=251,Wheel:0 [635356812952132037]
X:649,Y=280,Wheel:0 [635356812952292037]
X:602,Y=317,Wheel:0 [635356812952532037]
X:555,Y=358,Wheel:0 [635356812952692037]
X:535,Y=371,Wheel:0 [635356812952932037]
X:526,Y=380,Wheel:0 [635356812953092037]
X:523,Y=381,Wheel:0 [635356812953332037]
X:523,Y=383,Wheel:0 [635356812953732037]
X:523,Y=391,Wheel:0 [635356812953982037]
X:519,Y=406,Wheel:0 [635356812954142037]
X:518,Y=414,Wheel:0 [635356812954382037]
X:516,Y=418,Wheel:0 [635356812954542037]
X:516,Y=422,Wheel:0 [635356812954772037]
X:516,Y=426,Wheel:0 [635356812954932037]
X:516,Y=427,Wheel:0 [635356812955172037]
X:516,Y=430,Wheel:0 [635356812955332037]
X:515,Y=433,Wheel:0 [635356812955572037]
X:512,Y=434,Wheel:0 [635356812955732037]
X:512,Y=435,Wheel:0 [635356812956302037]
X:512,Y=436,Wheel:0 [635356812956702037]
X:511,Y=439,Wheel:0 [635356812957662037]
X:510,Y=440,Wheel:0 [635356812957822037]
X:506,Y=442,Wheel:0 [635356812958062037]
X:499,Y=442,Wheel:0 [635356812958212037]
X:497,Y=443,Wheel:0 [635356812958452037]
X:496,Y=443,Wheel:0 [635356812958612037]
X:494,Y=443,Wheel:0 [635356812959182037]
X:492,Y=442,Wheel:0 [635356812959342037]
X:490,Y=442,Wheel:0 [635356812959582037]
X:490,Y=441,Wheel:0 [635356812959742037]
X:490,Y=440,Wheel:0 [635356812960542037]
X:490,Y=440,Wheel:0 [635356812962612037]
```

```
X:490,Y=440,Wheel:0 [635356812963892037]
X:491,Y=438,Wheel:0 [635356813007732037]
X:499,Y=436,Wheel:0 [635356813007972037]
X:519,Y=434,Wheel:0 [635356813008132037]
X:538,Y=432,Wheel:0 [635356813008372037]
X:560,Y=432,Wheel:0 [635356813008532037]
X:585,Y=430,Wheel:0 [635356813008772037]
X:597,Y=430,Wheel:0 [635356813008932037]
X:606,Y=427,Wheel:0 [635356813009092037]
X:608,Y=427,Wheel:0 [635356813009332037]
X:612,Y=427,Wheel:0 [635356813009492037]
X:620,Y=432,Wheel:0 [635356813009732037]
X:635,Y=436,Wheel:0 [635356813009892037]
X:651,Y=440,Wheel:0 [635356813010132037]
X:661,Y=440,Wheel:0 [635356813010292037]
X:662,Y=440,Wheel:0 [635356813010532037]
X:665,Y=441,Wheel:0 [635356813011252037]
X:673,Y=442,Wheel:0 [635356813011422037]
X:683,Y=444,Wheel:0 [635356813011652037]
X:689,Y=444,Wheel:0 [635356813011812037]
X:696,Y=446,Wheel:0 [635356813012052037]
X:702,Y=446,Wheel:0 [635356813012212037]
X:704,Y=447,Wheel:0 [635356813012452037]
X:705,Y=447,Wheel:0 [635356813012692037]
X:710,Y=448,Wheel:0 [635356813012942037]
X:720,Y=448,Wheel:0 [635356813013092037]
X:726,Y=448,Wheel:0 [635356813013332037]
X:731,Y=448,Wheel:0 [635356813013492037]
X:734,Y=448,Wheel:0 [635356813013732037]
X:735,Y=448,Wheel:0 [635356813014132037]
X:736,Y=447,Wheel:0 [635356813014542037]
X:740,Y=443,Wheel:0 [635356813014862037]
X:748,Y=440,Wheel:0 [635356813015022037]
X:750,Y=438,Wheel:0 [635356813015252037]
X:751,Y=437,Wheel:0 [635356813015652037]
X:758,Y=439,Wheel:0 [635356813033892037]
X:785,Y=453,Wheel:0 [635356813034052037]
X:850,Y=472,Wheel:0 [635356813034292037]
X:945,Y=501,Wheel:0 [635356813034452037]
X:1024,Y=544,Wheel:0 [635356813034692037]
X:1066,Y=564,Wheel:0 [635356813034852037]
X:1087,Y=579,Wheel:0 [635356813035092037]
X:1101,Y=595,Wheel:0 [635356813035252037]
X:1111,Y=613,Wheel:0 [635356813035492037]
X:1122,Y=620,Wheel:0 [635356813035652037]
X:1133,Y=626,Wheel:0 [635356813035892037]
X:1147,Y=635,Wheel:0 [635356813036052037]
```

```
X:1159,Y=654,Wheel:0 [635356813036292037]
X:1169,Y=669,Wheel:0 [635356813036452037]
X:1178,Y=681,Wheel:0 [635356813036692037]
X:1180,Y=689,Wheel:0 [635356813036852037]
X:1181,Y=697,Wheel:0 [635356813037012037]
X:1185,Y=705,Wheel:0 [635356813037252037]
X:1187,Y=718,Wheel:0 [635356813037412037]
...
```

列表 C.3: An example for a touch panel event log file

```
[ 32264.245626] /dev/input/event0: EV_ABS ABS_MT_TRACKING_ID 00003ad4
[ 32264.245656] /dev/input/event0: EV_ABS ABS_MT_TOUCH_MAJOR 0000000d
[ 32264.245656] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000027
[ 32264.245656] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000325
[ 32264.245687] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 000006bb
[ 32264.245687] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32264.302602] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000053
[ 32264.302633] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 0000031b
[ 32264.302633] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32264.309316] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000055
[ 32264.309316] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 0000031a
