

2019

On Enhancing Security of Password-Based Authentication

Yue Li

William & Mary - Arts & Sciences, liyueam10@gmail.com

Follow this and additional works at: <https://scholarworks.wm.edu/etd>



Part of the [Computer Sciences Commons](#)

Recommended Citation

Li, Yue, "On Enhancing Security of Password-Based Authentication" (2019). *Dissertations, Theses, and Masters Projects*. William & Mary. Paper 1563898928.

<http://dx.doi.org/10.21220/s2-j1wq-4306>

This Dissertation is brought to you for free and open access by the Theses, Dissertations, & Master Projects at W&M ScholarWorks. It has been accepted for inclusion in Dissertations, Theses, and Masters Projects by an authorized administrator of W&M ScholarWorks. For more information, please contact scholarworks@wm.edu.

On Enhancing Security of Password-based Authentication

Yue Li

Chengdu, Sichuan, China

Bachelor of Engineering, Chinese University of Hong Kong, 2013

A Dissertation presented to the Graduate Faculty
of The College of William & Mary in Candidacy for the Degree of
Doctor of Philosophy


Department of Computer Science

College of William & Mary
May, 2019

APPROVAL PAGE


This Dissertation is submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy




Yue Li

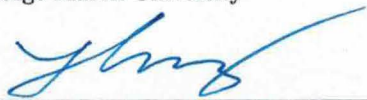
Reviewed by the Committee, May 2019




Committee Chair
Professor Haining Wang, Electrical and Computer Engineering
University of Delaware



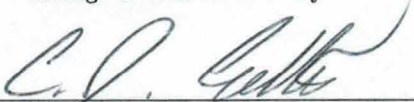
Associate Professor Kun Sun, Information Sciences and Technology
George Mason University



Associate Professor Gang Zhou, Computer Science
College of William & Mary



Assistant Professor Xu Liu, Computer Science
College of William & Mary



Professor Chase Cotton, Electrical and Computer Engineering
University of Delaware

COMPLIANCE PAGE

Research approved by

Raymond McCoy

Protocol number(s): PHSC-2015-09-14-10591-ksun

Date(s) of approval: 10/15/2015

ABSTRACT

Password has been the dominant authentication scheme for more than 30 years, and it will not be easily replaced in the foreseeable future. However, password authentication has long been plagued by the dilemma between security and usability, mainly due to human memory limitations. For example, a user often chooses an easy-to-guess (weak) password since it is easier to remember. The ever increasing number of online accounts per user even exacerbates this problem. In this dissertation, we present four research projects that focus on the security of password authentication and its ecosystem.

First, we observe that personal information plays a very important role when a user creates a password. Enlightened by this, we conduct a study on how users create their passwords using their personal information based on a leaked password dataset. We create a new metric—Coverage—to quantify the personal information in passwords. Armed with this knowledge, we develop a novel password cracker named Personal-PCFG (Probabilistic Context-Free Grammars) that leverages personal information for targeted password guessing. Experiments show that Personal-PCFG is much more efficient than the original PCFG in cracking passwords.

The second project aims to ease the password management hassle for a user. Password managers are introduced so that users need only one password (master password) to access all their other passwords. However, the password manager induces a single point of failure and is potentially vulnerable to data breach. To address these issues, we propose BluePass, a decentralized password manager that features a dual-possession security that involves a master password and a mobile device. In addition, BluePass enables a hand-free user experience by retrieving passwords from the mobile device through Bluetooth communications.

In the third project, we investigate an overlooked aspect in the password lifecycle, the password recovery procedure. We study the password recovery protocols in the Alexa top 500 websites, and report interesting findings on the de facto implementation. We observe that the backup email is the primary way for password recovery, and the email becomes a single point of failure. We assess the likelihood of an account recovery attack, analyze the security policy of major email providers, and propose a security enhancement protocol to help securing password recovery emails by two factor authentication.

Finally, we focus on a more fundamental level, user identity. Password-based authentication is just a one-time checking to ensure that a user is legitimate. However, a user's identity could be hijacked at any step. For example, an attacker can leverage a zero-day vulnerability to take over the root privilege. Thus, tracking the user behavior is essential to examine the identity legitimacy. We develop a user tracking system based on OS-level logs inside an enterprise network, and apply a variety of techniques to generate a concise and salient user profile for identity examination.

TABLE OF CONTENTS

Acknowledgments	vii
Dedications	viii
List of Tables	ix
List of Figures	xi
Chapter 1. Introduction	1
1.1 Personal Information in Passwords and Its Security Implications	3
1.2 BluePass: A Secure Hand-free Password Manager	3
1.3 Email as a Master Key: Analyzing Account Recovery in the Wild	4
1.4 UTrack: Enterprise User Tracking Based on OS-Level Audit Logs	4
1.5 Dissertation Organization	5
Chapter 2. Personal Information in Passwords and Its Security Implications	6
2.1 Introduction	6
2.2 Personal Information in Passwords	8
2.2.1 12306 Dataset	8
2.2.1.1 Introduction to Dataset	9
2.2.1.2 Basic Analysis	9
2.2.2 Personal Information	11
2.2.2.1 New Password Representation	12
2.2.2.2 Matching Method	12
2.2.2.3 Matching Results	14

2.2.2.4	Gender Password Preference	15
2.2.3	Domain Information	17
2.3	Personal Information in English-based Datasets	18
2.3.1	Methodology	18
2.3.2	Results	19
2.4	Correlation Quantification	20
2.4.1	Coverage	20
2.4.1.1	Computation Method	20
2.4.2	Coverage Results on 12306	24
2.4.3	Coverage Usage	25
2.5	Personal-PCFG	26
2.5.1	Attack Scenarios	26
2.5.2	A Revisit of PCFG	27
2.5.3	Personal-PCFG	27
2.5.3.1	Personal Information Matching	28
2.5.3.2	Password Pre-processing	28
2.5.3.3	Guess Generation	28
2.5.3.4	Adaptive Substitution	29
2.5.4	Cracking Results	29
2.6	Password Protection	32
2.7	Discussion	34
2.7.1	Limitation	34
2.7.2	Ethical Considerations	34
2.8	Conclusion	35
Chapter 3.	BluePass: A Secure Hand-free Password Manager	36
3.1	Introduction	36
3.2	System Overview and Threat Model	38

3.2.1	System Overview	39
3.2.2	Threat Model	41
3.3	System Architecture	41
3.3.1	Core Functions	41
3.3.2	Account Management	44
3.3.3	Recovery	45
3.4	Security Analysis	46
3.4.1	Two-Factor Security	46
3.4.2	Data Breach and Brute-force Attacks	46
3.4.3	Broken HTTPS or Bluetooth	47
3.5	Implementation	48
3.5.1	BluePass Server	48
3.5.2	BluePass Client-side Application	48
3.5.3	BluePass Mobile Application	49
3.6	Evaluation	50
3.6.1	Comparative Evaluation Framework	50
3.6.2	Password Auto-fill Latency	51
3.6.3	Power Consumption	53
3.7	User Study	54
3.8	Discussion	56
3.8.1	RSA Key Pair	56
3.8.2	BluePass Limitations	56
3.9	Conclusion	57
Chapter 4.	Email as a Master Key: Analyzing Account Recovery in the Wild	58
4.1	Introduction	58
4.2	Terminology and Definitions	60
4.2.1	Recovery Primitive, Method, and Protocol	60

4.2.2	Website Classification	62
4.3	Account Recovery in the Wild	63
4.3.1	Demographics	64
4.3.2	Primitive and Method Usage	65
4.4	Attack Assessment	68
4.4.1	Threat Model	68
4.4.2	Possibility to Break-in	69
4.4.2.1	Lack of Credentials	69
4.4.2.2	Classification-based Authentication	70
4.5	Damage Estimation and Email Security	71
4.5.1	Damage	71
4.5.2	Assessing Email Security	73
4.6	Securing Email-based Account Recovery	74
4.6.1	SEAR Specification	75
4.6.2	Implementation	76
4.7	Conclusion	77
Chapter 5.	UTrack: Enterprise User Tracking Based on OS-Level Audit Logs	78
5.1	Introduction	78
5.2	Motivations and Challenges	80
5.2.1	Motivations	80
5.2.2	Challenges	81
5.2.2.1	Accurate Modeling of User Behaviors	81
5.2.2.2	Identifying Data Triggered by Users	82
5.3	System Overview	83
5.4	UTrack Event Association	85
5.4.1	Tracking In-host User Activities	86
5.4.1.1	Process Lineage	86

5.4.1.2	User Log-on Sessions	87
5.4.2	Tracking Cross-host User Activities	88
5.4.3	System Cold Start	90
5.4.4	Scope	91
5.5	Pinpointing User Activities	91
5.5.1	Interactiveness Detection	92
5.5.2	Non-interactive Process Pruning	93
5.5.3	Data Modeling	94
5.6	Implementation and Evaluation	95
5.6.1	Experiment Environment	95
5.6.2	User Tracking	96
5.6.3	User-centric Activity Tracking	97
5.6.4	Data Modeling	99
5.6.5	Graph Presentation	101
5.6.6	Use Cases	103
5.7	Conclusion	103
Chapter 6.	Related Work	104
6.1	Password Study	104
6.2	Password Strength Measurement	105
6.3	Password Cracking	105
6.4	Password Manager	106
6.5	Enhancing Password Security	106
6.6	Multi-factor Authentication	106
6.7	Password Recovery	107
6.8	User Tracking and UBA	107
6.9	Log Audit	108
6.10	User Interaction Detection	109

Chapter 7. Conclusion and Future Work	110
Bibliography	113

ACKNOWLEDGMENTS

I would like to thank the numerous people who helped me in preparing this dissertation. Without their hard work, patience, and guidance it would not have been possible.

First, I would like to thank my advisor Dr. Haining Wang and Dr. Kun Sun

Next, I would like to thank my current and past research group members including Dr. Zhenyu Wu, Dr. Zhichun Li, Dr. Kangkook Jee, Dr. Jungwan Rhee, Shengye Wan, Jianhua Sun, and Luren Wang.

Lastly, I would like to thank our Computer Science Department Chair, Professor Robert Michael Lewis, and the wonderful Computer Science administration team, Vanessa Godwin, Jacquelyn Johnson, and Dale Hayes.

I would like to dedicate this dissertation to my wife, Rachel Wei, and my parents Yan Li and Yun Du, who provided endless support and love throughout my time at William & Mary.

LIST OF TABLES

2.1	Most Frequent Passwords.	10
2.2	Resistance to guessing	10
2.3	Most Frequent Password Structures.	11
2.4	Personal Information.	12
2.5	Most Frequent Password Structures.	15
2.6	Personal Information Usage.	15
2.7	Most Frequent Structures in Different Genders.	16
2.8	Most Frequent Personal Information in Different Genders.	16
2.9	Domain Information in Passwords.	17
2.10	NameSets	18
2.11	Matching Results	18
3.1	Server Side Data	43
3.2	Mobile Device Data	43
3.3	BluePass Scheme Evaluation	50
3.4	Delay Statistics	52
4.1	Recovery Primitive Distribution	65
4.2	Recovery Methods with Multiple Primitives	67
4.3	Websites Vulnerable to Account Recovery Attacks	70
4.4	Damage Estimation	72
4.5	Examining Major Email Providers	73
4.6	Password Policies	73

5.1 Servers with the Most Network Connections	97
5.2 Classification Results	98

LIST OF FIGURES

2.1	An Example of Coverage Computing.	23
2.2	Coverage distribution - 12306.	24
2.3	PCFG vs. Personal-PCFG (Offline).	30
2.4	PCFG vs. Personal-PCFG (Online).	31
2.5	Representative Points – Online attacks.	31
2.6	Coverage distribution.	33
3.1	BluePass Authentication	40
3.2	BluePass Architecture	42
3.3	BluePass Latency	52
3.4	BluePass Power Consumption	53
3.5	Survey Results	55
4.1	Recovery Methods – Single-Primitive	66
4.2	Account Recovery Examples.	76
5.1	UTrack Overview	84
5.2	Session Root Isolation	86
5.3	Virtual Process	90
5.4	Data Modeling	95
5.5	Batches in Processes	99
5.6	Lifespan of Processes	99
5.7	Example User Profile	101

Chapter 1

Introduction

Securing the user authentication process is a fundamental requirement in building an information system. Password-based authentication remains the most dominant method ever since it was introduced more than 30 years ago. Along with its popularity, many works have been done to study its security, management, and usability. A long-plaused drawback of password is that users tend to choose weak passwords (i.e., passwords that are easily remembered but also could be easily cracked), mainly due to human limited memory. There are many previous works focusing on understanding the composition of a user password. An early work shows that many passwords are merely dictionary words [86]. Recent works also demonstrate that passwords are phonetically similar to a user's native language [87], and many passwords share common patterns [121], or semantic structures [114]. It is also revealed that date or birthdate are prevalent in passwords and 4-digit PINs [23, 115]. Many password crackers are also built to demonstrate that a large amount of user chosen passwords can be easily cracked within a short period of time [29, 56, 86, 87, 89, 121].

Realizing that passwords are not secure, alternative authentication methods have been proposed, such as biometrics-based [59] or graphics-based [37, 60]. However, none of these new techniques can surpass text passwords in every aspect of security, usability, and deployment in practice. It is believed that passwords will not be easily replaced in the near future [21]. As such, people also strive to enhance the security of password authentication. A first challenge encountered is that it is hard to measure the strength of a password, after showing that entropy is not an accurate indicator [26, 120]. Nowadays, a commonly used measure is to estimate the password strength by how easy it can be

cracked by using modern crackers [67]. Furthermore, it is difficult for passwords to achieve high security levels. Traditional wisdom suggests users to create strong passwords and not to re-use a same password across different services. However, users seldom follow these security suggestions due to usability considerations [22, 36]. Some researches aim to nudge the users to choose stronger passwords, including setting strict password policy [120], providing strength feedback [43, 67, 112] and password managers [77, 85, 97, 105, 119]. It is also common that people introduce more factors in user authentication besides passwords, such as additional knowledge [25, 94] and hardware tokens [11]. Many websites now employ two factor authentication to provide better protections. However, due to its usability problem, the adoption rate is still low [93]. It is estimated that only 6.4% of Google users have used Google’s two factor authentication. To mitigate the usability issues, some works attempt to make the second factor transparent to users [88, 91].

The study of password has moved toward a multi-dimensional future. However, it is still a one-time authentication method that serves as a gatekeeper for user access control. It is very possible that user identity has changed during other steps. For instance, in an Advanced Persistent Threat (APT) attack, an attacker may launch phishing attacks or exploit zero-day vulnerabilities to take over other users’ privilege or escalate her own privilege. These security threats cannot be resolved with passwords only. A potential way is to monitor user activities and achieve behavior-based user identity audit. User behavior Analytics (UBA) describes a range of such techniques that capture abnormal user behaviors [5, 101, 109, 111]. A foundation for UBA to be successful is to accurately track a user’s activities inside the network. Without an accurate and complete user profile, UBA will be much less effective, or even useless.

In this dissertation, we present four projects. The first three projects focus on password-based authentication, including studying personal information in passwords, proposing a secure and usable password manager, and analyzing password recovery protocols in the wild. The fourth project explores continuous user behavior monitoring by tracking users’ activities inside a network. We introduce each of the project in the following.

1.1 Personal Information in Passwords and Its Security Implications

Almost all past works related to password study try to understand the statistical pattern of passwords, and leverage this knowledge to launch statistical attacks on passwords [87, 114, 121]. Namely, an adversary keeps trying password candidates from high probability to low probability based on her constructed password model. However, passwords are usually highly personalized, and different users create different passwords in a semantic sense, even if the same password creation method is used. Personal information is believed to be frequently included in a user’s password. Thus, it is important to understand its usage in password creation. Toward this end, we analyze the use of personal information in passwords, and uncover what types of personal information are included in passwords and how they are used. We quantitatively correlate the personal information and a password by developing a novel metric—Coverage. Armed with this knowledge, we further develop a new password cracker—Personal-PCFG—to efficiently generate personalized password guesses. Our evaluation shows that Personal-PCFG is much faster than the state-of-art in password cracking. The detailed analysis and experiments are presented in Chapter 2.

1.2 BluePass: A Secure Hand-free Password Manager

One way to enhance the usability and security of passwords is to use a password manager [65, 102]. A password manager allows a user to create different strong passwords and store them in a password vault. The user only needs to remember a single master password to access all her other passwords. However, such a scheme introduces a single point of failure, as whoever can access the master password is also able to access all other passwords. Furthermore, central storage of password vaults on a cloud also makes massive data breach possible. To address these issues, we propose a decentralized password manager, BluePass, that features both dual possession security and a hand-free user experience. BluePass isolates the password vault and the decryption key, and leverages the short-range Bluetooth to automate the log-in process to ensure the probity of a mobile device, where the password vault resides. Through comprehensive evaluation, we show that BluePass has

low latency, incurs little power overhead, and has good usability. We elaborate the design and evaluation of BluePass in Chapter 3.

1.3 Email as a Master Key: Analyzing Account Recovery in the Wild

Although the security of password itself has been extensively studied, the recovery process is largely overlooked and has yet to be analyzed. The password recovery process allows a user to reset a password in case she forgets the original password. There are some previous works examining the use of security questions as a way to recover passwords [63], and conclude that security questions are very weak and should not be solely used to recover a user account [20, 99]. However, the overview of password recovery implementations still remains unknown to the public. The following questions still need to be answered: (1) Are there still many websites that rely on security questions? (2) What other schemes are employed? (3) Are they secure? (4) What happens if they are not secure? We believe that examining the password recovery implementation is very important since attackers may leverage this mechanism to launch attacks to many other accounts. As such, we examine the password recovery mechanisms in the wild, which helps the security communities to gain an overview of contemporary password recovery implementations. Our analysis confirms that most of the websites rely on a single user email account to reset a password. We find that multiple online accounts would be easily compromised if an email account was compromised, and many email providers fail to properly protect users' email accounts. Finally, we propose a security enhancement protocol based on the current emailing infrastructure in a secure and backward compatible fashion. The details are presented in Chapter 4.

1.4 UTrack: Enterprise User Tracking Based on OS-Level Audit Logs

We finally go beyond the password authentication and focus on a more fundamental level—user identity. Authentication is essentially a way to ensure user legitimacy. However, a user's identity

could be changed or hacked regarding to user accounts in a computer network. It is important to continuously check the user identity of each real user inside a network for security purposes. In order to do so, we develop a novel user tracking system based on OS-level logs. We track user activities under both in-host and cross-host scenarios, and apply a variety of techniques to solve the “needle in a haystack” problem that many log-based techniques faced. As a result, we are able to construct concise and accurate user profiles within a reasonable period of time. The system is presented in Chapter 5.

1.5 Dissertation Organization

This dissertation is organized as follows. Chapter 2 introduces a study on personal information in passwords and its application in password cracking. Chapter 3 presents the novel password manager, BluePass, that features a dual possession security and a hand-free user experience. Chapter 4 studies the contemporary password recovery protocols in the wild, and investigate the likelihood, damage, and defense of a password reset attack. Chapter 5 explores beyond the one-time password authentication, and focuses on continuous monitoring of user identity. We propose a novel enterprise-level user tracking system based on OS-level logs. Chapter 6 surveys related works, and finally, Chapter 7 concludes this dissertation and discusses future works.

Chapter 2

Personal Information in Passwords and Its Security Implications

2.1 Introduction

The text-based password is still believed to remain a dominating and irreplaceable authentication method in the foreseeable future. Although researchers have proposed different authentication mechanisms, no alternative can bring all the benefits of passwords without introducing extra burdens to users [21]. However, passwords have long been criticized as being one of the weakest links in authentication. Due to the human memorability limitation, user passwords are usually far from truly random [19, 81, 87, 115, 123]. For instance, “secret” is more likely a human-chosen password than “zjorqpe.” In other words, human users are prone to choose weak passwords simply because they are easier to remember. As a result, most passwords are chosen within a small portion of the entire password space, leaving them vulnerable to brute-force or dictionary attacks.

To increase password security, online authentication systems have started to enforce stricter password policies. Meanwhile, many websites provide password strength meters to help users create secure passwords. However, these meters are proven to be ad-hoc and inconsistent [38, 43]. To better assess the strength of passwords, one needs to have a deeper understanding of how users construct their passwords. Knowing the exact tactics to create passwords also helps an attacker to crack passwords. Meanwhile, if a user is aware of the potential vulnerability induced by a commonly

used password creation method, the user can avoid using the same method for creating passwords.

Toward this end, researchers have made significant efforts to unveil the structures of passwords. Traditional dictionary attacks on passwords have shown that users tend to use simple dictionary words to construct their passwords [56, 86]. The first language of a user is also preferred when constructing passwords [19]. Besides, passwords are commonly phonetically memorable [87] even though they are not simple dictionary words. It is also indicated that users may use keyboard strings such as "qwerty" and "qweasdzxc," trivial strings such as "password" and "123456," and date strings such as "19951225" in their passwords [76, 100, 115]. However, most studies reveal only superficial password patterns, and the semantic-rich composition of passwords remains to be fully uncovered. Fortunately, an enlightening work investigates how users generate their passwords by learning the semantic patterns [114].

This project studies password semantics from a different perspective: the use of personal information. We utilize a leaked password dataset, which contains personal information, from a Chinese website for this study. We first measure the usage of personal information in password creation and present interesting observations. We are able to obtain the most popular password structures with personal information embedded. It is also observed that males and females behave differently when using personal information in password creation. In addition, we assess the usage of names in other leaked password datasets that do not come with any personal information.

Next, we introduce a new metric called Coverage to accurately quantify the correlation between personal information and user passwords. Since it considers both the length and continuation of personal information, Coverage is a useful metric to measure the strength of a password. Our quantification results using the Coverage metric confirm our direct measurement results on the dataset, showing the efficacy of Coverage. Moreover, Coverage is easy to integrate with existing tools, such as password strength meters. To demonstrate the security vulnerability induced by using personal information in passwords, we propose a semantics-rich Probabilistic Context-Free Grammars (PCFG) method called Personal-PCFG, which extends PCFG [121] by considering personal information symbols in password structures. Personal-PCFG takes a user’s personal information as another input vector and generates highly personalized guesses for a specific account. With the

assistance of such knowledge, Personal-PCFG is able to crack passwords much faster than original PCFG. It also makes an online attack more feasible.

Finally, we present simple distortion functions to defend against these semantics-aware attacks such as Personal-PCFG or [114]. Our evaluation results demonstrate that distortion functions can effectively protect passwords by significantly reducing the unwanted correlation between personal information and passwords.

Though our study is primarily based on a dataset collected from a Chinese website, we also try to extend our measurement study to English-based websites. We conclude that as long as memorability plays an important role in password creation, the correlation between personal information and user passwords remains, regardless of the language. We believe that our work on personal information quantification, password cracking, and password protection could be applicable to any other text-based password datasets from different websites.

2.2 Personal Information in Passwords

Intuitively, people tend to create passwords based on their personal information because human beings are limited by their memory capacities. We show that personal information plays an important role in a human-chosen password by dissecting passwords in a mid-sized password dataset. Understanding the usage of personal information in passwords and its security implications can help us further enhance password security. To start, we introduce the dataset used throughout this study.