[ 32264.309347] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32264.315999] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000319
[ 32264.316030] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32264.322683] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000053
[ 32264.322683] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000318
[ 32264.322713] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32264.338308] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000050
[ 32264.338308] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000317
[ 32264.338308] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32264.344594] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 0000004c
[ 32264.344625] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000316
[ 32264.344625] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 000006bc
[ 32264.344625] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32264.350942] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000043
[ 32264.350973] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000315
[ 32264.350973] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32264.356710] /dev/input/event0: EV_ABS ABS_MT_TOUCH_MAJOR 0000000c
[ 32264.356740] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000035
[ 32264.356740] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000314
[ 32264.356740] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32264.362508] /dev/input/event0: EV_ABS ABS_MT_TOUCH_MAJOR 0000000b
[ 32264.362539] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000029
[ 32264.362539] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000313
[ 32264.362539] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 000006bb
[ 32264.362539] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32264.368276] /dev/input/event0: EV_ABS ABS_MT_TOUCH_MAJOR 0000000a
[ 32264.368307] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 0000001f
[ 32264.368307] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000314
[ 32264.368307] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32264.379842] /dev/input/event0: EV_ABS ABS_MT_TRACKING_ID ffffffff
[ 32264.379873] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32265.212575] /dev/input/event0: EV_ABS ABS_MT_TRACKING_ID 00003ad5
[ 32265.212606] /dev/input/event0: EV_ABS ABS_MT_TOUCH_MAJOR 0000000d
[ 32265.212606] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000018
[ 32265.212606] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 0000040f
[ 32265.212606] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 0000073f
[ 32265.212636] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32265.325490] /dev/input/event0: EV_ABS ABS_MT_TRACKING_ID ffffffff
[ 32265.325521] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32266.055959] /dev/input/event0: EV_ABS ABS_MT_TRACKING_ID 00003ad6
[ 32266.056020] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000036
[ 32266.056020] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 000000a9
[ 32266.056020] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 000005ac
[ 32266.056020] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32266.168722] /dev/input/event0: EV_ABS ABS_MT_TRACKING_ID ffffffff
[ 32266.168722] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32266.958272] /dev/input/event0: EV_ABS ABS_MT_TRACKING_ID 00003ad7
[ 32266.958303] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 0000002f
[ 32266.958333] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000250
[ 32266.958333] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 00000726
[ 32266.958333] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32267.062154] /dev/input/event0: EV_ABS ABS_MT_TOUCH_MAJOR 0000000c
[ 32267.062185] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000031
[ 32267.062215] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000246
[ 32267.062215] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 00000720
[ 32267.062215] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32267.067922] /dev/input/event0: EV_ABS ABS_MT_TOUCH_MAJOR 0000000b
[ 32267.067922] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000026
[ 32267.067922] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000245
[ 32267.067953] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 0000071f
[ 32267.067953] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32267.073690] /dev/input/event0: EV_ABS ABS_MT_TOUCH_MAJOR 0000000a
[ 32267.073690] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 0000001d