2.2.1 12306 Dataset

A number of password datasets have been exposed to the public in recent years, usually containing several thousand to millions of real passwords. As a result, there are several password-cracking studies based on analyzing these datasets [19, 76]. In this project, a dataset called 12306 is used to illustrate how personal information is involved in password creation.

2.2.1.1 Introduction to Dataset

At the end of 2014, a Chinese dataset was leaked to the public by anonymous attackers. It is reported that the dataset was collected by trying passwords from other leaked passwords in an online attack. hereafter, it is called 12306 dataset because all passwords are from a website www.12306.cn, which is the official site of the online railway ticket booking system in China. There is no data available on the exact number of users of the 12306 website; however, we infer at least tens of millions of registered users in the system, since it is the official website for the entire Chinese railway system.

The 12306 dataset contains more than 130,000 Chinese passwords. Having witnessed so many leaked large datasets, its size is considered medium. What makes it special is that together with plaintext passwords, the dataset also includes several types of personal information, such as a user’s name and the government-issued unique ID number (similar to the U.S. Social Security Number). As the website requires a real ID number to register and people must provide accurate personal information to book a ticket, the information in this dataset is considered reliable.

2.2.1.2 Basic Analysis

We first conduct a simple analysis to reveal general characteristics of the 12306 dataset. For data consistency, users whose ID number is not 18-digit long are removed. These users may have used other forms of ID (e.g., passport number) for registration and account for 0.2% of the dataset size. The dataset contains 131,389 passwords for analysis after being cleansed. Note that different websites may have different password creation policies. From the leaked dataset, we can infer that the password policy is quite simple—all passwords cannot be shorter than six characters. Besides, there is no restriction on types of characters used.

The average length of passwords in the dataset is 8.44. The most common passwords are listed in Table 2.1. The dominating passwords are trivial passwords (e.g., 123456, a123456, etc.), keyboard passwords (e.g., 1qaz2wsx, 1q2w3e4r, etc.), and phrasal passwords (e.g., “I love you”). Both “5201314” and “woaini1314” mean “I love you forever” in Chinese. The most commonly used passwords in 12306 dataset are similar to those are found in a previous study [76]; however, popular 12306 passwords distribute more sparsely in the password space. The most popular password,

Table 2.1: Most Frequent Passwords.

Rank	Password	Amount	Percentage
1	123456	389	0.296%
2	a123456	280	0.213%
3	123456a	165	0.125%
4	5201314	160	0.121%
5	111111	156	0.118%
6	woaini1314	134	0.101%
7	qq123456	98	0.074%
8	123123	97	0.073%
9	000000	96	0.073%
10	1qaz2wsx	92	0.070%

Table 2.2: Resistance to guessing

H_∞	\tilde{G}	λ_5	λ_{10}	$\tilde{G}_{0.25}$	$\tilde{G}_{0.5}$
8.4	16.85	0.25%	0.44%	16.65	16.8

“123456,” accounts for less than 0.3% of all passwords while the number being 2.17% in [76]. The password sparsity may be due to the importance of the website service nature, where users are less prone to use very trivial passwords like “123456” and there are fewer Sybil accounts as a real ID number is mandatory for registration.

Similar to [76], the resistance to guessing of the 12306 dataset is measured in terms of several useful metrics, including the worst-case security bit representation (H_∞), the guesswork bit representation (\tilde{G}), the α -guesswork bit representations ($\tilde{G}_{0.25}$ and $\tilde{G}_{0.5}$), and the β -success rates (λ_5 and λ_{10}). The results are listed in Table 2.2, showing that 12306 has a substantially higher worst-case security and β -success rates than the previously studied datasets. This is mainly because the users of 12306 avoid using extremely guessable passwords such as “123456.” It implies that users have certain password security concerns when creating passwords for critical services like 12306. However, their security concern is limited to the avoidance of using extremely guessable passwords. As indicated by the values of α -guesswork, the overall password sparsity of the 12306 dataset is not higher than that of the previously studied datasets.

We also study the composition structures of passwords in 12306. The most popular password structures are listed in Table 2.3. Similar to a previous study [76], our results again show that Chinese users prefer digits in their passwords as opposed to letters that are favored by English-

speaking users. Specifically, the top five structures all have a significant portion of digits. The reason behind this may be that Chinese characters are logogram-based, and digits seem to be the best alternative when creating a password.

Table 2.3: Most Frequent Password Structures.

Rank	Structure	Amount	Percentage
1	D_7	10,893	8.290%
2	D_8	9,442	7.186%
3	D_6	9,084	6.913%
4	L_2D_7	5,065	3.854%
5	L_3D_6	4,820	3.668%
6	L_1D_7	4,770	3.630%
7	L_2D_6	4,261	3.243%
8	L_3D_7	3,883	2.955%
9	D_9	3,590	2.732%
10	L_2D_8	3,362	2.558%

“D” represents digits and “L” represents English letters. The number indicates the segment length. For example, L_2D_7 means the password contains 2 letters followed by 7 digits.

In summary, the 12306 dataset is a Chinese password dataset that has general Chinese password characteristics. Users have certain level of security concern by choosing less trivial passwords. However, in terms of the overall sparsity, the 12306 dataset is no higher than previously studied datasets.

2.2.2 Personal Information

Other than passwords, 12306 dataset also includes multiple types of personal information, as listed in Table 2.4.

Note that the government-issued ID number is a unique 18-digit number, which intrinsically includes the owner’s personal information. Specifically, digits 1-6 represent the birthplace, digits 7-14 represent the birthdate, and digit 17 represents the gender—odd being male and even being female. We take out the 8-digit birthdate and treat it separately since birthdate is a critical piece of personal information in password creation. Thereby, six types of personal information are considered: name, birthdate, email address, cell phone number, account name, and ID number (birthdate excluded).

Table 2.4: Personal Information.

Type	Description
Name	User’s Chinese name
Email address	User’s registered email address
Cell phone	User’s registered cell phone number
Account name	The username used to log in to the system
ID number	Government-issued ID number

2.2.2.1 New Password Representation

To better illustrate how personal information correlates to user passwords, we develop a new representation of a password by adding more semantic symbols besides the conventional “D,” “L,” and “S” symbols, which stand for digit, letter, and special character, respectively.

The password is first matched to the six types of personal information under this new representation. For example, a password “alice1987abc” can be represented as $[Name][Birthdate]L_3$, instead of $L_3D_4L_3$ as in the traditional representation. The matched personal information is denoted by corresponding tags— $[Name]$ and $[Birthdate]$ in this example; for segments that are not matched, we still use “D,” “L,” and “S” to describe the symbol types.

Representations like $[Name][Birthdate]L_3$ are more accurate than $L_5D_4L_3$ in describing the composition of a user password by including more detailed semantic information. Using this representation, the following matching method is applied to the entire 12306 dataset to uncover the way personal information appears in password structures.

2.2.2.2 Matching Method

We propose a matching method to locate personal information in a user password, which is shown in Algorithm 1. The high-level idea is that we first generate all substrings of the password and sort them in descending-length order. Then we match these substrings, from the longest to the shortest, to all types of personal information. If a match is found, the match function is recursively applied over the remaining password segments until no further matches are found. The segments that are not matched to any personal information will still be labeled using the traditional “LDS” tags.

In Algorithm 1, we first ensure that the length of a password segment is at least 2. Then we

Algorithm 1 Personal Information Matching.

```
1: procedure MATCH(pwd,infolist)
2:   newform  $\leftarrow$  empty_string
3:   if len(pwd) == 0 then
4:     return empty_string
5:   end if
6:   substring  $\leftarrow$  get_all_substring(pwd)
7:   reverse_length_sort(substring)
8:   for eachstring  $\in$  substring do
9:     if len(eachstring)  $\geq$  2 then
10:      if matchbd(eachstring,infolist) then
11:        tag  $\leftarrow$  "[BD]"
12:        leftover  $\leftarrow$  pwd.split(eachstring)
13:        break
14:      end if
15:      ...
16:      if matchID(eachstring,infolist) then
17:        if tag != None then
18:          tag  $\leftarrow$  tag + "&[ID]"
19:        else
20:          tag  $\leftarrow$  "[ID]"
21:        end if
22:        leftover  $\leftarrow$  pwd.split(eachstring)
23:        break
24:      end if
25:    else
26:      break
27:    end if
28:  end for
29:  if leftover.size()  $\geq$  2 then
30:    for i  $\leftarrow$  0 to leftover.size()-2 do
31:      newform  $\leftarrow$  MATCH(leftover[i],infolist) + tag
32:    end for
33:    newform  $\leftarrow$  MATCH(leftover[leftover.size() - 1])+newform
34:  else
35:    newform  $\leftarrow$  seg(pwd)
36:  end if
37:  results  $\leftarrow$  extract_ambiguous_structures(newform)
38:  return results
39: end procedure
```

try to match eligible segments to each kind of the personal information (line 10 and line 16). Note that personal information may not always appear as it is. Instead people sometimes may mangle them a bit or use abbreviations. As each case is different, we do not present the specific algorithms

used for each type of the personal information. Instead, we describe the methods as follows. For the Chinese names, we convert them into Pinyin form, which is the alphabetic representation of Chinese characters. Then we compare password segments to 10 possible permutations of a name, such as *lastname+firstname* and *last_initial+firstname*. If the segment is exactly the same as one of the permutations, we consider it a match. For birthdate, we list 17 possible permutations, such as YYYYMMDD, and compare them with a password segment. If the segment is the same as any permutation, we consider it a match. For account name, email address, cell phone number, and ID number, we further constrain the length of a segment to be at least 3 to avoid mismatching by coincidence.

Note that for one password segment, it may result in matches of multiple types of personal information. In such cases, all matches are counted. Thus, the results of Algorithm 1 contain all possible matches.

2.2.2.3 Matching Results

After applying Algorithm 1 to 12306 dataset, we find that 78,975 out of 131,389 (60.1%) of the passwords contain at least one of the six types of personal information. Apparently, personal information is frequently used in password creation. The ratio could be even higher if we know more personal information about users. We present the top 10 password structures in Table 2.5 and the usage of personal information in passwords in Table 2.6. As mentioned above, a password segment may match multiple types of personal information, and we count all of these matches. Therefore, the sum of the percentages is greater than 60.1%. Within 131,389 passwords, we obtain 1,895 password structures. Based on Tables 2.5 and 2.6, we can see that people largely rely on personal information when creating passwords. Among the 6 types of personal information, birthdate, account name, and name are most popular with a more than 20% occurrence rate, and 12.66% of users include email in their passwords. However, only small percentage of people include their cellphone and ID number in their passwords (less than 3%).

Table 2.5: Most Frequent Password Structures.

Rank	Structure	Amount	Percentage
1	[ACCT]	6,820	5.190%
2	D7	6,224	4.737%
3	[NAME][BD]	5,410	4.117%
4	[BD]	4,470	3.402%
5	D6	4,326	3.292%
6	[EMAIL]	3,807	2.897%
7	D8	3,745	2.850%
8	L1D7	2,829	2.153%
9	[NAME]D7	2,504	1.905%
10	[ACCT][BD]	2,191	1.667%

Table 2.6: Personal Information Usage.

Rank	Information Type	Amount	Percentage
1	Birthdate	31,674	24.10%
2	Account Name	31,017	23.60%
3	Name	29,377	22.35%
4	Email	16,642	12.66%
5	ID Number	3,937	2.996%
6	Cell Phone	3,582	2.726%

2.2.2.4 Gender Password Preference

As the user ID number in our dataset actually contains gender information (i.e., the second-to-last digit in the ID number representing gender), we compare the password structures between males and females to see if there is any difference in password preference.

The average password lengths for males and females are 8.41 and 8.51 characters, respectively, which suggests that gender does not greatly affect the length of passwords. Applying the matching method to each gender, we then observe that 61.0% of male passwords contain personal information while only 54.1% of female passwords contain personal information. The top 10 structures for each gender are listed in Table 2.7, and personal information usage is shown in Table 2.8. These results demonstrate that male users are more likely to include personal information in their passwords than female users.

Additionally, we have two other interesting observations. First, the total number of password structures for females is 1,756, which is 10.3% more than that of males. Besides, 28.38% of males' passwords fall into the top 10 structures while only 23.94% of females' passwords fall into the

top 10 structures. Thus, passwords created by males seem denser and possibly more predictable. Second, males and females vary significantly in the use of name information. While 23.32% of males' passwords contain their names, only 12.94% of females' passwords contain their names. We conclude that the use of name is the main difference in personal information usage between males and females.

Table 2.7: Most Frequent Structures in Different Genders.

Rank	Male		Female	
	Structure	Percentage	Structure	Percentage
1	[ACCT]	4.647%	D6	3.909%
2	D7	4.325%	[ACCT]	3.729%
3	[NAME][BD]	3.594%	D7	3.172%
4	[BD]	3.080%	D8	2.453%
5	D6	2.645%	[EMAIL]	2.372%
6	[EMAIL]	2.541%	[NAME][BD]	2.309%
7	D8	2.158%	[BD]	1.968%
8	L1D7	2.088%	L2D6	1.518%
9	[NAME]D7	1.749%	L1D7	1.267%
10	[ACCT][BD]	1.557%	L2D7	1.240%
NA	TOTAL	28.384%	TOTAL	23.937%

Table 2.8: Most Frequent Personal Information in Different Genders.

Rank	Male		Female	
	Information Type	Percentage	Information Type	Percentage
1	[BD]	24.56%	[ACCT]	22.59%
2	[ACCT]	23.70%	[BD]	20.56%
3	[NAME]	23.31%	[NAME]	12.94%
4	[EMAIL]	12.10%	[EMAIL]	13.62%
5	[ID]	2.698%	[CELL]	2.982%
6	[CELL]	2.506%	[ID]	2.739%

In summary, passwords of males are generally composed of more personal information, especially the name of a user. In addition, the password diversity for males is lower. Our analysis indicates that the passwords of males are more vulnerable to cracking than those of females. At least from the perspective of personal-information-related attacks, our observations are different from the conclusion drawn in [84] that males have slightly stronger passwords than females.

2.2.3 Domain Information

Cao et al. [27] proposed using domain information to crack user passwords. It draws our attention because we have shown the involvement of personal information in a user’s password, so naturally we are also interested in the involvement of the domain information as another aspect of semantic information in password creation. By domain information, we mean the information of an Internet domain (e.g., a web service). For example, the famous “Rockyou” dataset is leaked from *www.rockyou.com*, and the domain information here is “rockyou.” In our personal information study, the domain information is “12306.” It is reasonable for users to include domain information in their passwords to keep their passwords different from site to site but still easy to remember. This approach may be promising to balance password security and memorability; however, the idea has not been validated with a large-scale experiment. Therefore, we attempt to verify whether domain information is involved in password creation. In addition to the medium-sized 12306 dataset, we study more password datasets, including Tianya, Rockyou, PHPBB, and MySpace datasets. In each dataset, we search the domain information and its meaningful substrings in the passwords. The results are shown in Table 2.9.

Table 2.9: Domain Information in Passwords.

Dataset	Password Amount	Domain Info Amount	Percentage
Rockyou	14,344,391	44,025	0.3%
Tianya	26,832,592	29,430 ¹	0.11%
PHPBB	184,389	2,209	1.2%
12306	131,389	490	0.4%
MySpace	37,144	72	0.2%

From Table 2.9, we can see that some users indeed include domain information in their passwords. Our results indicate that all the datasets examined have 0.11% to 1.2% of passwords that contain domain information. However, the small percentage indicates that while the inclusion of domain information in a password helps users to create different passwords for different websites, not many users are currently using such a method.

¹We find that 45,574 passwords in the Tianya dataset is “111222tiany.” It does not make much sense that so many users would have the same password, and it is highly likely that they are sybil accounts. Thus, these duplications are removed from our analysis.

Table 2.10: NameSets

Set Name	Total Number	Unique Number	Common	LongCommon4	LongCommon5
Firstname	138,797,749	4,347,667	1,652	1,519	1,232
Lastname	138,797,749	5,369,437	1,497	1,390	1,179

“Common” means common names, where the name has more than 10,000 occurrences in the dataset, “LongCommon4” means the common names with length no less than 4, and “LongCommon5” means the common names with length no less than 5. We will use these filtered data in our experiments.

Table 2.11: Matching Results

Set Name	Total Number	Match4	Exact Match4	Match5	Exact Match5
Rockyou	14,344,391	3,540,629 (24.7%)	6,919 (0.05%)	1,750,702 (12.2%)	7,153 (0.05%)
PHPBB	184,389	32,180 (17.5%)	1,709 (0.9%)	14,418 (7.8%)	1,661 (0.9%)
MySpace	37,144	11,521 (31.0%)	221 (0.6%)	5,965 (16.1%)	206 (0.5%)

“Match4” means the name is at least 4-characters long. “Exact Match” indicates the password is exactly the same as the password.

2.3 Personal Information in English-based Datasets

So far, we have only discussed personal information in 12306 dataset. However, due to cultural or language differences, analyzing a single dataset may result in bias conclusions. To gain a better understanding of how personal information generally resides in passwords, we try to extend the analysis to English-based password datasets. Unfortunately, up to this point, there is no available English-based password dataset that incorporates personal information. Due to the lack of personal information, English-based datasets cannot provide us with an accurate correlation as we had in the 12306 dataset. However, we can still make use of some easily inferable personal information from the password itself for a pilot study. One type of inferable information is the user’s name. Specifically, we examine the name usage in user passwords by matching commonly used names to the passwords. A name used in the password is a strong indicator that the user has included personal information in the password. Though the name included in the password may have nothing to do with the account owner, the probability of such cases should be fairly low.

2.3.1 Methodology

In this study, we use the three English-based leaked password datasets (Rockyou, Phpbb, and Myspace) as in Section 2.2.3 and two name datasets (firstname set and lastname set) collected from Facebook. While the three password datasets have been used extensively in many works, the two

name datasets are not as prevalent in the research. We show the basic statistics on these two sets in Table 2.10.

For each of the leaked password datasets, we match the password to a first name or last name from the name datasets. Specifically, we try to find exact occurrences of the name in the passwords. For instance, "mary" can be matched to a password "maryspassword." However, this method inevitably results in wrong matches since some 2-grams or 3-grams are widely shared in English words, and people are known to use words from their first language in their passwords. Thus, it may not be clear whether the user is using a name or an English word that coincidentally contains the name. To mitigate this problem, we deliberately ignore names that are less than 4 or 5 characters long in 2 separate experiments. Furthermore, we only consider common names with more than 10,000 occurrences in the datasets to mitigate the effect of too many wrong matches from less commonly used names. We show the filtered results in Table 2.11. A match is considered found when a first name or a last name is in the password.

Note that there could be both under-matching and over-matching in this study. On the under-matching side, short names such as "Mary" and "Dave" are ignored under "LongCommon5" criteria. Furthermore, some users may use their name initials in passwords, which makes accurate matching much harder. Our study does not consider such cases. On the over-matching side, the problem of commonly shared n-grams still persists. With stricter filtering (ignoring short names), there should be much fewer wrong matches, which in turn results in under-matching.

2.3.2 Results

From Table 2.11, we can see that websites in English-speaking cultures still have a significant use of names in their passwords. For instance, the largest dataset "Rockyou" has more than 24.7% of passwords containing a name of length at least 4, and 12.2% of passwords containing a name of at least length 5. Furthermore, the exact match is not negligible. Exact matches are a strong indication of the use of personal information. In the 12306 dataset, the exact match has lower than a 1% probability, which is basically consistent with this study. Similarly, PHPBB and MySpace also indicate a large name use in their password sets. Our findings are consistent with the expectation

that people from different cultures and language backgrounds tend to include their personal information in their passwords. Furthermore, the extent of name use in their passwords does not largely differ—while 12306 has roughly 23% name use in a passwords, the three English-based datasets have 17.5% to 31.0% name use when only names of at least length 4 are considered.

Although our experiments shed light on general personal name use in English-based passwords, we cannot use these datasets in our study on personal information correlation and password cracking since all personal information is inferred from passwords.

2.4 Correlation Quantification

While the analysis above show the correlation between each type of personal information and passwords, they cannot accurately measure the degree of personal information involvement in an individual password. Thus, we introduce a novel metric—Coverage—to quantify the involvement of personal information in the creation of an individual password in an accurate and systematic fashion.

2.4.1 Coverage

The value of Coverage ranges from 0 to 1. A larger Coverage implies a stronger correlation. Coverage “0” means no personal information is included in a password, and Coverage “1” means the entire password is perfectly matched with one type of personal information. Though Coverage is mainly used for measuring an individual password, the average Coverage also reflects the degree of correlation in a set of passwords. In the following, we describe the algorithm to compute Coverage, presents a detailed example to illustrate how Coverage works, and elaborates the key features of Coverage.

2.4.1.1 Computation Method

To compute Coverage, we take the password and personal information in terms of strings as input and adopt a sliding window approach. To conduct the computation, a dynamic-sized window sliding from the beginning to the end of the password is maintained. The initial size of the window is 2. If

the segment behind the window matches to a certain type of personal information, the window size grows by 1. Then we try again to match the new segment to the personal information. If a match is found, the window size is further enlarged until a mismatch happens. At this point, the window resets to the initial size and continues sliding from where the mismatch happens.

Meanwhile, an array called *tag array* with the same length as the password is used to record the length of each matched password segment. For example, assuming a password with a length of 8, and its tag array is [4,4,4,4,0,0,2,2]. The first four elements in the array (i.e., {4,4,4,4}), indicate that the first 4 password symbols match a certain type of personal information. The following two elements in the array ({0,0}) indicate that the 5th and 6th symbols have no match. The last two elements in the array ({2,2}) imply that the 7th and 8th symbols again match a certain type of personal information. The personal information types matched with different password segments may or may not be the same. After eventually sliding window through the entire password string, the tag array is used to compute the value of Coverage—the sum of squares of the matched password segment length divided by the square of the password length. Mathematically, we have

$$CVG = \sum_{i=1}^n \left(\frac{l_i^2}{L^2} \right), \quad (2.1)$$

where n denotes the number of matched password segments, l_i denotes the length of the corresponding matched password segment, and L is the length of the password. We show the algorithm of computing Coverage in Algorithm 2. A match is found if at least a 2-symbol long password segment matches to a substring of certain personal information.

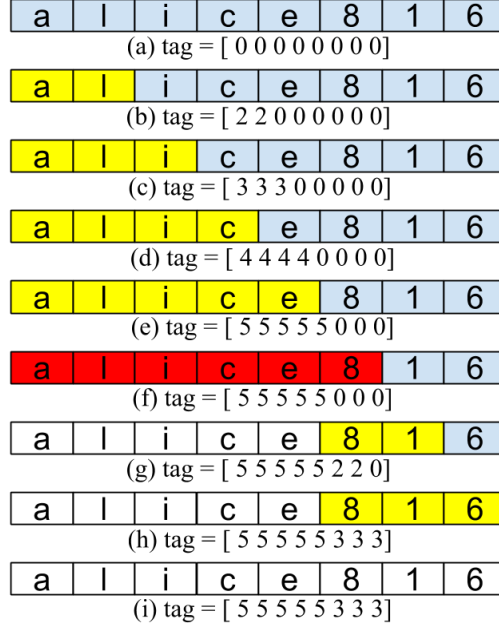
To better illustrate how Coverage is computed, we show a simple example in Figure 2.1. Here we assume a user named Alice, who was born on August 16, 1988. One password of Alice happens to be “alice816.” If applying the matching algorithm in Section 2.2.2.2, the structure of this password will be [NAME][BD]. Apparently, her password is highly related to her personal information. To quantify this correlation, we follow the Algorithm 2 to compute Coverage for her. The computation steps are shown in Figure 2.1, and each step is detailed as follows. In step (a), the tag array is initialized as [0,0,0,0,0,0,0,0]. Note the personal information includes “alice” as the name and “19880816” as the birthdate. In step (b), the window size is initialized to 2 so that the first two

Algorithm 2 Compute Coverage.

```
1: procedure CVG(pwd,infolist)
2:   windowSize  $\leftarrow$  2
3:   pwdlen  $\leftarrow$  len(pwd)
4:   matchtag  $\leftarrow$  [0]*pwdlen
5:   matchmark  $\leftarrow$  0
6:   cvg  $\leftarrow$  0
7:   while windowSize  $\leq$  len(pwd) do
8:     passseg  $\leftarrow$  pwd[0 : windowSize]
9:     if passseg = substring of anyinfo in infolist then
10:      for j  $\leftarrow$  matchmark to matchmark+windowSize do
11:        matchtag[j]  $\leftarrow$  windowSize
12:      end for
13:      if windowSize  $\neq$  len(pwd) then
14:        windowSize  $\leftarrow$  windowSize+1
15:      end if
16:    else
17:      matchmark  $\leftarrow$  matchmark+windowSize
18:      pwd  $\leftarrow$  pwd[windowSize :]
19:      windowSize  $\leftarrow$  2
20:    end if
21:  end while
22:  for eachitem in matchtag do
23:    cvg  $\leftarrow$  cvg + eachitem
24:  end for
25:  return cvg/(pwdlen * pwdlen)
26: end procedure
```

symbols in the password are covered. As "al" is a substring of Alice's name, a match is found. Therefore, we extend the window size by 1, and the tag array is updated as [2,2,0,0,0,0,0,0]. From step (c) to step (e), the window keeps growing since matches are continuously found. The tag array also keeps being updated. Until step (f), the window now covers "alice8," which is not a match of "alice" or "19880816." Therefore, the window size is reset to 2 and the window slides to the position of the symbol of the previous window that causes the mismatch (i.e., the position of "8"). The tag array remains unchanged. In step (g), the window of size 2 now covers "81," which is a substring of her birthdate, so again we extend the window by 1 and update the tag array to [5,5,5,5,5,2,2,0]. After the window grows by 1 in step (h), "816" is again found as a match. The tag array is updated to [5,5,5,5,5,3,3,3]. The window does not grow or slide anymore since it has reached the end of the password. In the last step (i), the tag array is ready to be used in computing the Coverage value.

Figure 2.1: An Example of Coverage Computing.



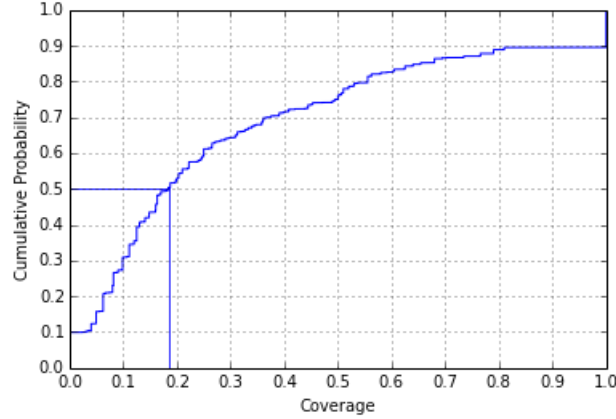
Gray boxes hold unvisited password symbols. Yellow and red boxes denote that the symbols inside are covered by the sliding window. White boxes denote that the symbols inside have been settled (i.e. the window stops extending).

Based on Equation 2.1, the coverage is computed as

$$CVG = \sum_{i=1}^2 \frac{l_i^2}{L^2} = \frac{5^2 + 3^2}{8^2} = 0.52.$$

Coverage is independent of password datasets. As long as we can build a complete string list of personal information, Coverage can accurately quantify the correlation between a user's password and its personal information. For personal information segments with the same length, Coverage stresses the continuation of matching. A continuous match is stronger than fragmented matches. That is to say, for a given password of length L , a matched segment of length l ($l \leq L$) has a stronger correlation to personal information than two matched segments of length l_1 and l_2 with $l = l_1 + l_2$. For example, a matched segment of length 6 is expected to have a stronger correlation than 2 matched segments of length 3. This feature of Coverage is desirable because multiple shorter segments (i.e., originated from different types of personal information) are usually harder to guess and more likely to involve a wrong match due to coincidence. Since it is difficult to differentiate a

Figure 2.2: Coverage distribution - 12306.



real match from a coincidental match, we would like to minimize the effect of false matches by taking squares of the matched segments to compute Coverage in favor of a continuous match. Coverage is independent of password datasets. As long as we can build a complete string list of personal information, Coverage can quantify the correlation between the password and these information.

2.4.2 Coverage Results on 12306

We compute the Coverage value for each user in the 12306 dataset and show the result as a cumulative distribution function in Figure 2.2. To easily understand the value of Coverage, we discuss a few examples to illustrate the implication of a roughly 0.2 Coverage. Suppose we have a 10-symbol-long password. One matched segment with length 5 will yield 0.25 Coverage. Two matched segments with length 3 (i.e., in total 6 symbols are matched to personal information) yield 0.18 Coverage. Moreover, 5 matched segments with length 2 (i.e., all symbols are matched but in a fragmented fashion) yield 0.2 Coverage. Apparently, Coverage of 0.2 indicates a fairly high correlation between personal information and a password.

The median value for a user's Coverage is 0.186, which implies that a significant portion of user passwords have relatively high correlation to personal information. Furthermore, around 10.5% of users have Coverage of 1, which means that 10.5% of passwords are perfectly matched to exactly one type of personal information. However, around 9.9% of users have zero Coverage, implying no use of personal information in their passwords.

The average Coverage for the entire 12306 dataset is 0.309. We also compute the average Coverages for male and female groups, since we observe that male users are more likely to include personal information in their passwords in Section 2.2.2.4. The average Coverage for the male group is 0.314, and the average Coverage for the female group is 0.269. This complies with our previous observation and indicates that the correlation for male users is higher than that of female users. Conversely, it also shows that Coverage works very well to quantify the correlation between passwords and personal information.

2.4.3 Coverage Usage

Coverage could be very useful for constructing password strength meters, which have been reported as mostly ad-hoc [38]. Most meters give scores based on password structure and length or blacklist commonly used passwords (e.g., the notorious “password”). There are also meters that perform simple social profile analysis, such as that a password cannot contain the user’s names or the password cannot be the same as the account name. However, these simple analysis mechanisms can be easily manipulated by slightly mangling a password, while the password remains weak. Using the metric of Coverage, password strength meters can be improved to more accurately measure the strength of a password. Moreover, it is straightforward to implement Coverage as a part of the strength measurement (only a few lines of Javascript should do). More importantly, since users cannot easily defeat the Coverage measurement through simple mangling methods, they are forced to select more secure passwords.

Coverage can also be integrated into existing tools to enhance their capabilities. There are several Markov model-based tools that predict the next symbol when a user creates a password [71,120]. These tools rank the probability of the next symbol based on the Markov model learned from dictionaries or leaked datasets, and then show the most probable predictions. Since most users would be surprised to find that the next symbol in their mind matches the tool’s output exactly, they may choose a less predictable symbol. Coverage helps to determine whether personal information prediction ranks high enough in probability to remind a user of avoiding the use of personal information in password creation.

2.5 Personal-PCFG

After investigating the correlation between personal information and user passwords through measurement and quantification, we further study their potential usage to crack passwords from an attacker’s point of view. Based on the PCFG approach [121], we develop Personal-PCFG as an individual-oriented password cracker that can generate personalized guesses towards a targeted user by exploiting the already known personal information.

2.5.1 Attack Scenarios

We assume that the attacker knows a certain amount of personal information about the targets. The attacker can be an evil neighbor, a curious friend, a jealous husband, a blackmailer, or even a company that buys personal information from other companies. Under these conditions, targeted personal information is rather easy to obtain by knowing the victim personally or searching online, especially on social networking sites (SNS) [50, 72]. Personal-PCFG can be used in both offline and online attacks.

In traditional offline password attacks, attackers usually steal hashed passwords from victim systems and then try to find out the unhashed values of these passwords. As a secure hash function cannot be simply reversed, the most popular attacking strategy is to guess and verify passwords by brute force. Each guess is verified by hashing a password (salt needs to be added) from a password dictionary and comparing the result to the hashed values in the leaked password database. High-probability password guesses can usually match many hashed values in the password database and thus are expected to be tried first for efficiency purpose. For offline attacks, Personal-PCFG is much faster in guessing the correct password than conventional methods, since it can generate high-probability personalized passwords and verify them first.

For an online attack, the attacker does not even have a hashed password database, so he or she instead tries to log in directly to the real systems by guessing the passwords. It is more difficult to succeed in online attacks than offline attacks because online service systems usually throttle

login attempts in a given period of time. If the attempt quota has been reached without inputting a correct password, the account may be locked temporarily or even permanently unless certain actions are taken (e.g., call the service provider). Therefore, online attacks require accurate guesses, which can be achieved by integrating personal information. Personal-PCFG can crack around 1 out of 20 passwords within only 5 guesses.

2.5.2 A Revisit of PCFG

Personal-PCFG is based on the basic idea of PCFG method [121] and provides an extension to further improve its efficiency. Before we introduce Personal-PCFG, we briefly revisit principles of PCFG. PCFG pre-processes passwords and generates base password structures such as “ $L_5D_3S_1$ ” for each of the passwords. Starting from high-probability structures, the PCFG method substitutes the “D” and “S” segments using segments of the same length learned from the training set. These substitute segments are ranked by probability of occurrence learned from the training set. Therefore, high-probability segments will be tried first. One base structure may have a number of substitutions; for example, “ $L_5D_3S_1$ ” can have “ $L_5123!$ ” and “ $L_5691!$ ” as its substitutions. These new representations are called pre-terminal structures. No “L” segment is currently substituted since the space of alpha strings is too large to learn from the training set. Next, these pre-terminals are ranked from high probability to low probability. Finally “L” segments are substituted using a dictionary to generate actual guesses. Besides, PCFG method also carries an efficient algorithm to enumerate passwords from high probability to low probability on the fly. These guesses are hashed to compare with the values in password databases. Since PCFG can generate statistically high-probability passwords first, it can significantly reduce the guessing number of traditional dictionary attacks.

2.5.3 Personal-PCFG

Personal-PCFG leverages the basic idea of PCFG. Besides “L,” “D,” and “S” symbols, it features more semantic symbols, including “B” for birthdate, “N” for name, “E” for email address, “A” for account name, “C” for cellphone number, and “T” for ID number. Richer semantics make Personal-

PCFG more accurate in guessing passwords. To make Personal-PCFG work, an additional personal information matching phase and an adaptive-substitution phase are added to the original PCFG method. Therefore, Personal-PCFG has 4 phases in total, and the output of each phase will be fed to the next phase as input. The output of the last phase is the actual guesses. We now describe each phase in detail along with simple examples.

2.5.3.1 Personal Information Matching

Given a password, we first match the entire password or a substring of the password to its personal information. The detailed algorithm is similar to Algorithm 1. However, this time we also record the length of the matching segment. We replace the matched segments in a password with corresponding symbols and mark each symbol with its length. Unmatched segments remain unchanged. For instance, we assume Alice was born on August 16, 1988, and her password is “helloalice816!”. The matching phase will replace “alice” with “ N_5 ” and “816” with “ B_3 .” The leftover “hello” is kept unchanged. Therefore the outcome of this phase is “*helloN₅B₃!*”

2.5.3.2 Password Pre-processing

This phase is similar to the pre-processing routine of the original PCFG; however, based on the output of the personal information matching phase, the segments already matched to personal information will not be processed. For instance, the sample structure “*helloN₅B₃!*” will be updated to “*L₅N₅B₃S₁!*” in this phase. Now the password is fully described by semantic symbols of Personal-PCFG, and the output in this phase provides base structures for Personal-PCFG.

2.5.3.3 Guess Generation

Similar to the original PCFG, we replace “D” and “S” symbols with actual strings learned from the training set in descending probability order. “L” symbols are replaced with words from a dictionary. Similar to PCFG [121], we output the results on the fly, so we do not need to wait for all of the possible guesses being calculated and sorted. The guesses keep being generated for next step. Note that we have not replaced any symbols for personal information, so the guesses are still not actual guesses. We do not handle personal information in this step, since personal information of

each user is different. Thus, the personal information symbols can only be substituted until the target user is specified. Therefore, in this phase, the base structures only generate pre-terminals, which are partial guesses that contain part of actual guesses and part of Personal-PCFG semantic symbols. For instance, the example " $L_5N_5B_3S_1$ " is instantiated to " $helloN_5B_3!$ " if "hello" is the first 5-symbol-long string in the input dictionary and "!" has the highest probability of occurrence among 1-symbol special characters in the training set. Note that for "L" segments, each word of the same length has the same probability. The probability of "hello" is simply $\frac{1}{N}$, in which N is the total number of words of length 5 in the input dictionary.

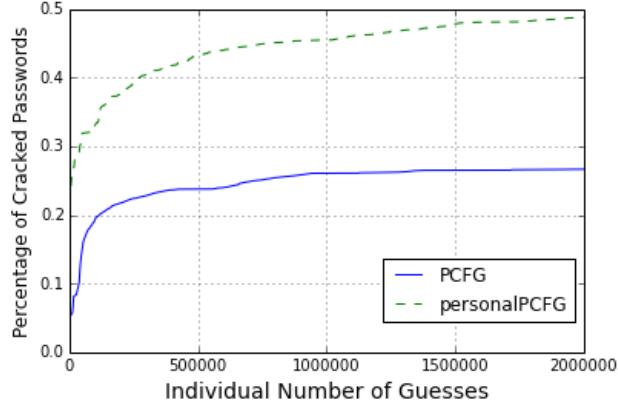
2.5.3.4 Adaptive Substitution

In the original PCFG, the output of guess generation can be applied to any target user. However, in Personal-PCFG, the guess will be further instantiated with personal information, which is specific to only one target user. Each personal information symbol is replaced by corresponding personal information of the same length. If there are multiple candidates of the same length, all of them will be included for trial. In our example " $helloN_5B_3!$," " N_5 " will be directly replaced by "alice." However, since " B_3 " has many candidate segments and any length 3 substring of "19880816" may be a candidate, the guesses include all substrings, such as "helloalice198!," "helloalice988!," ..., "helloalice816!" We then try these candidate guesses one by one until we find one candidate that matches exactly the password of Alice. Note that on the contrary of having multiple candidates, not all personal information segments can be replaced because same-length segments may not always be available. For instance, a pre-terminal structure " $helloN_6B_3!$ " is not suitable for Alice since her name contains only 5 characters. In this case, no guess from this structure should be generated for Alice.

2.5.4 Cracking Results

We compare the performance of Personal-PCFG and the original PCFG using the 12306 dataset, which has 131,389 users. We use half of the dataset as the training set, and the other half as the testing set. For the "L" segments, both methods need to use a dictionary, which is a critical part

Figure 2.3: PCFG vs. Personal-PCFG (Offline).



of password cracking. To eliminate the effect of an unfair dictionary selection, we use “perfect” dictionaries in both methods. Perfect dictionaries are dictionaries we collected directly from the testing set, so that any string in the dictionary is useful and any letter segments in the passwords must appear in the dictionary. Thus, a perfect dictionary is guaranteed to find correct string segments efficiently. In our study, both the PCFG perfect dictionary and the Personal-PCFG perfect dictionary contain 15,000 to 17,000 entries.

We use *individual number of guesses* to measure the effectiveness of Personal-PCFG compared with PCFG. The individual number of guesses is defined as the number of password guesses generated for cracking each individual account (e.g., 10 guess trials for each individual account), which is independent of the password dataset size. In Personal-PCFG, the aggregated individual number of guesses (i.e., the total number of guesses) is linearly dependent on the password dataset size. By contrast, in a conventional cracking strategies like PCFG, each guess is applied to the entire user base, and thus the individual number of guesses equals the total number of guesses. Regardless of such discrepancies between Personal-PCFG and conventional cracking methods, the bottleneck of password cracking lies in the number of hashing operations. Due to the salting mechanism, the total number of hashes is bounded by $G \cdot N$ for both Personal-PCFG and other password crackers, where G is the individual number of guesses and N is the size of the dataset.

Given the different number of guesses, we compute the percentage of those cracked passwords in the entire password trial set. Figure 2.3 shows the comparison result of the original PCFG and Personal-PCFG in an offline attack. Both methods have a quick start because they always try

Figure 2.4: PCFG vs. Personal-PCFG (Online).

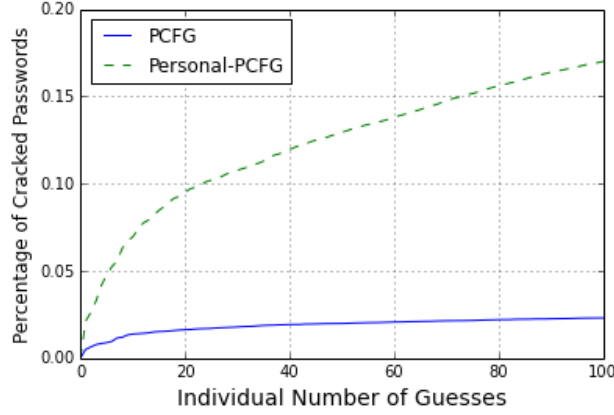
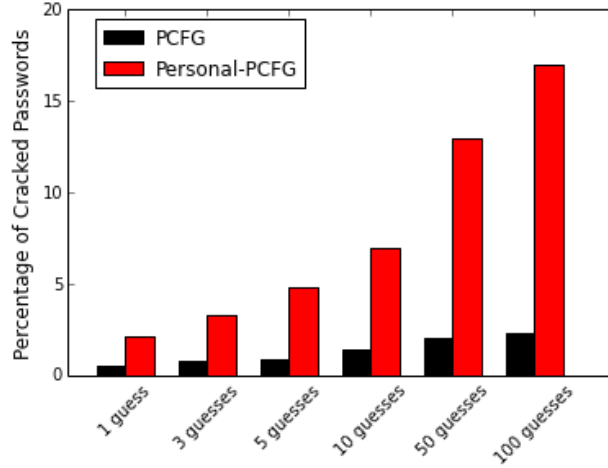


Figure 2.5: Representative Points – Online attacks.



high probability guesses first. Figure 2.3 clearly indicates that Personal-PCFG can crack passwords much faster than PCFG does. For example, with a moderate size of 500,000 guesses, Personal-PCFG achieves a similar success rate that can be reached with more than 200 million guesses by the original PCFG. Moreover, Personal-PCFG is able to cover a larger password space than PCFG because personal information provides rich personalized strings that may not appear in the dictionaries or training set.

Personal-PCFG not only improves the cracking efficiency in offline attacks but also increases the guessing success rate in online attacks. Online attacks are only able to try a small number of guesses in a certain time period due to the system constraint on the login attempts. Thus, we limit the

number of guesses to be at most 100 for each target account. We present the results in Figure 2.4, from which it can be seen that Personal-PCFG is able to crack 309% to 634% more passwords than the original PCFG. We then show several representative guessing numbers in Figure 2.5. For a typical system that allows 5 attempts to input the correct passwords, Personal-PCFG is able to crack 4.8% of passwords within only 5 guesses. Meanwhile, the percentage is just 0.9% for the original PCFG, and it takes around 2,000 more guesses for PCFG to reach a success rate of 4.8%. Thus, Personal-PCFG is more efficient to crack the passwords within a small number of guesses.

Therefore, Personal-PCFG substantially outperforms PCFG in both online and offline attacks, due to the integration of personal information into password guessing. The extra requirement of Personal-PCFG on personal information can be satisfied by knowing the victim personally or searching on social networking sites (SNS).

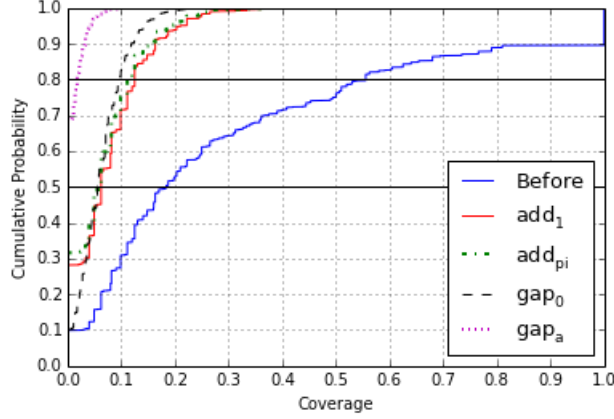
2.6 Password Protection

In almost all systems, users are able to choose and update their passwords. However, they may sacrifice password security for usability since a long and random secure password is less memorable. As user passwords are easier to be compromised when their personal information is available to the attacker, we investigate how users can protect their passwords against such attacks.

To increase the password security while retaining good memorability, we suggest the *Distortion Function*, which performs a transformation on user passwords. A distortion function converts user passwords to be more secure by breaking password semantics. Therefore, the user only needs to remember the original password and apply a simple function on it to create a stronger password. This distortion function can be chosen by users, so it could be either linear or non-linear.

We conduct a proof-of-concept study to show the effectiveness of a distortion function on password security. In this study, two types of distortion functions are introduced. The first type maps each password character to another character. For instance, add_1 function simply replaces each letter with the one 1 letter later in the alphabet and replaces a single digital number i with $(i + 1) \bmod 10$. It is similar to the Caesar Cipher [66]. In another example, add_{π} is a non-linear function that shifts password letters by a corresponding position specified by π , which is 314159... . It shifts a letter

Figure 2.6: Coverage distribution.



to N positions later in the alphabet with wraparound, where N is the corresponding digit of π . For instance, "abc" becomes "dcb" after applying this distortion function. The second type of distortion function adds an extra fixed character between any pair of characters in passwords. The length of a password becomes $2l - 1$, where l is the length of the original password. We call this distortion function gap_x , in which "x" represents the extra symbol. For example, when $x = "a"$, Alice's password "alice816" will be extended to "aalaiacaea8ala6" after the distortion function is applied.

The distortion function must be simple enough for users to remember and generate the passwords. We apply a number of easy-to-remember distortion functions on each of the passwords in the 12306 dataset individually and calculate the Coverage for the converted passwords. As Figure 2.6 shows, the distortion functions are effective in increasing password security by greatly reducing the correlation between user passwords and personal information. Moreover, we notice that the impacts of various distortion functions are also different. For example, add_1 performs the best (Coverage is 0 for all users, so it is not shown in Figure 2.6) since users rarely have special characters in their personal information. Surprisingly, the non-linear add_{pi} function does not produce a better result than other linear functions such as add_1 because digits preferred by Chinese users are more likely to have coincidentally wrong matches due to its low entropy.