[ 32267.073690] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000244
[ 32267.073690] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 0000071e
[ 32267.073720] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32267.079427] /dev/input/event0: EV_ABS ABS_MT_TOUCH_MAJOR 00000009
[ 32267.079458] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000016
[ 32267.079458] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000243
[ 32267.079458] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 0000071c
[ 32267.079458] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32267.090963] /dev/input/event0: EV_ABS ABS_MT_TRACKING_ID ffffffff
[ 32267.090993] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32267.982748] /dev/input/event0: EV_ABS ABS_MT_TRACKING_ID 00003ad8
[ 32267.982809] /dev/input/event0: EV_ABS ABS_MT_TOUCH_MAJOR 0000000d
[ 32267.982809] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 0000002b
[ 32267.982809] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000095
[ 32267.982809] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 000005ba
[ 32267.982839] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32268.094106] /dev/input/event0: EV_ABS ABS_MT_TRACKING_ID ffffffff
[ 32268.094137] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32270.875143] /dev/input/event0: EV_ABS ABS_MT_TRACKING_ID 00003ad9
[ 32270.875173] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000024
[ 32270.875173] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 0000013d
[ 32270.875173] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 000006aa
[ 32270.875204] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32270.940481] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000074
[ 32270.940511] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000133
[ 32270.940511] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32270.947164] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000072
[ 32270.947164] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000132
[ 32270.947164] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32270.954061] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 0000006f
[ 32270.954092] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 000006ab
[ 32270.954092] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32270.969808] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 0000006b
[ 32270.969839] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 000006ac
[ 32270.969839] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32270.975790] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000063
[ 32270.975820] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000133
[ 32270.975820] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 000006ad
[ 32270.975820] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32270.981863] /dev/input/event0: EV_ABS ABS_MT_TOUCH_MAJOR 0000000c
[ 32270.981893] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 0000004c
[ 32270.981893] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000136
[ 32270.981893] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 000006ae
[ 32270.981924] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32270.987661] /dev/input/event0: EV_ABS ABS_MT_TOUCH_MAJOR 0000000b
[ 32270.987661] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 0000003a
[ 32270.987661] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 0000013a
[ 32270.987691] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 000006ab
[ 32270.987691] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32270.999196] /dev/input/event0: EV_ABS ABS_MT_TRACKING_ID ffffffff
[ 32270.999227] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32271.449026] /dev/input/event0: EV_ABS ABS_MT_TRACKING_ID 00003ada
[ 32271.449056] /dev/input/event0: EV_ABS ABS_MT_TOUCH_MAJOR 0000000d
[ 32271.449056] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000028
[ 32271.449056] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000306
[ 32271.449087] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 00000728
[ 32271.449087] /dev/input/event0: EV_SYN SYN_REPORT 00000000
[ 32271.551504] /dev/input/event0: EV_ABS ABS_MT_TOUCH_MAJOR 0000000a
[ 32271.551534] /dev/input/event0: EV_ABS ABS_MT_PRESSURE 00000025
[ 32271.551534] /dev/input/event0: EV_ABS ABS_MT_POSITION_X 00000313
[ 32271.551534] /dev/input/event0: EV_ABS ABS_MT_POSITION_Y 00000725
[ 32271.551565] /dev/input/event0: EV_SYN SYN_REPORT 00000000
...
```