We conclude that distortion functions can mitigate the problem of including personal information in user passwords without significantly sacrificing password usability. Moreover, distortion functions are also a cure for semantics-aware password cracking methods [114], which leverage semantic pat-

terns in passwords to crack other passwords. After applying a distortion function, the semantic pattern is no longer available. Distortion functions are also effective against PCFG [121], since it generates unrecognizable letter segments, which are not likely to be covered in commonly-used password dictionaries.

What differentiates the use of a distortion function from using an extra piece of user-chosen secret is that it breaks password semantics, which makes it much harder for an attacker to interpret since there could be many possible (password, function) pairs that produce the same final password. Thus, it becomes increasingly difficult for an attacker to learn how users construct their passwords, and the inaccurate training cripples the efficiency of training-based attack. Furthermore, we do not claim that a distortion function is able to eliminate the trade-off between security and usability. Instead, the distortion function is only used to effectively mitigate the problem of personal information residing in passwords.

2.7 Discussion

2.7.1 Limitation

In most of our study, only a single dataset is used. Most users of the 12306 website are Chinese, and the number of males and females is not balanced. Consequently, there might be cultural, language, and gender biases on the analytical results. Moreover, the effectiveness of the Coverage metric and Personal-PCFG is only validated on a single website. Publicly available password datasets leaked with personal information are very rare. We have done an estimation on English-based datasets, which shows that no matter the language or culture, people include personal information in their passwords, though to slightly different extents. However, although we have tried to extend the analytical work to more datasets, it is infeasible to test the effectiveness of Coverage and Personal-PCFG since personal information is directly derived from passwords.

2.7.2 Ethical Considerations

Though using leaked password datasets for more accurate and convincing study has been a mainstream method on password studies, we do realize that studying leaked datasets involves ethical

concerns. We only use the datasets for researching purpose. All data are carefully stored and used. We will not expose any personal information or password or use this information in any way other than for research use.

2.8 Conclusion

In this work, we conduct a comprehensive quantitative study on how user personal information resides in human-chosen passwords. To the best of our knowledge, we are the first to systematically analyze personal information in passwords. We have some interesting and quantitative discoveries such as 3.42% of the users in the 12306 dataset use their birthdate as a password, and male users are more likely than female users to include their name in passwords. We then introduce a new metric, Coverage, to accurately quantify the correlation between personal information and a password. Our coverage-based quantification results further confirm our disclosure on the serious involvement of personal information in password creation, which makes a user password more vulnerable to a targeted password cracking. We develop Personal-PCFG based on PCFG but consider more semantic symbols for cracking a password. Personal-PCFG generates personalized password guesses by integrating personal information in the guesses. Our experimental results demonstrate that Personal-PCFG is significantly faster than PCFG in password cracking and eases the feasibility of mounting online attacks. Finally, we propose using distortion functions to protect weak passwords that include personal information. Through a proof-of-concept study, we confirm that distortion functions are effective in defending against personal-information-related and semantics-aware attacks.

Chapter 3

BluePass: A Secure Hand-free Password Manager

3.1 Introduction

Text-based password still dominates online authentication despite that it has long been plagued by a well-known and long-standing problem: the wide use of weak password. Due to limited human memory, users tend to choose weak passwords [18, 22]. However, weak passwords are easy to guess and thus are vulnerable to a variety of attacks [19, 81, 87, 115, 123]. Today's increasing number of accounts a user possesses even worsen the problem since the user poorly manage their passwords. For example, on average users may reuse one password for as many as 3.9 online accounts [44]. As such, instead of impractically expecting users to select a strong password for each account, password managers are developed as built-in or standalone gadgets to help users manage their credentials. A password manager includes a vault that stores all encrypted passwords of a user, and the user only needs to remember one master password, which is used to generate the decryption key to the vault, to access all the passwords in the vault. To support user authentication on different devices, password managers usually synchronize the vaults to their own servers and provide a downloading service to their users. However, a password manager has its own security and usability problem. For example, password managers usually synchronize the local vault to the remote server, which makes data breach possible [5]. Furthermore, to enhance usability, many browser built-in password

managers do not necessarily need a master password, which makes it vulnerable to unauthorized use and meanwhile sacrifices portability. Even being used, a master password becomes a single point of failure. Usability issues of a password manager may even lead to reduced security, stemming from incomplete user mental models [31].

For critical online services, users may desire more secure authentication than merely password. Toward this end, two-factor authentication (2FA) is proposed to include another layer of protection to user accounts. Nowadays many leading service providers such as Google and Microsoft, have integrated 2FA into their online systems. However, 2FA suffers from limited adoption due to undesired extra burden on users. It is estimated that in 2015, only around 6.4% of Google users are using 2FA [93]. In order to improve usability, transparent 2FA has been proposed [34, 88] by leveraging additional devices (mainly user smartphones) to automatically complete the enhanced authentication procedure without user involvement. However, these approaches are hard to deploy because of imperative modifications at both the web server and the client sides.

In this project, we propose BluePass, an enhanced password manager that partially inherits the security benefit of 2FA to improve the security and usability of existing password managers. One of the key features of BluePass is to isolate the storage of the password vault from that of the decryption key. Here the password vault is the set of all the encrypted site passwords of a user. Specifically, the password vault is stored locally in a mobile device (e.g., a user’s smartphone) and the decryption key is stored in the BluePass server, which can be accessed and downloaded only once to a computer after authentication through a master password. The mobile device communicates with the computer using Bluetooth in a transparent manner. When a user needs to log in a website, the computer will automatically request the site password from the mobile device. The encrypted site password will then be delivered through Bluetooth. Afterwards, the computer is able to decrypt the site password using the local decryption key and auto-fill the web forms for the user. BluePass relies on Bluetooth for communication rather than other channels, because Bluetooth can be both transparent to users and a subtle indicator of co-location of the user mobile device.

BluePass is secure since it does not store password vaults on a server and is not vulnerable to massive password breach. Furthermore, a server data breach is likely to leak both password vaults

and hashed master passwords. By cracking the master password table offline, it is almost guaranteed that most master passwords can be cracked out, given today’s computing power and the weakness of user-selected passwords. Attackers are given direct access to password vaults under such a case, since the vault decryption key is generated from the master password. By contrast, in BluePass, the password vault and its decryption key are stored separately, and decryption key is not generated from master passwords, losing one of them will not practically leak any password.

While BluePass itself uses 2FA, it does not require any modifications on the website servers. Thus, the underlying password framework remains unaltered, i.e., logging into a website still only needs one site password. BluePass is also usable since it demands little effort to configure on the computer and no extra effort from a user to authenticate afterwards. .

We implement a BluePass prototype in Android and Google Chrome and evaluate its efficacy in terms of security, overhead, and usability. First, we conduct a comprehensive security analysis to demonstrate that BluePass can defend against various attacks. Then we evaluate the auto-fill time latency of BluePass by recording the time between login forms being detected and the forms being automatically filled. We also run a series of experiments, in which we retrieve passwords under different frequencies, to measure the energy overhead of BluePass. Based on our experimental results, BluePass is energy efficient while automatically filling in the login forms with user-unperceived latency. Afterwards, we conduct a user study including 31 volunteers to examine the usability of BluePass. The results show the test subjects regard BluePass as both secure and usable. Moreover, the majority of testers report that they are willing to use BluePass to manage their passwords.

3.2 System Overview and Threat Model

Before presenting the BluePass system, we first introduce important BluePass notations for clarification purposes.

- *BluePass server*: a server that is mainly responsible for registering users and distributing keys to user computers.
- *Key pair* (K_1, K_2): a pair of RSA keys that are used by the mobile device and the computer

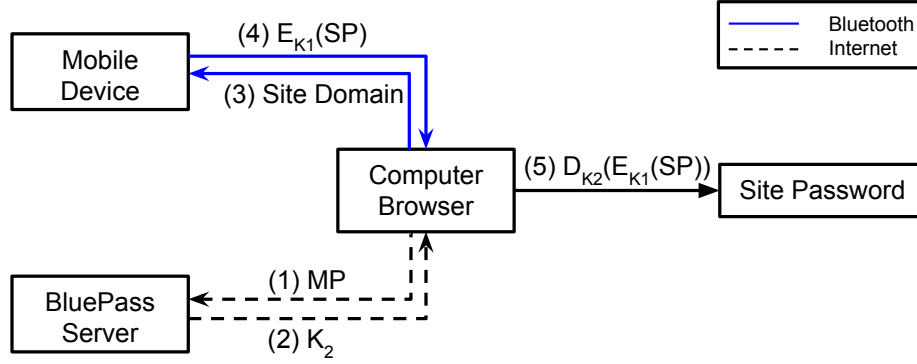
to encrypt/decrypt site passwords. K_1 is only stored in the mobile phone while K_2 is stored in the BluePass server for re-distribution. We manage to use only one pair of keys to protect bi-directional communication, and the details can be found in Section 3.8.1.

- *Master password (MP)*: a user uses its master password to authenticate itself to the BluePass server and retrieve its own decryption key K_2 . A master password is the only password a user needs to remember.
- *Site password (SP)*: passwords to access online services, which will be encrypted by K_1 and then stored in the BluePass mobile application .
- *Trusted computer*: a computer that the user trusts, such as the user's personal computer. It stores the decryption key K_2 for a long term.
- *Untrusted computer*: a computer that the user does not trust, such as a library computer. The decryption key K_2 must be retrieved from the BluePass server every time a browser is opened in the untrusted computer. K_2 is only temporarily stored in a browser instance, and is removed when the browser instance is terminated.
- *Client-side (computer/browser) application*: the user installs it on the computer, which is in charge of detecting and auto-filling login forms, communicating with the mobile device, and decrypting the received site passwords.
- *Mobile application*: the user installs the app on its mobile device. The app stores the encrypted site passwords and delivers the encrypted site passwords to the user computer through Bluetooth.

3.2.1 System Overview

BluePass works on two premises. First, a site password can only be recovered by having both the encrypted site password $E_{K_1}(SP)$ that is only stored in the mobile device and the corresponding decryption key K_2 that is distributed through the BluePass server. Second, the encrypted site password $E_{K_1}(SP)$ can only be retrieved from the mobile device to the user computer through

Figure 3.1: BluePass Authentication



Solid lines indicate that the process needs to be run whenever a site password is requested. Dashed lines indicate that it does not necessarily run when a site password is requested. In a *trusted computer*, it runs only once for a long term while in a *untrusted computer*, it runs once in each browser instance.

Bluetooth, which requires the proximity of the two devices. The flow chart of BluePass password authentication is shown in Figure 3.1.

The working mechanism of BluePass mainly includes three phases, which are detailed as follows.

Phase 1: Registration is a once-in-a-lifecycle operation, in which a user needs to register for the BluePass service. The user installs the BluePass mobile app on its mobile device and uses its master password to log into the BluePass account. The mobile device is then initialized with an empty password vault.

Phase 2: Configuration is to install and configure the user devices. First, a client-side application needs to be installed on the user computer. Then, the user will log into the BluePass server and download the decryption key K_2 into the computer. The user will store the key either for a long term or temporarily, depending on whether the computer is trusted or untrusted. Note that the installation of the client-side application on a computer is also a one-time operation. The retrieval of K_2 from the BluePass server is needed each time opening a browser only when the user is on a untrusted computer.

Phase 3: Authentication is almost transparent to the user. In a trusted device, the user only needs to carry the registered mobile phone and wait for the passwords being automatically filled. In a untrusted device, the user needs to re-enter the master password every time a new browser instance is opened since the key K_2 is deleted when a browser instance is closed.

3.2.2 Threat Model

Attackers aim at stealing one or (preferably) all site passwords in the password vault. In the design of BluePass, all the site passwords of a user are encrypted and stored in the user's mobile device. We assume that the attacker cannot access the encrypted site passwords in the mobile device and knows the decryption key from the computer at the same time.

All attacks can be classified into two categories: *co-located attacks* and *remote attacks*. A co-located attack can only happen within the Bluetooth communication range of the user mobile device, while a remote attack can be launched from anywhere. In a co-located attack, since the attacker could access the encrypted site passwords through sniffing, we must prevent the decryption key from falling into the hand of the attacker. Therefore, both the BluePass server and the master password cannot be compromised. Moreover, the communications for key distribution must be protected. By contrast, in a remote attack, since the attacker cannot access the mobile device through Bluetooth, either the BluePass server or the master password could be compromised. Also, no secure communication is required for key distribution. As the Bluetooth reachability is very limited (33 feet for class 2 Bluetooth devices), a co-located attack is much more difficult to launch than a remote attack.

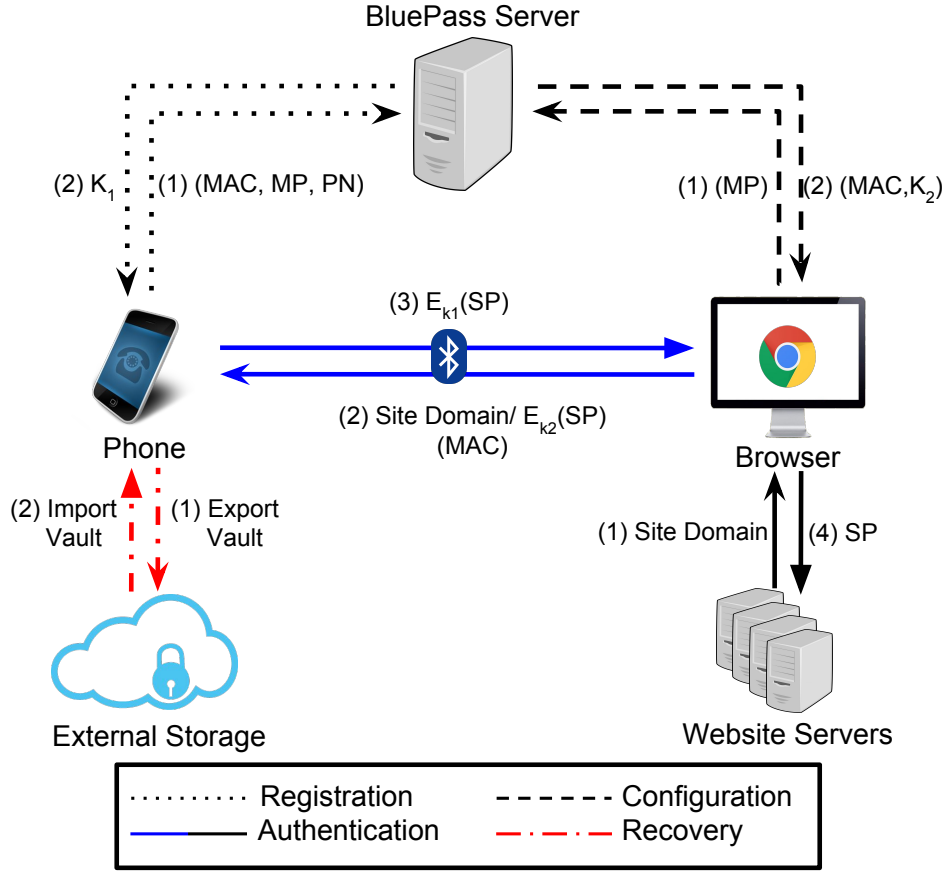
3.3 System Architecture

3.3.1 Core Functions

As mentioned in Section 3.2.1, BluePass mainly consists of three phases. The first two phases, registration and configuration of BluePass, are mostly one-time effort; however, the third phase, authentication, will be triggered each time a user needs to log in a website. Figure 3.2 illustrates BluePass architecture and the data flow of these three phases.

Registration. The black dotted lines in Figure 3.2 show the registration process. To register a BluePass service, the user only needs to download a BluePass application to the mobile phone and create a master account on the BluePass server. The creation of the master account is similar to the creation of an account in any website. Upon logging into the master account on the mobile

Figure 3.2: BluePass Architecture



app, the user can choose to bind the mobile device. The binding process should follow a traditional 2FA mechanism. Namely, the user re-authenticate herself with another authentication factor, for example, a sms. Afterward, the device information, specifically, the MAC address of the device Bluetooth, will be uploaded to the BluePass server. The MAC address is used for the client-side application to automatically locate the associated mobile device without user involvement. For a newly associated device, the BluePass Server generates a pair of asymmetric keys (K_1, K_2) . It then distributes K_1 to the mobile phone and keeps only K_2 on the server side. We list the database of the BluePass server in Table 3.1 and that of the mobile device in Table 3.2 populated with made-up data. The registration should only be done once on the mobile device. After registration, the mobile device is initialized as a password vault. Note that the key pair of (K_1, K_2) is not used as a conventional public-key pair, where the public key is known to all and the private key is kept in secret. Instead, the key pair is used for a two-way communication channel and both of them should

be kept in secret.

After registration, the user has initialize a password vault in its own mobile device and associated the BluePass account with this device.

Table 3.1: Server Side Data

Username	Salt	$H(MP + Salt)$	K_2	Device MAC address
Alice	ifu92@fb	<i>a4f3b3c9e61b838f8cda07...</i>	<i>VDSnrzjqFBY9...</i>	BC:F5:AC:9D:9A:57
Bob	01dm.a<w	<i>daa4a403bfec911a3ef199...</i>	<i>yKhTC3dNAkE...</i>	BC:F5:AC:9D:9A:58

Table 3.2: Mobile Device Data

Domain	username	K1	$E_{K_1}(Password)$
.yahoo.com/	aliceweb1	<i>AoGAKooOHMT...</i>	<i>Encrypted_Password₁</i>
.yahoo.com/	aliceweb2	<i>VN9SdOeFbo4w...</i>	<i>Encrypted_Password₂</i>
.google.com/	aliceweb2	<i>B1FUeDXiqv4j...</i>	<i>Encrypted_Password₃</i>

Configuration. The computer needs to be configured to run BluePass, which is shown in the dashed black lines in Figure 3.2. The user installs and runs a client-side application, and then logs into the BluePass server to fetch the Bluetooth MAC address of the mobile device and K_2 generated during the registration. At this point, the user can choose whether the computer is trusted or not. If the computer is trusted, the Bluetooth MAC address and K_2 will be stored in the browser for a long term. Otherwise, they will be deleted after the user closes the current browser instance. Knowing the device Bluetooth MAC address enables the computer to pair with the device automatically by using RFCOMM insecure mode, in which the Bluetooth data is broadcasted and the target MAC address is specified in the data. BluePass does not rely on secure Bluetooth communication. Using RFCOMM insecure mode enhances usability while not degrading security.

Authentication. The authentication phase is the only phase that a user will constantly experience during use of Bluepass. The solid lines in Figure 3.2 show the data flow of BluePass authentication process. First, the user directs the browser to a website it wants to login. The BluePass client-side application will examine the Document Object Model (DOM, which is a tree structure representing the webpages) of the returned page and check the existence of a login form. If a login form is present, the application requests the corresponding credentials from the mobile phone using Bluetooth. After receiving the request, the mobile application returns the encrypted credentials. If no related credential exists, BluePass will instead respond with a "NO_PASSWORD"

flag. We realize that auto-filling in a non-HTTPS environment is vulnerable to JavaScript injection attacks [102], so we only do auto-filling for websites that are based on HTTPS. For other websites, BluePass will pop up a window for a user’s consent before filling the login form. Note that none of the above steps require any user interactions. This fully automated authentication enables users to login a website in a hand-free manner. When there exists more than one account for a specific website, the browser will let the user choose an account to be decrypted and automatically filled in the forms since there is no way to predict which account will be used.

3.3.2 Account Management

Account management is essential to a password manager. Users should be able to add, edit, or delete the credentials in BluePass. These functions must be correctly designed to guarantee the security of BluePass.

The addition of an online account into BluePass can be done when a user has manually inputted the login credentials into a new website. BluePass adopts a similar approach just as current browser built-in password managers. If the “NO_PASSWORD” flag is sent back, the browser knows that no login credentials are associated to this particular website. If the user manually inputs the credentials, the browser will capture the value in the form before submission and prompt a non-intrusive dialog window, asking whether the user wishes to store the login information into BluePass. Specifically, there are three options: “yes”, “not this time”, and “never”. If “yes” is chosen, the browser will encrypt the credentials using key K_2 and send it to the mobile device (see Figure 3.2). The mobile device will decrypt the information using K_1 and encrypt it again using K_1 . Mathematically the process is denoted as $E_{K_1}(D_{K_1}(C))$, in which $C = E_{K_2}(SP)$. Then the encrypted credentials are stored in the BluePass database.

The edition of an online account is similar to the addition process. The browser monitors if the user has modified the value in the login form when being submitted. If the password is changed, the browser will prompt a dialog that asks for user permission to update the login credentials in BluePass. Upon user consent, the browser will send the updated values in an encryption and decryption procedure similar to that of adding a new account. Note that the chosen option of

“never” should also be recorded in the password vault, which prevents the dialog from prompting repeatedly. In this case, the password vault records the domain name and the username without storing a password. When an empty password is passed back, the application is acknowledged that the user does not wish to store the login credentials. The revocation of the “never” status can be done in the administration page in the mobile applications.

The deletion of login credentials can also be done on the mobile application’s administration page. The mobile application shows a list of websites whose site passwords are stored in the mobile phone. The user can choose to delete one of the websites’ login credentials. However, the user needs to manually input the website’s URL and login credentials. Before the deletion is granted, the user must input the correct master password. This will prevent an attacker from manipulating the user’s online accounts.

3.3.3 Recovery

When using a cloud-based password manager, users can backup their password vaults on the server side. On the contrary, BluePass is de-centralized and stores local copies on mobile devices. Though users usually do not lose their mobile devices quite often, it is essential for BluePass to back up and recover the password vault when the mobile devices are lost, which is illustrated with red lines in Figure 3.2.

Users can choose to back up their vaults to an external storage including a portable hard disk, a USB, or a cloud storage. If a user loses the mobile device, it can recover the vault from the external storage. Backing up the vault to a user-owned physical device may require the user to periodically back up and synchronize the password vault to the external storage device. Alternatively, BluePass allows users to synchronize their password vaults to a cloud drive provider. Nowadays many large drive providers, such as Google Drive or Dropbox, have published APIs to facilitate data synchronization. Note such design still ensures the 2FA design of BluePass – an attacker needs to breach both the BluePass Server and the cloud provider server to collect the two necessary pieces of secret.

3.4 Security Analysis

BluePass is secure in a sense that as long as a user does not lose two factors at the same time, the user's login information is safe. We conduct a security analysis on BluePass to verify the robustness of BluePass against various attack vectors.

3.4.1 Two-Factor Security

We have introduced that BluePass relies on the premise that two factors need to be possessed to derive a site password. The two factors are user mobile device and a master password. Now we discuss the security of BluePass when one of the factors is compromised.

Master password. An attacker may be able to compromise the master password of a user, which can be done through different ways such as guessing, phishing, shoulder-surfing, etc. The compromise of a trusted computer is also equivalent to losing the master password because the only purpose of having the master password is to retrieve K_2 from BluePass server, which can be directly extract K_2 from a trusted computer. In such scenarios, the attacker is able to obtain key K_2 . However, if the attacker does not have the password vault of the user, K_2 is merely a meaningless token and the security of BluePass holds. Besides, the user is able to change the master password and re-generate a new key pair.

Mobile device. If an attacker gains access to the mobile device by either compromising the device or stealing the device, it may be able to access the encrypted password vault and the encryption key K_1 . However, without the decryption key K_2 , the attacker cannot decrypt the site passwords from the encrypted password vault. Unlike cloud-based password managers, BluePass does not keep master password and the vault on the same storage, thus obtaining K_2 together with the password vault is not practical. Moreover, the mobile phone itself may have its own protection, such as an unlock code or fingerprint verification, and remote data erasal.

3.4.2 Data Breach and Brute-force Attacks

A serious threat to a password manager is data breach. Under this scenario, the attacker may be able to mount a brute force attack against the master password of a user. In a normal password

manager such as LastPass or 1Password, the loss of a master password also means the loss of an entire password vault, namely, when an attacker successfully mounts a brute force attack against the master password, it can also retrieve all the passwords from the password vault since the key used to encrypt the vault is derived from the master password. Again, BluePass does not centralize the password vault storage. Instead, the password vault of a user is stored locally in its own mobile device. A server data breach would at most leak the user master passwords and then further leak the decryption keys. However, as the password vault of each user is not stored at the BluePass server, a data breach at the BluePass server cannot break BluePass.

On the other hand, assuming that a password vault is lost from a user’s mobile device, we believe that brute-force cracking such an encrypted password vault is impractical given the current computing power. We emphasize that the password vault is protected by K_1 , which is 2048-bit long randomly generated RSA key. Cracking K_1 is much harder than cracking a master password, which is generated by a human user within limited and predictable password space.

3.4.3 Broken HTTPS or Bluetooth

If an attacker compromises the HTTPS communication, it will be able to steal the encryption/decryption key pair (K_1, K_2) of a user. However, K_1 and K_2 are only transmitted through the web when a user installs BluePass on its mobile device (K_1) or when the user log on BluePass from a new computer (K_2), which makes the attack strictly time sensitive. Even though, having the key pair does not help the attacker to identify any of the user’s site password, unless the attacker can also eavesdrop on the Bluetooth connection (i.e., co-located attack) to capture the encrypted password in transmission. On the other hand, eavesdropping Bluetooth alone does not compromise BluePass either, since the content is encrypted.

To succeed, the attacker needs to compromise both HTTPS and Bluetooth communications to steal site passwords from users. However, such a successful attack is very difficult to launch, due to time (to steal the keys) and location (to eavesdrop the Bluetooth) constraints. Furthermore, a large scale attack is infeasible since Bluetooth signals can only be sniffed within a short range.

3.5 Implementation

BluePass consists of three major components that cooperate with each other on user authentication, namely, a BluePass server for user registration and key distribution, a BluePass client application on the laptop for detecting and auto-filling the website login forms, and a BluePass app on the mobile phone serving as the password vault and administration console. We build the BluePass client application in a Macbook Air running OS X 10.10.4 and Chrome 46.0.2490.80. We implement the BluePass app on a Nexus 5 running Android version 4.4.2.

3.5.1 BluePass Server

We implement a BluePass server using CherryPy [45], a python web framework. We use self-signed certificate in https to protect communication. The key pair (K_1, K_2) is generated using Pycrypto¹ on the server side. Sqlite database is used to store user data (see Table 3.1 for detail). When registering to the service, we do not use the standard 2FA to verify the phone number since it is not necessary for evaluation and user study. After registration, the user needs to log in BluePass on both mobile application to upload mobile phone Bluetooth MAC address and download K_1 and client-side application to download K_2 and Mobile phone Bluetooth MAC address.

3.5.2 BluePass Client-side Application

We build the BluePass client-side application on Chrome platform, which consists of 2 modules: one Chrome application for Bluetooth communication and one Chrome extension for password auto-filling. We use two modules because currently Chrome extension does not support Bluetooth API while Chrome application does. However, only Chrome extensions allow reading and modifying the DOM of web pages, which unavoidably makes us separate client-side application functionality into 2 modules. Chrome application is more like a native application, but it is built on Chrome platform to deliver content in HTML, CSS and Javascript (e.g., Google Doc, Google Drive). It uses the *chrome.Bluetooth* API to connect to the Bluetooth device and then communicate with the smart

¹<https://www.dlitz.net/software/pycrypto/>

phone through Bluetooth. The Chrome extension is responsible for detecting the authentication form and automatically fill the form after decrypting the site password from the mobile application.

The communication between the Chrome application and the chrome extension is implemented through Chrome External Messaging². Specifically, this extension specifies the Application ID, which is a unique identifier for the Chrome application. After Chrome extension delivers the data to the application that is binded to the ID and has a pre-added listener, the listener can extract the data. The communication from Chrome application to Chrome extension works similarly.

Our prototype implements the BluePass client on the Chrome platform to simplify the communications among different modules; however, the framework of BluePass can be widely deployed on more platforms as long as both the computer and the mobile device have Bluetooth support and the browser extension is able to communicate with local applications on the computer. First, Bluetooth has become a standard device on modern computers and smartphones. Second, communication between browser extensions and native applications has been supported by most modern browsers, including Internet Explorer, Chrome, Firefox, Safari, Opera, etc.

3.5.3 BluePass Mobile Application

The BluePass mobile application starts a BluePass service, which runs in the background of Android and has a dedicated thread to listen to the incoming Bluetooth connection, which helps transparently authenticate a user to a registered website. The BluePass service inherits from the *Service* class in Android and keeps running until the user explicitly stops the service.

BluePass mobile application has a simple and clear user interface, which shows the status of the background BluePass service, either “running” or “suspended”. The user can easily change the service status by clicking “Start BluePass Service” or “Stop BluePass Service” buttons. When the service status is running, the Bluetooth listener starts listening and remains active even the mobile device turns off the screen and goes to sleep. Whenever users would like to stop the service, they just need to open the application and click the “Stop BluePass Service” button.

We use RFCOMM Bluetooth protocol to establish communication between the mobile phone and the computer, since RFCOMM is widely supported and provides public APIs in most mod-

²<https://developer.chrome.com/extensions/messaging>

Table 3.3: BluePass Scheme Evaluation

Scheme	Usability								Deployability						Security											
	Memorywise-Effortless	Scalable-for-Users	Nothing-to-Carry	Physically-Effortless	Easy-to-Learn	Efficient-to-Use	Infrequent-Errors	Easy-Recovery-from-Loss	Accessible	Negligible-Cost-per-User	Server-Compatible	Browser-Compatible	Mature	Non-Proprietary	Resilient-to-Physical-Observation	Resilient-to-Targeted-Impersonation	Resilient-to-Throttled-Guessing	Resilient-to-Unthrottled-Guessing	Resilient-to-Internal-Observation	Resilient-to-Leaks-from-Other-Verifiers	Resilient-to-Phishing	Resilient-to-Theft	No-Trusted-Third-Party	Requiring-Explicit-Consent	Unlinkable	
Password			●		●	●	○	●	●	●	●	●	●	●		○										
Firefox (with MP)	○	●	○	○	●	●	●		●	●	●		●	●	○	○										
LastPass	○	●	○	○	●	●	○	●	○	○	●		●	●	○	○	○		○		●	●	●	●	●	●
Tapas	○	●	○	○	●	●	●		○	●	●			●	●	○					●	●	●	●	●	●
BluePass																										
Sound-Proof			○		●	●	○	○	●	●		●	●	●	○		●	●	●	●	●	●	●	●	●	●

● indicates that the scheme fully carry the characteristic and ○ indicates that the scheme partially carry the characteristic (the Quasi prefix). We take rows 1-3 from [21], row 4 from [85], and row 6 from [88].

ern operating systems. Android supports two modes of RFCOMM connections, *secure mode* and *insecure mode*. The secure mode requires successful pairing before any RFCOMM channel can be established while the insecure mode allows connection without pairing two devices. Secure mode RFCOMM adds another layer of encryption. However, as BluePass communication is secured by (K_1, K_2) so that it does not rely on Bluetooth security. While insecure mode may fit better since it saves a pairing step from the user, Chrome application does not support insecure RFCOMM communication due to security concern. Therefore, we use the secure RFCOMM connection mode. Consequently, in the registration phase, the user also needs to pair the mobile phone and the computer first if they have never been paired before. Note that pairing only needs to be done once in a computer unless the user manually deletes paired devices on the mobile phone or computer.

3.6 Evaluation

3.6.1 Comparative Evaluation Framework

We use the comparative authentication scheme evaluation framework [21] to compare BluePass with other related authentication schemes. The results are summarized in Table 3.3. We can see that BluePass is *physically-effortless* since the entire authentication process is transparent to the user and *Quasi-Nothing-to-Carry* since users still need to carry their mobile phones though

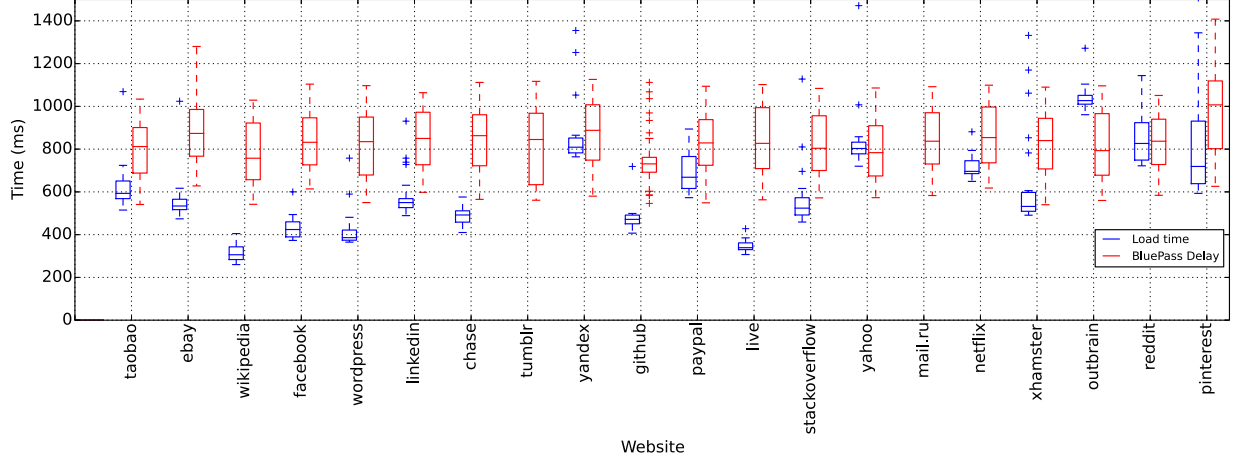
they carry them anyway. BluePass is accessible since it does not require the cellphone to have signal or cellular data. BluePass is *Quasi-Resilient-to-Throttled-Guessing* and *Quasi-Resilient-to-Unthrottled-Guessing*. Although BluePass itself does not enhance the security of the underlying password mechanisms, it can help defend throttled and unthrottled guessing by generating long random passwords for users and motivating users to use more secure passwords since they do not need to remember the passwords.

Bonneau et al. [21] points out that the framework does not describe all possible properties of an authentication scheme. Besides these factors, BluePass also keeps a simple and clean user mental model, which is highly suggested since wrong mental models easily make user passwords weaker [31]. Furthermore, BluePass strengthen usability by not requiring users to delete their password traces after use on a untrusted computer as other password manager (e.g., log out master account or delete local password vault).

3.6.2 Password Auto-fill Latency

For a usable password manager, the time required to fill the password field should be short. We record the delay between the time that the password input form is detected and the time that the form is automatically filled (denoted as T_{bp}). Since the delays on different websites may be different due to the specific website design, we choose 20 major providers from Alexa Top 100 website [107]. For each site, we make up a username/password pair and test the pair of credentials for at least 50 times. The password of each site is a randomly generated 16 byte string composed of all 4 characters types (Uppercase character, lower case character, digit, and special character).

Besides the Bluetooth communication latency, we also measure the loading time (denoted as T_{load}) for a website since page rendering (bottleneck to load a page) and Bluetooth communication tasks are running in parallel, indicating that the actual latency a user is experiencing is roughly $T_{bp} - T_{load}$, which is the time difference between BluePass running time and page loading time. T_{load} is measured by injecting a piece of javascript code, which measures the time when all javascripts on the webpage that need to run immediately are being executed subtracting the time that the browser is ready to send the HTTP request.

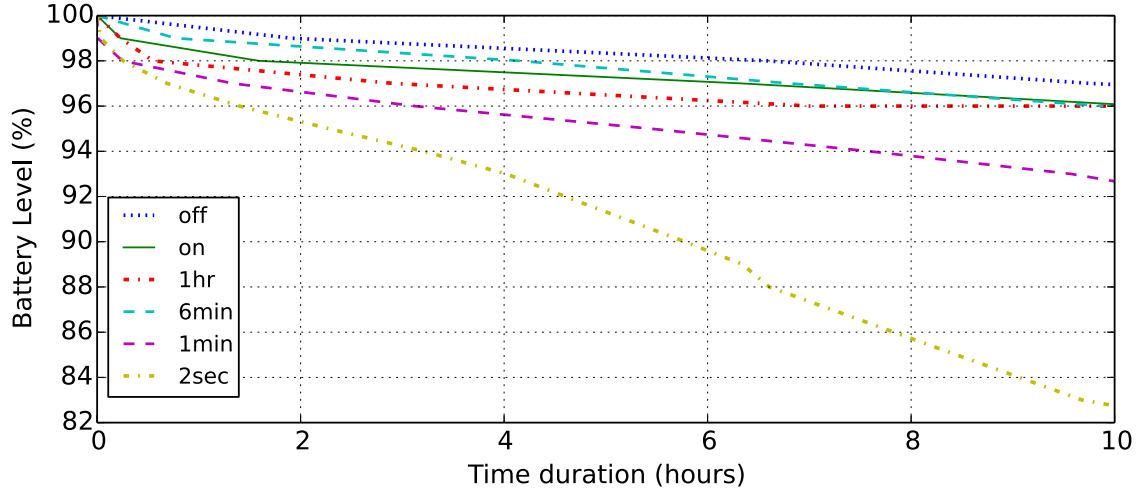
Figure 3.3: BluePass Latency**Table 3.4:** Delay Statistics

	Median	Mean	SD	Skewness
T_{bp}	778.0	814.6	158.3	1.6
T_{load}	599.5	837.6	691.3	2.6
T_{bp} (removed)	775.0	812.5	155.2	1.6
T_{load} (removed)	570.0	631.4	259.8	2.6

The results for all 20 sites are shown in Figure 3.3. Figure 3.3 does not show the T_{load} results for two web services, Tumblr and mail.ru, that have much higher T_{load} (averaged 2,700-2,800 ms). Generally the BluePass delay time (T_{bp}) is slightly higher than the page loading time (T_{load}). To illustrate the extent of the time gap, we show statistical analysis in Table 3.4. In the last two rows, we do not include Tumblr and mail.ru in our analysis since they have significantly higher T_{load} that are not representative for normal cases. With the two sites excluded, the average T_{bp} is 814.6 ms , which is short enough to be acceptable by most users. Furthermore, the actual delay that a user experiences is $T_{bp} - T_{load}$, which is only 181.1 ms in average. The standard deviation for T_{load} is higher than T_{bp} . The loading time T_{load} could be different under various factors, such as network condition, website implementation, etc. Since T_{load} highly depends on the website implementation, heavy javascript use in a site could largely contribute to a high T_{load} .

On contrast, T_{bp} is relatively stable since Bluetooth communication and mobile device computing are almost the same in each login attempt. Since the delay caused by BluePass is bounded by $T_{bp} - T_{load}$, BluePass imposes a very low latency on the password auto-filling process. According to our user study, users can hardly notice the latency.

Figure 3.4: BluePass Power Consumption



3.6.3 Power Consumption

One major concern of BluePass usage is the power consumption overhead on the mobile device, since BluePass requires the mobile device serve as a Bluetooth server that keeps listening to incoming connections. We measure the extra power consumption imposed by BluePass through monitoring the power levels of the mobile device when running BluePass password retrieval process in different frequencies. For comparison, we also record the power level of the device when Bluetooth is turned off (we call it a clean state).

To monitor the current battery level of the mobile device, we register a broadcast receiver in a simple battery monitoring application on the mobile device to listen to battery level changing event, upon which the current battery level and the timestamp are recorded. We tune the login frequency in the browser side (by refreshing a webpage in different frequency) to evaluate different use cases.

Except for the login frequency and BluePass on/off status, we keep all other settings exactly the same, such as installed and running application on the device as well as the network status (e.g., Wifi connection is turned off). We use a Nexus 5 mobile phone for evaluation, which has 2100 mAh battery capacity. As it takes a long time to use up the battery that has been fully charged, we run each experiment for 10 hours before charging the phone and running the next experiment. Though the granularity of battery usage broadcasting is in percentage level that may not be highly accurate, it is sufficient to evaluate the power efficiency of BluePass in a 10-hour test period.

Figure 3.4 illustrates the battery level dynamics through time under different experiment setups. "On" means the Bluetooth is turned on and "off" means the Bluetooth is turned off. Other lines represent the Bluepass log-in frequency. A reasonable frequency of login attempted by a normal user should not exceed 100 times a day, which means that the login frequency should lie around 0-10 times per hour. With 10 logins per hour, the power consumption is only 1% more than a clean state. We believe it is an unnoticeable overhead for users, given that almost 90% of users charge their phone more frequently than once per 2 days [90]. Besides, we can see that a significant power overhead is only incurred when the user tries to log in very frequently (17% when trying to log in every 2 seconds). However, normal users would not try logging in at such a high frequency.

In our experiments, the mobile phone is in a state that does not receive cellular or wifi signal, so the battery drains very slowly. When the mobile phone is in normal daily usage, the battery usage becomes much higher. However, the BluePass power consumption remains the level of 1% of total power with 10 hours use.

3.7 User Study

To verify how real users rate the security and usability of BluePass, we conduct a user study to gather feedback and comments from normal users. Upon approval of IRB of our institution, we recruit 31 volunteers to use and comment on BluePass. The volunteers include 16 males and 15 females. As the study is only in a school scale, most of them age 20 - 30 years old. Besides, most of them have a bachelor degree. In order to spread our study of different computer expertise, we deliberately recruit volunteers from 10 fields of study.

We ask each of the volunteer to finish a series of tasks. They are (1) register to BluePass server and configure BluePass, (2) create a new account in our self-deployed test site, (3) log in the test site (Migrate password), (4) try using BluePass to log in again (Log in from a primary computer), (5) change the current password and try using BluePass to log in (Change Password), (6) configure BluePass in another computer and log in (Log in from another computer), (7) turn off BluePass and try logging in, and (8) turn on BluePass and try logging in. We also create a test website that has only login and changing password functions for the volunteers to operate on.

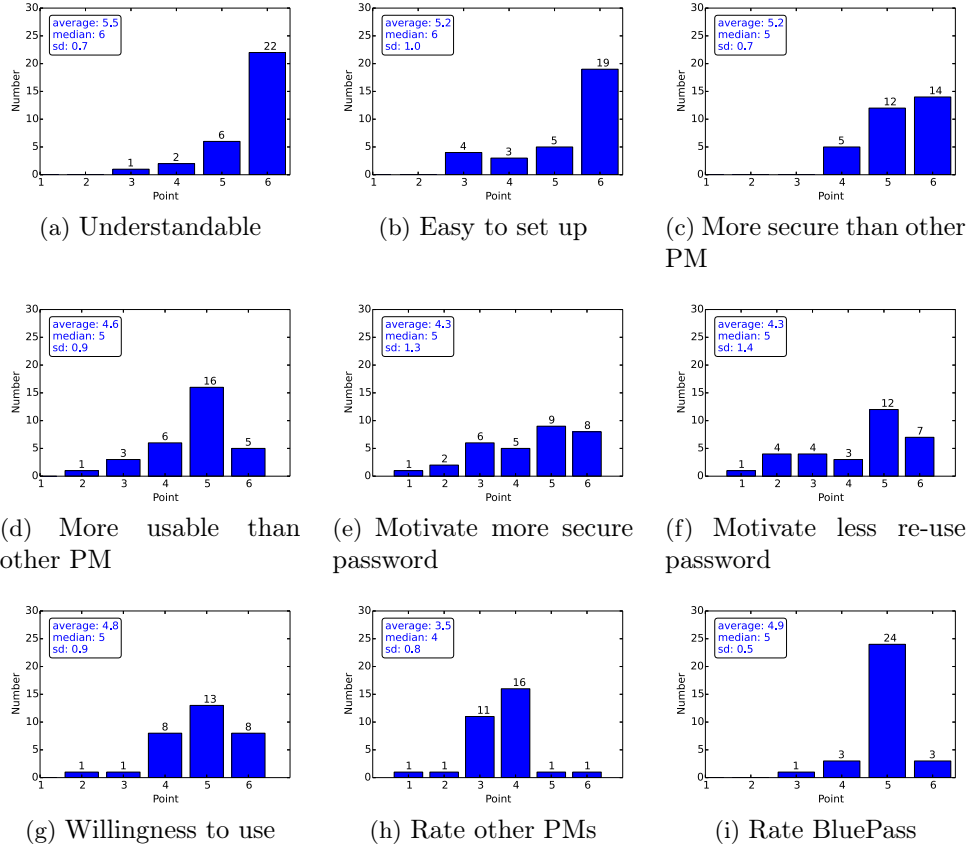


Figure 3.5: Survey Results

After finishing the tasks, the testers take a post-study questionnaire. The questionnaire mainly uses 6-point scale rating where 1 point means strongly disagree and 6 point means strongly agree. The results are shown in Figure 3.5. Testers generally think the concept of BluePass is understandable and it is fairly easy to set up. 87% of testers (27 out of 31) agree that BluePass is more usable than any other password manager they have used before.

BluePass motivates the testers to increase password security. More than 70% (22 out of 31) of the testers state they are motivated to choose more secure passwords and less likely to re-use existing passwords, thus making their passwords stronger. However, though the testers report they are motivated to use more secure passwords, we notice that only 4 testers have tried using random passwords generated by BluePass to create/change their passwords, which may result from the fact that users feel “unsafe” to use a non-memorable password.

The majority of testers (94%) expresses willingness to use BluePass to manager their passwords.

We also ask the testers to compare BluePass to other favorite password managers they have used, and testers show large preference to BluePass over existing password managers. To summarize, BluePass is generally considered more secure and usable than existing password managers by the testers. Most of them show preference to BluePass and willingness to use it. Thus, it is reasonable to conclude that BluePass does help users secure their passwords.

3.8 Discussion

3.8.1 RSA Key Pair

BluePass can use only one RSA key pair (K_1, K_2) to achieve bi-directional communication between the mobile phone and the computer. We must guarantee that the compromise of K_1 will not lead to the compromise of K_2 , and vice versa. We know that all public key cryptography algorithms ensure that it is hard to derive the private key from the public key, but not vice versa. For instance, given an ECC private key, it is easy to derive the ECC public key, since $public_key = private_key * G$. However, for RSA, in theory, it is hard to derive either e or d from knowing the other one. Therefore, we can use only one pair of RSA keys with careful parameter settings.

There are two minor things to notice in the detailed RSA implementation. First, in practice e is usually chosen a small/fixed number, but this should be avoided. Second, RSA private keys are often stored in their "Chinese Remainder Theorem" form, which includes the two secret numbers often denoted p and q , from which the totient is computed. With totient and the private exponent, the public exponent is quickly computed. Therefore, BluePass cannot use the Chinese Remainder Theorem to speed up the calculation.

3.8.2 BluePass Limitations

BluePass has several limitations. First, a user has to carry a powered-on mobile phone to make BluePass work; otherwise, BluePass falls back to conventional ways that users remember and input passwords. Second, BluePass cannot work well when the mobile device or the computer does not support Bluetooth communication. In those cases, the hand-free benefit cannot be offered by

BluePass. Instead, the users have to use their phones to display their site passwords after inputting their master passwords.

3.9 Conclusion

This chapter introduces a hand-free password manager called BluePass for achieving both strong security and high usability. BluePass attains the security level of two-factor authentication by storing password vaults in a mobile device and the decryption key in the user computer separately. Exploiting the automatic bluetooth communication between the mobile device and the computer, BluePass enables a hand-free password retrieval process for users. BluePass also places the decryption keys to remote servers to support password portability while de-centralizing the storage of password vaults to prevent a single point of failure. We implement a BluePass prototype on Android and Google Chrome platforms. Through system evaluation, we show that the password retrieval latency a user experiences is less than 200 milliseconds on average, and BluePass only consumes a negligible 1% battery power with 10 hours normal use on a mobile device. Through a user study comprising of 31 testers, we demonstrate that BluePass does motivate users to choose stronger passwords and less likely to reuse existing passwords.

Chapter 4

Email as a Master Key: Analyzing Account Recovery in the Wild

4.1 Introduction

Text-based passwords have been used as a dominating solution of user authentication for many decades [86], due to their favorable usability and the fact that they cannot be entirely replaced by other authentication approaches in the foreseeable future [21, 22]. Since text-based passwords are vulnerable to cracking and theft attacks [87, 121], significant research efforts have been made toward enhancing password security from different aspects, including measurement [67, 120], password policy [18], password meters [38, 43], and password managers [77].

Whereas it is critical to secure a password at its creation and input procedures, account recovery as an important component in the entire framework of password-based authentication has been largely overlooked. Account recovery is an irreplaceable link in the password authentication chain. Not being able to provide an easy way to recover the password can cause user frustration, human labor waste, or even user loss. Meanwhile, the account recovery process should also be carefully designed to avoid backdoor threats. Today, most websites rely on accessibility of a registered email of a user to recover or reset forgotten passwords. Though email-based recovery is deployable, compatible, and easy to use, its security implication is understudied. A compromised email account could inevitably become a single-point-of-failure, since an attacker can easily reset the passwords of a

victim’s other online accounts. Note that such an account recovery attack can naturally circumvent security enhancements on passwords and directly compromise a large number of user accounts by resetting their passwords.

A simple and effective idea is to keep the email account safe. However, this does not happen in a practical world. There is a large number of email accounts leaking to malicious attackers. For example, it was suggested by a security firm in May 2016 [4] that more than 200 million email username/password combinations are in possession of hackers. Major email service providers including Gmail, Hotmail, Yahoo, and Mail.ru are all affected, and millions of email account credentials are compromised. Thus, it is important to understand the security implications of email-based account recovery. A systematic study on its vulnerability, potential damage, and defense has yet to be conducted.

In this project, we first quantitatively measure the vulnerability of most websites to an account recovery attack. In particular, we manually investigate the account recovery protocols and authentication schemes adopted by the Alexa top 500 websites. We observe that 92.5% of the web services we examined rely on emails to reset user passwords, and in 81.1% of websites, their user accounts can be compromised by solely accessing the registered emails. The difference of 11.4% is due to the lack of username knowledge (i.e., the username/password credential is incomplete) or classifier-based authentication, where abnormal login attempts will be blocked. Afterward, we demonstrate the damage that can be caused by password resets through case studies on four categories of websites, in which we show that significant privacy and financial losses are possible to incur. Then, we examine security policies of eight major email providers. We conclude that a significant portion of leading email service providers fail to take deserved effort to provide user email account protection, leaving them vulnerable to a variety of attack vectors.

Finally, we propose an account recovery protocol named Secure Email Account Recovery (SEAR) as a preliminary solution to address the single-point-of-failure problem of user email accounts. Specifically, the email provider adds an extra layer of protection, which can be in the form of an SMS authentication when a password reset email is intended to be opened. Thereby, the attacker cannot spread the attack by compromising an email account. We demonstrate that SEAR can be

easily implemented under the current network infrastructure with full backward-compatibility, and it can strengthen account security with an all-rounded usability consideration (i.e., similar user experience, no need for providing the phone number to all websites that one intended to protect, etc.).

Overall, the major contributions of this work are summarized as follows.

1. We identify the de facto account recovery protocols in the wild by examining the Alexa top 500 websites. In the measurement study, we build taxonomies on websites and account recovery credentials, which enable us to explore the account recovery problem from different perspectives and dimensions.
2. We systematically investigate the email-based account recovery vulnerability that widely exists in today’s web services. Our assessment reveals that the security risk is high and could cause severe damage to users.
3. We propose SEAR as a preliminary solution that can be seamlessly integrated into modern email infrastructures in a fully backward-compatible manner. The prototype of SEAR is implemented on open-source mail servers.

4.2 Terminology and Definitions

With the development of new schemes and years of advances in multiple dimensions, the account recovery process cannot be easily elaborated. In order to help understand and organize the heterogeneous makeup of the account recovery process, we first perform a classification on account recovery credentials and websites.

4.2.1 Recovery Primitive, Method, and Protocol

Account recovery is essentially another authentication process, which needs one or multiple legitimacy validations. Each validation is usually done on the server side by matching a mutually agreed-upon piece of credential ε to the one supplied by the login attempter. While an ε can be

represented by a series of symbols, we categorize ε into six types based on their sources, as listed below, and we call each type a *recovery primitive* (γ).

- **Email** (γ_{em}). Email primitive is the accessibility to a registered email. The validation process may be of various manners, such as sending a hyperlink to reset a password, sending a one-time code for inputting a password reset form, or even directly sending back the original password. Nevertheless, accessibility to the registered email is the only prerequisite.
- **Phone** (γ_{ph}). Similar to email, phone primitive demands accessibility to a phone that is associated with a pre-registered phone number. The website may choose to call the phone number or send a text message.
- **Security question** (γ_{sq}). Security question is a kind of knowledge-based authentication, which allows a password reset if questions are answered correctly. Normally, the answers to security questions are intrinsic to users, and hence no extra memory burden is introduced. An example is, "What is your favorite food?"
- **Private information** (γ_{pi}). Private Information is also knowledge-based authentication in a personally identifiable and thus not massively predictable sense, the answer to which is relatively unique among different users. Although users may or may not intrinsically remember it, they usually have access to the information from other channels. Examples of such information include a credit card number and Social Security Number.
- **Activity Information** (γ_{ai}). Activity information involves account activity traces. Some service providers believe that a user is expected to be able to recall some of the most basic activities of its account, such as the nickname/username, most login locations, and other users with whom they usually interact. It may even require assistance from acquaintances on the same website.
- **Recovery Token** (γ_{rt}). A recovery token is usually a non-memorizable piece of information that users possess. Examples are randomly generated tokens at registration or one-time codes generated by mobile applications (authenticators or website-designated apps).

In some cases, websites may ask for a combination of multiple γ for stronger authentication and provide multiple such combinations for users to choose from for increased flexibility. To set boundaries among these similar concepts, we define one way to recover a password as a *recovery method*. Fundamentally, a recovery method could consist of one or multiple recovery primitives, and all primitives should be supplied by a user correctly in order to recover its account. For example, on a website ω , one way to recover a user password may be $m_{\omega,1} = \{\gamma_{em}, \gamma_{sq}\}$, meaning that the password can be reset by whomever is in possession of the registered email and answers to the security questions. A recovery method is the most basic unit of a successful account recovery. Similarly, we define the set of all m that a website provides as the *recovery protocol* (p) of the website. For instance, for the website ω , its recovery protocol is $p_{\omega} = \{m_{\omega,1}, m_{\omega,2}, \dots, m_{\omega,i}\}$, indicating that there are i recovery methods and that any recovery method can be used alone to successfully recover an account.

4.2.2 Website Classification

We categorize websites into several groups, helping us look deeper into how different websites handle account recovery in a finer granularity, as well as conduct the damage assessment. Grosse and Upadhyay [51] have done a user account classification based on the values of the accounts. However, their classification is user-oriented, which heavily relies on user-subjective perspective and activity. Namely, different users may have different types of accounts on the same website, depending on the user's purpose for using the websites. By contrast, we take a website-oriented approach by classifying websites based on their service nature. We define the following six website groups with some terminologies acquired from [51].

1. **Routine.** A routine website is one in which users passively receive information. Most of its users produce zero or little long-residing content. Examples of routine accounts are online newspapers, those used for online education, and gaming or music websites.
2. **Spokesman.** Spokesman website accounts usually represent a user's opinion or identity. Users rely on spokesman websites to deliver and exchange information with other real users.

Examples of spokesman websites are online social networks, such as Facebook, Yelp, and LinkedIn.

3. **E-commerce.** E-commerce websites mainly involve trading. A business website could be an online retailer, such as Amazon and Ebay, or paid service providers, such as insurance companies. It is common to find addresses, shopping histories, phone numbers, and even payment information in user accounts on these websites.
4. **Financial.** A financial website usually concentrates on financial activities, such as deposits, withdrawals, and online transactions. Examples of financial websites are banking, brokerage, or wallet-type websites, such as Paypal.
5. **Tool.** A tool website does not usually produce a final product. Instead, it provides a tool or platform for helping build or shape the final product. Examples of tool websites are search engines, website builders, online graph drawers, and web traffic analyzers.
6. **Email.** Email websites provide online accounts that are associated with user email addresses, which can send and receive emails, such as Gmail or Outlook.

Nowadays, it is common for websites to have a heterogeneous service nature. It is sometimes hard to classify a website into a single type. For example, Google is a tool website since it offers a search engine. Meanwhile, it is also a spokesman website (Google+) and an email website (Gmail). As such, we sometimes classify a website as multiple types. While allowing such cases, we primarily categorize a website based on its main services and user recognition. For example, an online newspaper may have a review section under an article where users can express and discuss their opinions. However, most users may only browse the news without writing any comment. Thus, the online newspaper is categorized solely as a routine website, instead of a spokesman website.

4.3 Account Recovery in the Wild

We manually investigate the account recovery protocols adopted by the Alexa top 500 websites to help understand the protocol composition of modern websites. Since the top 500 websites are

ranked by their global web traffic, each of them has a large number of users (or visitors), and thus reflects the de facto techniques adopted for account recovery.

4.3.1 Demographics

Within the 500 most traffic-heavy websites, we identify 245 websites in which we are able to create an account. Among them, 239 (97.5%) websites have enabled an account recovery protocol (p). Since we are only interested in recovery protocols, we consider our dataset to contain only the 239 websites thereafter. There are fewer protocols than websites due to multiple reasons. First, we count the same protocols that share the same database only once, such as all regional Google sites and subsidiaries of Google, like Youtube. This type includes 99 websites. Google alone contributes 55 of them. Second, there are 40 websites that do not have login functionality. For example, some online newsletters do not need user logins. In addition, some recorded sites are just advertisement network referrer links or content delivery networks in which not even an accessible homepage is available. Examples are adnetworkperformance.com and www.t.co. Third, we fail to examine 51 websites with less commonly used languages. It is challenging to recognize and input CAPTCHA in these languages, which is a required process in order to register an account. Finally, on the rest of the websites, a local phone number or membership is mandatory for registration. Examples include most online banking systems. These websites are not open to an outsider, and thus we are unable to access them.

The websites being successfully examined bear a similar distribution on visitor origins in the Alexa top 500 list. Though only a limited number of websites are examined, these popular websites attract most web traffic. For instance, Google alone is reported to account for up to 40% of web traffic [3]. Therefore, we believe that our analysis is representative and can genuinely cover the mainstream of modern website account recovery protocols used by most online users.

Overall, our dataset contains 239 websites that enable account recovery, naturally including 239 password protocols. In these protocols, we identify 324 recovery methods. Then we identify 364 recovery primitives in these recovery methods. On average a website has 1.36 recovery methods, and each method involves 1.12 recovery primitives. This implies that most of the websites provide

Table 4.1: Recovery Primitive Distribution

Primitive	Number	Percentage	Self-sufficient	Percentage
Email	232	97.1%	213	89.1%
Phone	46	19.3%	40	16.7%
Security Question	22	9.2%	11	4.6%
Private Information	7	2.9%	0	0.0%
Activity Information	12	5.0%	10	4.2%
Recovery Token	3	1.3%	3	1.3%

“Self-sufficient” implies that the recovery primitive is the sole ingredient in a recovery method (i.e., $|m| = 1$, for example, $m = \{\gamma_{em}\}$).

only one recovery method, and recovery primitives in a recovery method are mostly homogeneous.

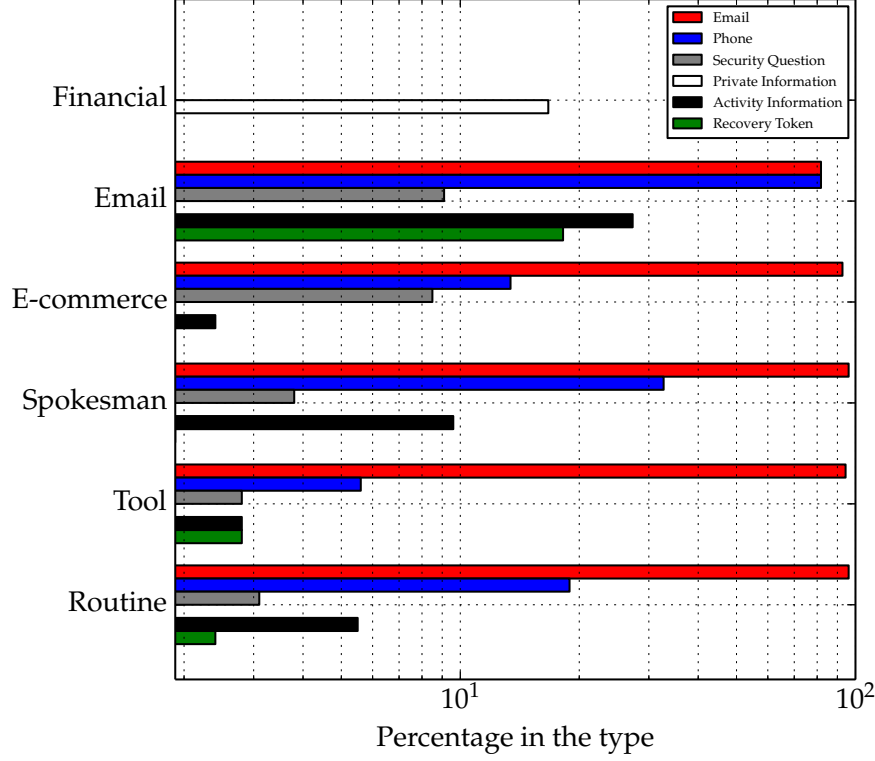
Note that nowadays, many websites have used Single Sign On (SSO) for logging in. SSO enables a user to use the account of an identity provider, such as Facebook, to log into other websites. In our dataset, 136 websites feature at least one SSO identity provider. The top three are Facebook (103 occurrences), Google (67 occurrences), and Twitter (35 occurrences). Regarding SSO, users should recover their accounts from the SSO identity provider website, such as Facebook.

4.3.2 Primitive and Method Usage

To illustrate the major composition of recovery protocols, we examine the usage of recovery primitives and list the overall occurrence of each primitive in Table 4.1. As shown in the table, using email to recover a password is prevailing: 97.1% of websites include email (γ_{em}) in their recovery protocols. Furthermore, among 89.1% of websites, email itself is sufficient to recover a password (namely, at least one of their recovery methods contains the element of email primitive only). It is evident that most of the top websites delegate the security responsibility of account recovery to email service providers, instead of extending and relying on their own security infrastructures.

The second most popular method is using a mobile phone, which is seen in a notable portion (19.3%) of websites. Surprisingly, 4.6% of websites still rely exclusively on security questions to recover passwords, which are suggested against by many previous researches [20, 46, 99]. Meanwhile, private information, activity information, and recovery tokens are much less used since they may involve more deployment costs and have privacy concerns. However, these primitives are commonly used in sensitive online services, such as financial institutes.

Figure 4.1: Recovery Methods – Single-Primitive



Log scale is used for a clearer presentation of small percentages. The sum of percentage can be more than 100 (with multiple recovery methods in a recovery protocol), or less than 100 (we show only single-primitive recovery methods)

From Table 4.1, we can also easily infer that most recovery methods contain only one recovery primitive. In fact, recovery protocols in 95% of the websites we examined include at least a single-primitive recovery method. As recovery methods with multiple recovery primitives are rarely found, scattered, and hardly organizable, we focus more on unveiling the structure of a single-primitive recovery method. Following the annotations introduced in Section 4.2.2, we identify 127 routine websites, 82 e-commerce websites, 52 spokesman websites, 36 tool websites, 6 financial websites, and 11 email websites. Their single-primitive recovery methods are portrayed in Figure 4.1. It is not surprising to see that different genres of websites use different single-primitive recovery methods to balance their own security and usability trade-off.

From the figure, we can also see that financial websites are quite different from the other five — only one website uses a single-primitive method for account recovery (private information). Financial websites are the only type of website that usually has multiple recovery primitives in a

recovery method. The other five types of websites all heavily rely on email (more than 80% for email sites and more than 90% for the other four) for account recovery. Using a phone follows as the second most commonly seen account recovery method. Interestingly, email websites themselves heavily rely on a mobile phone to recover passwords and are significantly more prone to use account recovery primitives other than email services. One possible explanation is that the email is already the end point of an account recovery chain and that email service providers prefer not to lead their users to their competitors' email services. Thus, they attempt to offer other remedies, such as a mobile phone.

The usage of multiple-primitive recovery methods is not very common, given the fact that on average a recovery method consists only 1.12 recovery primitives. Due to the sparsity, we list the total number of two-primitive and three-primitive methods used in each website category in Table 4.2. None of recovery methods we identified has more than three primitives. Financial and e-commerce websites deploy such multi-primitive recovery methods more than other websites, possibly due to their critical service nature. However, with equal importance, email websites do not have a single recovery method that has more than one primitive.

Table 4.2: Recovery Methods with Multiple Primitives

Primitive Number	Routine	Tool	Spokesman	E-commerce	Email	Financial
2 primitives	6	3	3	8	0	6
3 primitives	0	1	0	1	0	5

Overall, it can be inferred that most of the top websites delegate the security responsibility of account recovery to email service providers, instead of extending their own security infrastructures. There could be several reasons for this. First, it keeps high usability, as the registered email address becomes a centralized master key through which users can conveniently manage almost all of their online accounts. In other words, users incur almost no more cost when the number of online accounts increases. Second, email recovery mechanisms can be easily deployed at both sever and client sides. Third, the security obligation of account recovery is delegated to other online services, which significantly reduces the security responsibility of the website. Accordingly, the email of a user ends up as essentially another password manager and thus a potential single point of failure. The illegal access to an email account can pose a serious security threat on most websites, with

the only exception being financial websites. As a result, an attacker can easily compromise user accounts by mounting an account recovery attack, especially nowadays when email accounts are at a massive loss [4].

4.4 Attack Assessment

Since emails play a critical role in account recovery, it is necessary to evaluate the vulnerability that may be introduced by emails, especially when user email accounts are at risk.

Although we have observed that most websites rely on emails to recover user passwords, the assumption that possessing a password will compromise a user account may no longer hold under today’s multi-dimensional authentication context, where a password may not always be the sole gatekeeper. In fact, an account recovery attack should be considered successful only if an attacker can actually log in to the target website and impersonate the victim user, which rules out cases where the attacker steals a password but fails to access the account due to a lack of other credentials. We first define the capabilities of an attacker and then discuss the possibility of a successful attack.

4.4.1 Threat Model

We assume that an attacker has access to the victim’s primary email account and attempts to log in to a user account by exploiting the information included in the recovery email. Specifically, the attacker has no knowledge about the victim’s personal information and makes no attempt to obtain the information that is believed discoverable or guessable yet hardly quantitatively assessable, such as user-chosen usernames (when it is neither the email address nor included in the recovery email) or security question answers. We also assume that the attacker does not make extra efforts to bypass additional classifier-based authentication schemes, such as IP address or OS/browser fingerprinting. Note that we aim to set a baseline for the success rate of an account recovery attack so that we keep our attack model simple and clean. In the real world, attackers may try to use more sophisticated tactics to break into even more user accounts [10].

4.4.2 Possibility to Break-in

As suggested by Table 4.1, 213 out of the 239 websites solely rely on emails to recover user passwords, making 89.12% of the examined websites potentially vulnerable to account recovery attacks. However, an attack may not be successful for two reasons: the lack of other credentials and additional classifier-based authentication. Thus, these factors should also be taken into consideration for estimating the success rate of mounting account recovery attacks on these 213 websites. We discuss the impact of each factor as follows.

4.4.2.1 Lack of Credentials

The first factor is the lack of other credentials, and a user’s password is not the only credential needed to log in to a system. We investigate the use of a username, as it is also a required piece of information for successful authentication. A website needs to know a username or an email address first to locate an account so that the corresponding recovery methods, such as security questions for the designated user account, can be retrieved. We identify that 80.3% (192 out of 239) of websites allow the use of email addresses as usernames, and 178 of them can use emails to recover their passwords. On the other hand, 23 websites that do not treat email address as a type of username (i.e., a username is freely selected by its user) provide email-based username recovery or directly send a username in the account recovery email, meaning that the username itself can be accessible from the email account. Thus, in total, 84.1% (201 out of 239) of the examined websites are potentially vulnerable to account recovery attacks. In other words, among the 213 websites that allow emails to recover passwords, 12 of them are immune to account recovery attacks because attackers cannot know usernames through emails. Another lack-of-credential scenario is the two-factor authentication (2FA) in which an attacker has no access to the other authentication factor. In this case, the login will also fail. We found that 35 websites feature 2FA options. However, the general adoption rate is still believed to be quite low. By analyzing more than 100,000 Google accounts, Petsas et al. [93] estimated that Google 2FA is adopted by no more than 6.4% of its users in 2015. It is also unlikely for other websites to have a much higher adoption rate than Google. Furthermore, 2FA is an option disabled by default in all of the websites we have identified. Therefore, a 2FA-available website

Table 4.3: Websites Vulnerable to Account Recovery Attacks

All	R_1	$R_1 \& R_2$	$R_1 \& R_2 \& R_3$
239	213 (89.12%)	201 (84.1%)	194(81.1%)

R_1 : Allows email as an account recovery method.

R_2 : Username is directly obtainable. R_3 : No classifier is enabled.

should still be considered vulnerable to account recovery attacks since more than 90% of the users are not really protected.

4.4.2.2 Classification-based Authentication

The other factor taken into account is classification-based authentication. Leveraging more or de-centralized credentials may incur significant usability hassles and thus repel users. Therefore, many websites start to use a classifier to automatically verify a legitimate login attempt to balance usability with security, where a correct password is not sufficient for login. The classifier aims to detect anomalous login behaviors by taking many signals into the classification decision, such as the IP address, cookies, and OS/browser fingerprints. Alaca et al. [10] identified and evaluated 29 fingerprinting mechanisms, and each of them may produce multiple signals. If a login attempt is classified as suspicious, the system is likely to trigger a standard 2FA. Authentication classification is reported by Google [8] to effectively reduce 99.7% of account compromises using more than 120 features. However, the classification is a black-box that is hard to comprehend, especially when the features are numerous. To determine whether a website has enabled a classifier, we adopt an attacker-centric approach, where we probe all 239 websites by using the Tor network and VPN¹, which enables us to emulate an attacker. Specifically, we first train each website by manually logging in to the website on the same computer once per day for a period of one week. The computer has a fixed fingerprint and IP address. Then, we camouflage ourselves as a user in a different country with different operating systems and browsers (all cookies cleared) to log in to the same website three weeks after the training stage. Note that we have provided necessary information, which includes a backup email, phone, and security question, for the use of the 2FA to the website when the classifier has low confidence. Our methodology cannot guarantee 100% accuracy of the results

¹The Tor network is known for having abnormal login issues in some websites, so we use both Tor and VPN to obtain most accurate information.

since the classification systems of these websites are still unknown. However, we believe that our results are sufficiently close to the ground truth since a useful classifier should capture such obvious anomalies. Our results indicate that only 14 (5.9%) out of the 239 websites are using a classifier, as we are either required to complete a standard 2FA or blocked from logging in. Furthermore, 8 of the 14 websites rank top 30 in web traffic, and the others are mainly financial websites. Clearly, though useful, classification-based authentication has not been widely used, and thus account recovery vulnerabilities still remain, at least at the current stage.

After considering the above two factors, we are able to answer the question of how many websites are vulnerable under such an attack model. We concisely summarize the results in Table 4.3, which shows that overall, 81.1% of the websites we examined are vulnerable under our threat model. In addition, if an attacker is sophisticated and could emulate enough login signals to deceive the classifier, 84.1% of the websites would be vulnerable to account recovery attacks.

4.5 Damage Estimation and Email Security

As a large portion of websites are vulnerable to account recovery attacks when a registered email is compromised, we evaluate possible damages that could be caused and the security policies of major email providers, which are essential to throttle attacks on user email accounts.

4.5.1 Damage

The damage can be multi-fold. First, attackers are able to steal private information, such as home address and activity history of users. In fact, this is the main reason why an attacker is interested in user passwords. Second, the attacker may also actively impersonate legitimate users to post information, such as sending spam messages on the user’s behalf [8]. Third, they may cause financial loss by purchasing products and stealing credit card or bank information. Measuring the extent of the damage can be complex and error-prone since even the same type of websites could have very different user data and security policies.

We estimate the possible losses by examining typical websites from four major website groups, which are routine, tool, spokesman, and e-commerce. We do not examine the email group as

Egelman et al. [42] have already done a thorough investigation on how much sensitive information resides in one’s primary email account, reporting that a substantial amount of sensitive information can be found in the email archive, such as credit card numbers (16%) and SSN (20%). We also exclude the financial group due to the fact that all of the financial websites we examined in the Alexa top 500 websites are immune to the account recovery attack, as the email is insufficient to reset a password. In our examination, we select those websites with a single service type. In addition, we also try to select these websites that are likely used by normal users. A counter-example is a paid advertisement publisher, which has a high volume of web traffic, but few normal users would use it.

Table 4.4: Damage Estimation

	Site	Sensitive Information	Activity	Financial
Routine	netflix.com	Phone Number, Watch History, Credit Card Number, Credit Card Info		Subscribe/Update Service
	nytimes.com	Name, Location, Purchase History, Occupation, Income, Gender		
	weather.com	Name, Birthday, Gender, Phone Number, Home Address, Work Address		
	wikia.com	Location, Birthday, Name, Gender, Occupation, Posts		
Tool	github.com	Company, Location, Credit Card Number, Credit Card Info	Sabotage	Purchase Service/Data
	dropbox.com	All Files Stored, Access History	Sabotage	
	skype.com	Phone Number, Birthday, Location, Connection’s Phone Number, Birthday, Location, Gender		
	ebates.com	Name, Address, Shopping History	Spamming	
	facebook.com	Name, Address, Birthday, Gender, Work, Education, Phone Number, Contact’s Information, Posts, Messages	Spamming, Sabotage	
Spokesman	instagram.com	Name, Phone, Gender	Spamming, Sabotage	
	reddit.com	Private Messages	Spamming, Sabotage	
	quora.com	Private Messages	Spamming	
	livejournal.com	Birthday, Location, Private Messages, School, Posts	Spamming	
	amazon.com	Shopping History, Files on Cloud, Name, Address, Phone Number, Private Messages, Reviews, Browsing History, Credit Card Number, Credit Card Info		Purchase Products/Services
E-commerce	ebay.com	Name, Gender, Address, Buying History, Selling History, Private Messages, Phone Number, Credit Card Number, Credit Card Info	Sabotage	Purchase Products*
	walmart.com	Name, Address, Phone, Shopping History, Credit Card Number, Credit Card Info		
	gap.com	Name, Gender, Address, Phone Number, Shopping History, Credit Card Number, Credit Card Info		

Red color indicates that the information is fully obtainable while Blue color indicates that the information is only partially obtainable.

* When purchasing a product on Ebay, the user can modify the shipping address without re-inputting payment information.

We show the damage assessment in Table 4.4. It is evident that all of the websites we examined, to various extents, expose user private information, such as phone numbers, birthdates, and addresses, to attackers. In addition to private information, an attacker is able to actively mount subsequent attacks, such as sabotaging or spamming. Sabotaging may not be appealing to the attacker since it does not bring many benefits. However, using a real account for spamming is a common practice among spammers [8]. Furthermore, attackers may even purchase products in online stores with stolen payment information. For the attacker to receive the ordered package or intercept the delivery process, it may need to change the shipping address. We observe that many e-commerce websites require payment authentication, in terms of the credit card security code (Walmart and GAP), to post an order. Amazon requires to re-input the complete payment information if the shipping address is new. However, surprisingly, Ebay allows a user to change the shipping address freely without additional authentication, which makes financial losses largely possible if the account

is compromised by attackers.

Table 4.5: Examining Major Email Providers

Provider	Region	Length	Composition	2FA	Classifier
Gmail.com	USA	8	1	✓	✓
Yahoo.com	USA	7-10*	4-1*	✓	✓
Outlook.com	USA	8	2	✓	✓
AOL.com	USA	8	1	✓	
QQ.com	China	6	1	✓	
163.com	China	6	1	✓	
Sina.com.cn	China	6	1		✓**
China.com	China	6	1		
China.com.cn	China	6	1		
Rediff.com	India	6	1		
Yandex.com	Russia	8	1	✓	

* Minimum password length and composition can vary depending on each other. For example, a password of length 7 must have 4 types of characters to be accepted by Yahoo. However, a password of length 10 can have only 1 type of character.

** The on/off of the classifier is configurable, and the default is off.

Table 4.6: Password Policies

	Routine	Spokesman	E-commerce	Financial	Tool	Email	Overall
Length	5.74	5.54	6.35	7.33	5.91	7.0	5.92
Composition	1.20	1.19	1.57	1.67	1.36	1.18	1.33

For Yahoo.com, we choose a minimum length of 8 and a composition of 2 since this setting may fit well with more typical passwords.

4.5.2 Assessing Email Security

Since email is pivotal to account recovery, the security of user accounts in a website is heavily dependent on the email security. A more secure email service can certainly help to thwart account recovery attacks in the first place.

To this end, we evaluate the security policies of all 11 major email service providers in our dataset, which span different geo-locations, including North America, Asia, and Europe. The fields examined involve several authentication policies, including minimum password length, minimum password composition (uppercase letters, lowercase letters, digits, and special characters), whether 2FA is provided, and whether a classifier is used to filter out abnormal login attempts. The list of providers we examined and results are shown in Table 4.5.

We also list the password policies of all six types of websites in Table 4.6, with respect to minimum password length and minimum types of characters required (on average). We can see that the minimum length of passwords in email websites is seven, which is only less than that of financial websites. However, email websites have the weakest composition complexity policy, since most of them do not require more than one type of character in a password, and users are more likely to create predictable passwords under such a policy.

We also notice that a significant portion of email providers include 2FA functionalities in their authentication systems. Compared to the overall rate of 2FA-enabled websites, email providers show a much higher security concerns and offer 2FA enhancement to secure user accounts. However, the number of users that actually use 2FA is likely to be small [93]. A more effective solution might be using a classifier to verify a legitimate authentication attempt. Although some of the classification signals can be easily spoofed [10], it is still difficult for an attacker to correctly spoof all signals considered by the classifier, especially when the adopted signals are unknown [22]. Unfortunately, only 4 out of the 11 email providers have integrated such a protection mechanism. One of them (sina.com.cn) requires a user to turn on the classifier, but most users probably do not enable it as the default setting is off. The other 7 email providers are much easier to be compromised by phishing attacks and password guessing/cracking attacks. Under such a condition, those accounts that are associated with a weak email account are vulnerable to account recovery attacks.

Generally speaking, a large portion of major email service providers fail to provide adequate security protection on user email accounts. It makes an account recovery attack more likely to happen, and thus jeopardizes the security of the online accounts that rely on emails for account recovery.

4.6 Securing Email-based Account Recovery

It is not an uncommon scenario that an email provider and its users fail to adequately protect their email accounts. Thus, an email account compromise could trigger massive compromises of other accounts that use emails for recovery. To mitigate account recovery attacks, we propose a lightweight Secure Email Account Recovery (SEAR) protocol that can be seamlessly integrated into

current network infrastructures. SEAR does not attempt to change the current email-based account recovery model, but instead, it aims to prevent attackers from recovering other account passwords if an email account is compromised. SEAR requires little effort from a website and its users. In short, SEAR requires a 2FA only when an account recovery email needs to be opened. Meanwhile, the normal email checking experience remains unchanged.

4.6.1 SEAR Specification

The core of SEAR is to add a header in an email to indicate that the email is for account recovery purposes. This method is transparent to existing email infrastructures since Simple Mail Transfer Protocol (SMTP) allows users to customize headers. The basic workflow of SEAR is simple. The account provider (i.e., the website where an account recovery is undergoing) will add a header “tag:value” pair (we choose “Recover:1”) in the recovery email, and upon receiving an email with the recovery header, the email provider will require the recipient to re-authenticate itself via a second channel in order to access the recovery email’s content.

While our solution is straightforward in principle, there are practical challenges that should be carefully addressed. In particular, the current email-fetching protocols (IMAP or POP3) do not protect any specific emails. In other words, the user’s mail user agent (MUA), such as Thunderbird, Outlook, and Yahoo! Mail, will fetch whatever emails belonging to the recipient from the Mail Delivery Agent (MDA) of the email provider, without supporting any additional authentication process. Under this emailing infrastructure, it is infeasible to protect any specific emails. To address this challenge, we attempt to protect the recovery email content, instead of the email itself, through the use of an “intermediate token.” Specifically, the email provider intercepts the recovery email content and replaces it with an intermediate token. Thus, the recipient will only receive the token. Then, when the recipient opens the recovery email, the intermediate token should be sent back to the email provider by the recipient to initiate the re-authentication. The token submission can be done in different ways, e.g., clicking a URL that embeds the token as in our implementation (see Figure 4.2a). The email provider will release the content of the original recovery email only if the re-authentication is successful. Furthermore, to ensure the legitimacy of recovery emails, we

Figure 4.2: Account Recovery Examples.

(a) Actual recovery email

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=emailprovider.com; s=mail;
t=1459434822; bh=6WjAFUdPOQRG+fZLZp2+0XtHkOKwLEQ2Zi75pnBr1fl=;
h=Date:From:To:Subject:Recover:From;
b=MLnCsB9uY7z2HYz8yZtdVhnQrpeHu92+gGX84eedjKfkkn0i6gS1iEsP/496U344X
+P7tA0nDMVKnn8yFX0dOwK+aJAhkke9Mc3lVBtWRp/PjlrSbObN6z0sSZjGKhRvyKe
UAD5kXFSHHQJEKfNgu4w0vJ8nqG+4MxrQbgxAFEU=
Date: Thu, 31 Mar 2016 14:33:42 +0000
Subject: Recover an account
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Disposition: inline
Recover: 1
User-Agent: Mutt/1.5.21 (2010-09-15)

This is a password recovery email, please visit www.emailprovider.com/reset/tuouvADBDCFbakwpfxk.html
```

(b) Email content from the account provider



The screenshot shows a web browser window with the address bar displaying www.██████████.com/recovery/tuouvADBDCFbakwpfxk.html. The page content reads: "You have requested to reset password on ██████████. click the following link to reset your password. <http://www.██████████.com/recover/AOIFfoiwej2308jAFNIO13FASadsfio1e>".

enforce the use of the Domain Keys Identified Mail (DKIM) [33] protocol, which uses public-key cryptography to ensure the legitimacy of the sender and has already been deployed on more than 80% emails as reported by Google [41].

Overall, SEAR incurs little user experience change since it only requires a second authentication factor in the rare case when a user needs to reset or recover a password. More importantly, it can be seamlessly integrated into today’s emailing infrastructures and MDAs. Moreover, it is fully backward-compatible. When one or both email parties do not comply with SEAR, a recovery email will be treated as a normal email.

4.6.2 Implementation

We implement SEAR on well-known open-source projects to demonstrate its simplicity and compatibility. SEAR requires minor modifications on the account provider and email provider sides to meet its specification. We use two Amazon EC2 Ubuntu server instances [1] to act as the account provider and email provider, respectively. They both run Postfix [7], a widely adopted open-source Mail Transfer Agent (MTA). We use Mutt [80], a text-based email agent to generate recovery emails. The legitimacy of emails is protected via openDKIM [6]. On the email provider side, we modify

the “clean-up” procedure in Postfix source code to feature recovery header checking and subsequent actions. We use a URL to embed the intermediate token, such that the user can directly click the URL to submit the token to the email provider. A sample of email received by the user is shown in Figure 4.2a. The page pointed to by the URL requires a second authentication. If successful, the recovery email content is displayed on the webpage, as shown in Figure 4.2b. The user can then directly reset a password through the URL on this protected page in a conventional manner.

SEAR induces little overhead for two reasons. First, only a small fraction of emails are account recovery emails. By examining different everyday email accounts, we roughly estimate that account recovery emails make up 0.4% of all emails. All other emails are processed normally. Second, its implementation does not introduce any expensive operations. Even in major email providers’ distributed systems, it will not introduce any bottleneck. All modules used by SEAR are mature techniques, which have been extensively tested and used for other purposes. The storage overhead is also negligible since the extra data stored are just the intermediate tokens, which can be as small as 20 bytes in our implementation.

4.7 Conclusion

In this chapter, we investigate account recovery at popular websites by examining their recovery protocols. Through extensive analysis of the security features of those websites, we observe that 92.5% of them rely on emails to reset user passwords. Even worse, the user accounts in a significant portion (81.1%) of the websites we reviewed can be easily compromised by mounting an email recovery attack. However, many email service providers fail to realize such security threats and have not yet taken serious actions to protect recovery emails, leading to a single point of failure of using email for account recovery. To mitigate this problem, we introduce a lightweight Secure Email Account Recovery (SEAR) mechanism to provide extra protection on account recovery emails. SEAR can be seamlessly integrated into modern email infrastructures with a full backward-compatibility.

Chapter 5

UTrack: Enterprise User Tracking Based on OS-Level Audit Logs

5.1 Introduction

Nowadays, cyber-attacks have been becoming more sophisticated and stealthy. In an Advanced Persistent Threat (APT) attack, on average an attacker may lurk in the target network for more than half a year, escalating and maintaining the access privilege without being caught [110]. As a result, there is an increasing demand of user tracking inside an enterprise network, in order to improve the visibility for the network monitoring, and help security analysts to make informed decisions on the detection of insider, targeted, or APT attacks. A recently enabled paradigm in the security industry, called User Behavior Analytics (UBA) [5, 109], is built upon this foundation. UBA categorizes a range of techniques that keep monitoring user activities and identifying those that deviate from normal user sessions. While UBA is a rather broad concept that can be applied to many scenarios on a different level, granularity, and scope, its fundamental building block is to accurately identify and model user activities. Capturing user activities inaccurately or incompletely could result in incorrect detection or analysis, and even render a UBA system useless.

Towards more accurate user modeling and verification, contemporary UBA approaches attempt to fuse data from different data sources for creating a more comprehensive risk profile [101, 111]. Though useful in many scenarios [101, 109, 111], an inherent limitation is that they all lack a holistic

view on systems since data are collected from only a couple of security-sensitive applications, such as firewalls and proxies. Under such a setting, many meaningful events could be missed, not to mention the difficulties of correlating data with different syntax and semantics from a variety of sources. A natural approach would be to leverage log data at the operating system (OS) level, which can record data for all applications under homogeneous syntax and comprehensible semantics. Such an audit log system is widely deployed in many security infrastructures [69, 70, 74, 78, 79], mainly for forensics purposes.

In this project, we develop a novel user tracking system named UTrack by leveraging the rich system log data to universally monitor user session activities. Our ultimate goal is to present a user session profile that is accurate in tracking user activities and concise in the output report. We identify and tackle two major challenges. The first is to bridge the semantic gap between user accounts and human users in both in-host and cross-host scenarios. This is done by tracking causal relationship among processes through the user session root and correlating network events to identify network control channels. The second challenge is to address the “needle in a haystack” problem stemmed from the huge volume of log data through a variety of data reduction techniques. Unlike many previous works on log data reduction [75, 122] that target at information-lossless reduction, our data pruning approach is to prune data that may carry meaningful information but are out of the scope of user activity tracking.

We deploy UTrack in an enterprise network that comprises more than 100 hosts running either Windows or Linux operating systems with real users. We manage to process log data from all the hosts on a single machine, and demonstrate that UTrack is able to accurately identify and concisely present the events that represent activities of a real user inside the network in a human-consumable fashion.

In summary, we make the following contributions.

1. We develop a new universal user tracking mechanism (UTrack) based on OS-level audit logs.

UTrack aims to bridge the semantic gap between human users and computer user accounts by identifying and associating system events that appear in different user accounts and different hosts but belong to a single user session.

2. We apply effective data reduction methods on user session profiles to achieve a scalable and salient presentation. The reduction mainly involves detecting interactive processes and modeling common data patterns.
3. We implement UTrack in a real enterprise environment, with data collected from more than 100 hosts. Our evaluation results show that UTrack is accurate and concise in presenting user activities. UTrack scales well with a low resource consumption.

5.2 Motivations and Challenges

5.2.1 Motivations

Contemporary user behavior monitoring is mostly done on disparate applications and services. However, such a methodology has drawbacks that limit the usability of the monitoring system. The first drawback is the lack of completeness. In these systems, only a small portion of user activities are recorded and analyzed, since the logs are only generated from applications that are usually perceived to be of strong security indication, for instance, a firewall, a web proxy, or a sensitive database service. All other user activities are not actively monitored. However, a successful attack, especially an APT attack, usually comprises many individual steps. The traces of each step may be buried in seemingly less interesting events that are not recorded by applications. By connecting these dots, one may detect an intrusion that cannot be identified by conventional user behavior analytics. In contemporary user tracking schemes, the auditor lacks this holistic view on the entire system.

The other limitation is the difficulty of correlating log data. Data collected from different services and applications may be of different formats, granularity, and semantic levels. Parsing and correlating data from different sources is very challenging. As a result, data from individual sources are independently handled and analyzed in many cases. Shashanka et al. [101] attempted to associate subjects from different data sources, such as different IP addresses and user accounts. However, the capability of such an association is limited to a small scope, where the subjects are tightly bounded. Therefore, the inspector lacks view on the connections among critical pieces of

puzzle from all data sources.

System Opportunities: A universal user activity tracking system, which monitors activities of all users inside an enterprise network, is very useful to resolve or mitigate the aforementioned problems. However, recording all activities of individual users in the entire network may incur significant system overhead. To balance the trade-off between system overhead and data granularity, we leverage an OS level log system to collect data from each host inside a network. The OS level log system collects low level system objects, such as processes, files, and network connections, which largely preserve the running states of a computer at a certain time. Thus, it can be used to accurately reconstruct the causality among objects with clean semantics. Meanwhile, the data volume is at a manageable level. Nowadays, many enterprises have deployed such a log system for forensics purposes [69, 70, 74, 78, 79].

5.2.2 Challenges

5.2.2.1 Accurate Modeling of User Behaviors

When processing audit logs, a user account is often considered equivalent to the user itself. This is mostly true in some high-level network applications, such as Facebook and Twitter. However, this assumption no longer holds when it comes to low-level OS events.

Unlike application-specific logging that is clearly defined and has much higher semantic awareness, a generic OS-level log system monitors events with respect to individual user accounts. In an enterprise network, a user may have multiple user accounts, and a user account could be accessible by multiple users. For instance, a network administrator could access both its personal account and the root account on a web server. The web server may also be managed by several system administrators. As observed in our network, the discrepancy mainly comes from the following three scenarios.

Account Transition: Managing account privilege and ensuring proper isolation among different privilege levels are essential to an operating system. However, user accounts with lower privileges sometimes need higher privileges to accomplish certain tasks, and it is mainly accomplished in two ways, namely, setting the UID of a process (e.g., “ping” command) or having a higher privilege ac-

count to do the task (e.g., through “sudo” or “su” commands). Thus, a simple task done by a single user may involve several user accounts, and the same account may also be involved in activities performed by multiple users. Furthermore, one user account granted the root privilege is able to interact with the system on any other account’s behalf. Though this does not usually happen in normal operations, it is possible when an attacker exploits this system trait to mask its malicious behaviors.

System Service: In a typical operating system, there are many applications and services running in the background. Meanwhile, many system accounts are created to achieve a finer granularity of access control for those services. Although these accounts do not represent any individual user, they are delegated to perform certain tasks by other users. For example, when the PostgreSQL database server receives a request to access the database, the server daemon creates a child process to process the request. Therefore, all access activities at the database server are recorded as from the account “postgres”, regardless of the real user that is in fact accessing the database.

Credential Sharing: It is possible that the same account is shared among multiple real users. A typical scenario is the “root” account on a server, which may be managed and shared by several developers or administrators.

In general, there exists a semantic gap between user accounts and human users. It is no reliable to solely rely on the user accounts to track the behavior of a user, due to the lack of proper linkage of user account transition and service account delegation. After realizing this semantic gap, we develop our user tracking system by clearly setting boundaries between the two concepts - the term “user” always indicates a real human user, and the term “account” always indicates a user (or system) account in a computer system.

5.2.2.2 Identifying Data Triggered by Users

The other challenge we face is to sift out data that are directly related to user behaviors. We observe that only a small portion of log data are triggered by direct user interactions with the computer, and others are spontaneous or scheduled system events, such as automatic updater and cron jobs. Since human interactions are the natural target of a user tracking system, we attempt

to identify system events that are triggered by users’ actual interactions in order to achieve a much higher scalability and remove unnecessary distractions from security auditors. However, OS level log systems usually only record the causal relationships among primitive system objects, such as processes, files, and sockets, to gain clean data semantics and conserve system resource. They do not usually keep track of the operations on I/O devices, such as a click on the mouse or a tap on the keyboard. Without such information, it becomes non-trivial to identify events stemmed from users’ interactions. To achieve our goal, we follow the lineage of the UI management components to identify possible processes that have an open interface to the users and rely on many useful features to determine interactive processes.

We also need to address the semantic gap between the actual high-level user behaviors and low-level system interpretations. In many cases, a simple operation from a user may result in a large number of system events; however, most system events are highly repetitive and predictable, and thus carry little information. To eliminate this redundancy, we model the file sets that are frequently accessed by processes, and only record the events that do not fit in the model. In addition, we model repetitive execution “branches” of a process and compress the repetitive ones.

5.3 System Overview

Nowadays, many organizations have started to deploy an agent on each host in their enterprise networks. UTrack works under the same context of these forensics-purposed OS level log systems. Three types of system objects (process, files, and network sockets) and their interaction events (e.g., a process creating a child process or reading from a file) are recorded. Each event also carries attributes that describe the activity, such as the event time, user account, file pathnames, and socket IP addresses, etc.

UTrack aims to help a system auditor to understand the activities of users inside an interconnected enterprise network by associating both in-host and cross-host activities performed by the same user in a specific user session. The input to UTrack is a data stream collected from all hosts, and it can be either a real-time stream or an offline history database. Generally, it consumes the data from a start time (T_s) to an end time (T_e), and outputs user session profiles to describe the

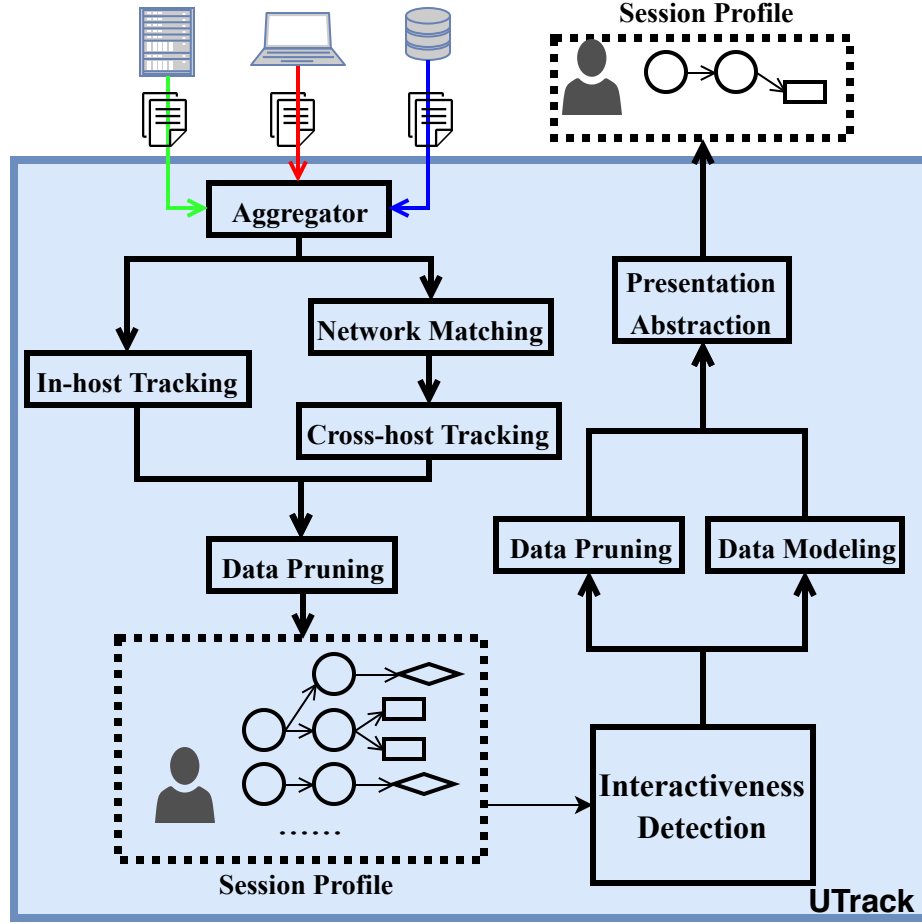


Figure 5.1: UTrack Overview

activities of a user session within the time period. If the data is an online stream, the end time T_e is set to a distant future time. The user session profile is represented in a forest structure, and the time is usually set in terms of weeks, days, or hours in different use cases.

Most forensics techniques consider a system object, such as an identified trojan process, as a Point of Interest (POI), and aim at understanding the provenance or impact of POIs. In contrast, UTrack attempts to understand the behavior of a user in a session. In other words, the user itself is the POI. In most cases, the behavior of a user is much more complex to describe than a single attack incidence in the system. Similarly or even worse, UTrack suffers from the same data explosion problem, which makes it difficult to focus on the real interesting events. Thus, it is imperative for UTrack to identify and keep the most relevant data, which can also help save system resource.

Figure 5.1 illustrates an overview of UTrack. UTrack consumes a data stream of log events from the aggregator, which receives, sorts, and sends out the data from the agent-enabled hosts. The first task of UTrack is to construct user session profiles by correlating both in-host and cross-host activities, mainly relying on the process lineage and network event matching. However, this session profile contains a large amount of system-generated data that may not be directly related to the user’s operations, but can easily overwhelm other interesting events. To mitigate this issue, we apply interactiveness detection on the user session profile to identify the processes that have actual interaction with the user. This step directs us to the events that are more relevant to user tracking. We also model the files, network connections, and “sub-branches” of the interactive processes to further compress the low-entropy events. The output of this step is a more salient session profile, which can be directly construed by human auditors or be inputted to further security measures.

UTrack works in an online fashion. It gradually builds the user session profiles while consuming system events on the fly. It is important for a UBA system to promptly analyze the data, so that anomalies can be identified in their early stage and triaged to prevent further damages. On the other hand, UTrack can also work offline in forensics analysis by reconstructing the data stream from the log database. In order to facilitate this feature, we build a data replayer to replay the history data from the database, which is detailed in Section 5.6. This data replaying tool is also useful on implementing, debugging, and evaluating our user tracking system.

Note that UTrack does not aim to replace conventional forensics techniques, such as backtracking or forward tracking. Instead, it is indeed complementary to those techniques to better secure an enterprise network. Nowadays, it is a common case that people do not really make good use of big data, and a large amount of collected data remain in the warehouse without generating any useful insights. UTrack demonstrates a new perspective to better leverage the collected rich system data for system security and management purposes.

5.4 UTrack Event Association

UTrack is capable of tracking users across an enterprise network by linking events from different hosts. Note that we no longer depend on the owner of the process (i.e. the user account) to determine

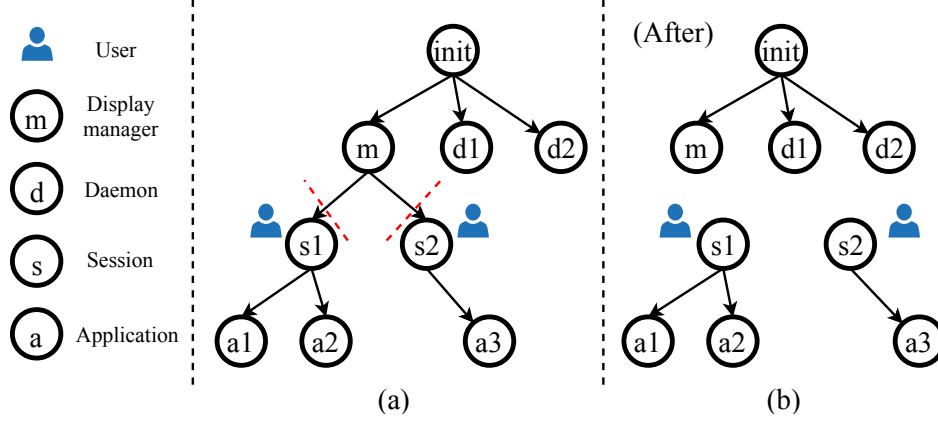


Figure 5.2: Session Root Isolation

the real performer of an event. Instead, the process owner is only used as side information to give a hint of who the performer might be. In the following, we first introduce the tracking mechanism on a single host and then extend it to the cross-host scenarios.

5.4.1 Tracking In-host User Activities

5.4.1.1 Process Lineage

Modern operating systems usually maintain all alive processes in a tree structure. For example, in Linux, every process, except the *init* process, is forked by a parent process. This parent-child relationship widely exists among processes and is useful to determine the performer of most system activities. Specifically, we consider the user of the child processes to be the same as that of the parent process, unless we have a special reason to cut the lineage and attribute the parent and children to different sessions. For instance, a user might open a Bash terminal and run the “ls” command in the terminal. Since the “ls” command is executed by a child process forked by the bash process, UTrack considers both processes to be performed by a single user.

This parent-child relationship is a fundamental building block of many forensics analysis techniques [69, 70, 74, 79], which usually expand the investigation from POI, i.e., the detected point of an attack. Timing is also considered to mitigate the possible dependency explosion and find the most relevant events. In contrast, UTrack tracks all the parent-child relationships among processes for constructing a more complete user session model.

We do not keep track of file data control flows, which are usually considered in forensics techniques. This is because when a process has written to a file, the file has a causal relationship with all the processes that read the file afterwards; however, this causal relationship is out of the scope of user tracking where the user activities are the target. Furthermore, it introduces too many dependencies that may unnecessarily complicate the analysis.

5.4.1.2 User Log-on Sessions

Since one OS usually structures all its processes in a tree (or forest in Windows), when activities from multiple users are recorded in a host, it is far from adequate to solely rely on the process lineage for tracking each user. Thus, we must find a way to attribute related nodes to different users. A critical observation is that a user must have an interface to interact with the computer, and usually the first step is to log on the computer for user authentication. In addition, the OS usually organizes the processes under one user log-on session in a tree structure, and normally there is a root node as the ancestor of all processes created in the user session. We call this node a *session root*. Figure 5.2 shows an example of Linux instance. Each user logging in the system has a corresponding session root (i.e., node s_1 and node s_2), which is usually a child process of a running service. Their activities (for instance, open an application) are reflected in the subtrees under the session root. UTrack utilizes session roots to identify the activities of each user. It brings us two benefits. First, it helps to separate processes and activities triggered by users from those generated by the OS or system services. Second, it can differentiate activities among multiple users who have logged on the host simultaneously.

UTrack identifies session roots from several known patterns. For a normal user, the most common way to interact with the computer is via a Graphic User Interface (GUI). Even command line interactions are included since the terminal window itself is created in the desktop environment. For example, in Linux, the X display manager (a process usually named **dm*) manages the login screen and organizes a user session in child process. In our experimental environment, the most common display manager is *lightdm*, and thus the session root in this case is a *lightdm* session child with a session ID. When users log on a server through virtual consoles or no X server is available on

the server, the session root is */sbin/login*, which is a child of the system *init* process. It is even easier for Windows, as it is a GUI-based OS and the user interaction with the OS is usually through the GUI. We determine the windows process “winlogon” as the session root, since it initiates the user authentication process and becomes the root of the desktop environment when the login succeeds.

Remote logins, such as through ssh and telnet, are envisioned as user cross-host activities since events from multiple hosts need to be correlated to track the relations. We elaborate how we handle cross-host activities in Section 5.4.2. When logins are from hosts that do not have an agent installed or the login happens before the tracking start time T_s , we identify session roots based on the service pattern. For example, the ssh daemon creates a dedicated shell for whoever has successfully logged on the computer via ssh. As such, the dedicated shell is considered as the session root.

5.4.2 Tracking Cross-host User Activities

In an enterprise network with many inter-connected hosts, one user may need to work on other hosts or request resources and services from servers. It is critical to track the cross-host user activities in order to achieve a better coverage than local-only tracking. A number of previous works have been done to help understand how a request is processed in a complex distributed system using middleware or application level instrumentation [16, 108], statistical inference [12, 96], or system call log and analysis techniques [98, 106]. However, they all cannot accurately work under generic-purposed OS logs.

We propose to track cross-host user activities based on one key observation that after receiving remote requests, a server will act on behalf of the requester. Most servers have a daemon listening to incoming requests and processing the requests accordingly. There are two types of server architectures, namely, event-driven servers and worker-based servers. For the event-driven servers, since a thread could handle multiple incoming requests in a non-blocking, interleaving manner, it is hard to correlate a remote request with the corresponding activities of the server without specific assistance from the server. Therefore, our main focus is on the worker-based servers, where a network request is solely handled by a worker. Worker-based servers are popularly used in enterprise networks with a moderate number of users due to the ease of coding and maintenance. A worker-based server may

support two working modes, namely, on-demand worker creation and a pre-allocated worker pool. As an example of the first mode, *sshd* daemon accepts a remote network connection, and creates an interactive command language interpreter process, such as a *bash* terminal. Thereafter, the newly created command interpreter process is controlled by the requester, and any activities performed by the process should be attributed to the requester, regardless of the user account that owns it on the server. We call this process a *delegate* of the remote user.

It is more tricky to handle the worker pool mode. In UTrack, one process node in a user session completely belongs to the user. However, it does not fit well with the mode of a worker pool, where multiple long-living workers are pre-created and each worker only dedicates a partial of its lifetime for a network request. In order to accommodate such a case, we introduce a new notation – *virtual process* – to model a span of the worker’s lifetime. When a worker begins to work on a requester, we create a new node (i.e., the virtual process) in the user’s model, and the new node records all the activities of the worker during this time. We illustrate this process in Figure 5.3, where two users make requests to a server at different times. The server dispatches the same worker to access two different files, f_1 and f_2 . A virtual node is created to represent the time lapse that a worker is processing each request. Eventually, the user model is constructed with each user associated with its own virtual process, which carries all data during the time when the real worker handles the individual request.

To track cross-host user activities, the first step is to find the communication channel between the server application (i.e., responder) and the user-controlled application (i.e., requester). Next, we try to identify a worker for the request using a rule-based method. In both worker modes, we observe that after establishing a network channel as a connection acceptor, a child process or a sibling (when the listener and workers are siblings) of the server process immediately accesses the same network channel and generates a number of events. Based on this pattern, we can determine the worker and attribute all the activities of the worker to the remote requester.

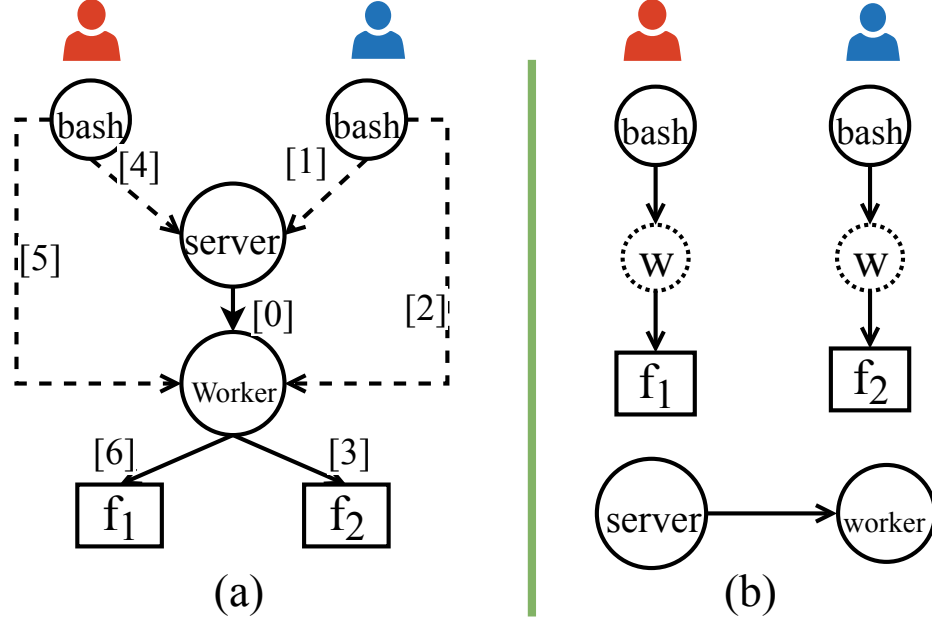


Figure 5.3: Virtual Process

5.4.3 System Cold Start

UTrack consumes data from agents during a pre-set time period to help understand the user sessions; however, it may encounter the cold start problem, namely, the history data is not available and the agents report events on scattered processes. If so, the linkage among processes may be missing. Since UTrack relies on the causal relations to identify user sessions, it requires to reconstruct these relations between processes. To address this problem, the agent periodically collects a system snapshot that stores the child-parent relationship among them. We use this information to reconstruct the causality relations among stand-alone processes and further extract user sessions in the reconstructed process tree. Note that the parent-child relationship recorded in the snapshot may not be coherent with that generated by UTrack, due to possible process delegation, user session identification and isolation, or the adoption of orphaned processes, etc. This discrepancy is in fact beneficial to our user tracking scheme. For instance, if a user starts a system service during the tracking period, the system service is regarded spawned by the user, and the activities of the service can be attributed to the user. However, if the tracking period is after the system start time and the service daemon is adopted by the “init” process, then the service becomes a part of the operating system and cannot represent any user. As such, we only reconstruct the parent-child relationship when the child process

has no existing parent in the UTrack model.

5.4.4 Scope

UTrack aims to connect events that are cross-host and cross-accounts. However, there are cases where UTrack cannot handle. For instance, it could fail to identify causality among processes due to inability to track IPC mechanisms, such as shared memory and shared files. It also cannot handle the event-driven servers, like those run NGINX. Similar limitations can be found in previous works [17, 98, 106].

5.5 Pinpointing User Activities

After correlating activities of users regardless of the process owner, UTrack collects user session profiles that keep track of all processes, files, and sockets in the memory during the entire tracking period. As a result, the generated data can become very large, and interesting events may be buried in piles of less-relevant data. Therefore, it is essential for UTrack to identify and keep only relevant and useful events. Redundant data must be pruned to release the pressure of huge system resource demand and to keep the security auditor from unnecessary distractions.

When conducting data pruning, we stick to the user-centric mentality by sifting out the events that are directly related to the user’s interaction with the computer system. This is because that a user session profile contains a collection of processes that are only used to facilitate user or system operations. For example, Ubuntu provides a number of tools and services, such as GNOME Virtual File System(gvfs) for I/O abstraction, update-notifier for newer version checking, zeitgeist for logging user activities, etc., which are less relevant to user’s actual behaviors. In contrast, interactive processes are the processes that a user interact with, such as a Bash shell or UI-based programs like Firefox, Notepad, etc. The behavior of interactive processes is likely a genuine reflection of the user operations. However, it is a challenge to identify those interactive processes from our OS level logging information, which does not include any user actions, such as mouse clicking or keyboard input. We relies on passive observation and prediction to find interactive processes, and multiple features have been identified to help distinguish interactive processes from other processes.

In addition to the interaction-oriented sifting, the data can be further compressed due to the highly repetitive patterns found in processes and files. We observe that the interactive processes are prone to generate sub-processes “branches” for different tasks. These branches could be similar to each other, regarding to the executable names, arguments, and files that are read. In many cases, these monotonous data can easily dominate a session profile and occupy over 90% data of the user profile. To address this issue, we model both the activities of an interactive process and the common files that are read by each executable, which significantly reduce the complexity of the user session profile.

5.5.1 Interactiveness Detection

The purpose of interactiveness detection is to find user-triggered events. We consider user-triggered events to be events directly resulting from a user action, such as opening a file using Notepad, etc. It should be noted that technically, all events are results of human user activities, since background procedures and processes, even the operating systems are installed by the user. However, since these processes are mostly regulated and expose behaviors dual to bots, we consider them to be non-user-triggered. Conceptually, we envision this procedure being similar to find bots/crawlers in a network, where the bots are essentially programmed by human users, but they expose very different behaviors and have little relation to active genuine users.

The interactiveness detection relies on passive observation of the OS events, so it faces several noteworthy challenges. First, passive observation is believed to be less accurate than active detection solutions, such as Catpcha [103,117]. Second, we do not have a specially tailored log system as those used in bot detection [48,118], or any side information such as social graph [28,35,48,116]. Lastly, system level events are low-level data whose semantic meanings are harder to derive. Sometimes, we need to associate other related events to truly understand the actual user operations.

To categorize unknown processes, we develop a machine learning approach that uses a number of useful features to distinguish an interactive process from other processes. Note that we need to keep the actual activities that are usually represented by child processes of an interactive process. For example, an interactive shell may run many commands, which is executed transiently. These

commands are not considered interactive processes. However, they represent the user’s activities and should be studied.

We introduce to use a new feature on the entropy of activity batches. A fundamental observation is that the interactive processes have irregular activities due to human involvement, so a process performing tasks at a fixed time interval is unlikely to be controlled by a real human user. However, treating each individual event as a task is problematic since a single task usually constitutes many steps and events. As a solution, we preprocess all events that are generated by the process to form a group of event batches, in which each batch represents a high-level task or operation. A batch consists of a group of events where each pair of adjacent events has an inter-arrival time of less than a threshold T . T should be carefully selected since it may result in putting all events into a single huge batch when it is too large, or losing the causality among events that are generated from a single task when it is too small.

We envision the time interval between two consecutive activity batches as a random process and decide if a random process is regular by computing the entropy rate based on empirically learned probability distribution [32, 47]. UTrack computes only the first order and second order entropy. It is expensive to calculate even higher order entropy, which may need prior knowledge to determine a probability distribution. Also, we observe that the first and second order entropy can achieve a satisfactory result.

5.5.2 Non-interactive Process Pruning

Instead of targeting at information-lossless pruning [75, 122], we can afford to remove less interesting data points when coping with our specific goal of user activity tracking. However, it does not mean we do not track other processes. Actually, we keep track of all alive processes that have any interaction with interactive processes or become interactive processes. The processes we pruned are those that do not have any lineage with an interactive process. In general, most processes that are not in a user session are pruned since they are system-triggered events.

For the processes in a user session, if they are not related to any user interactions, they are also pruned. We develop an online algorithm to prune those processes using a bottom up, and backward

propagation method. The pruning starts from the leaf process when the process is ended. If the leaf process can be pruned, it is removed from the child list of the parent process, and the parent process will be further checked to see if it can be pruned after the removal of its child process.

5.5.3 Data Modeling

The essence of identifying interactive processes is to find the activities of processes, since they are likely the direct results of the user’s operations. Therefore, all activities of the interactive processes are preserved in our user session profile. Due to its long-living and interactive nature, an interactive process usually has many sub-process branches representing user activities. However, these branches could be highly repetitive due to multiple reasons. First, even interactive processes may have periodic routines for updating, synchronization, etc. Second, user activities can be repeated. For instance, a user may run “ls” command many times in a terminal, and many commands intrinsically invokes “ls”. Third, there is a large gap between user operations and the interpretations of the computer system. Therefore, a single, seemingly atomic user operation may result in a large amount of low-level events. For example, when opening a Firefox browser, we observe that a significant portion of events are repetitive to serve the same low-level purpose, such as checking the system time or OS version.

There is a large room for the improvement on salience of a user session profile by modeling and compressing the files accessed by processes and the branches of interactive processes, respectively. Based on the observation that many processes with the same executable name and same arguments (e.g., Chrome.exe type=renderer . . . , we call them “mainexec”) may access a similar set of files, we are able to model commonly accessed files under a mainexec, and record only the difference. An example is shown in Figures 5.4(a) and 5.4(b). We notice that both mainexecs e_1 and e_2 access a common set of files ($\{f_1, f_2, f_3\}$), which can be abstracted by a model (m_1). This model-based technique has also been used in Arnold [40] to reduce instrumentation overhead.

Figures 5.4(c) and 5.4(d) show that the session profile can be further compressed if some branches are identical. Interactive processes often have identical branches that could easily overwhelm the auditor. Therefore, we can compress these identical branches by only recording the timing information

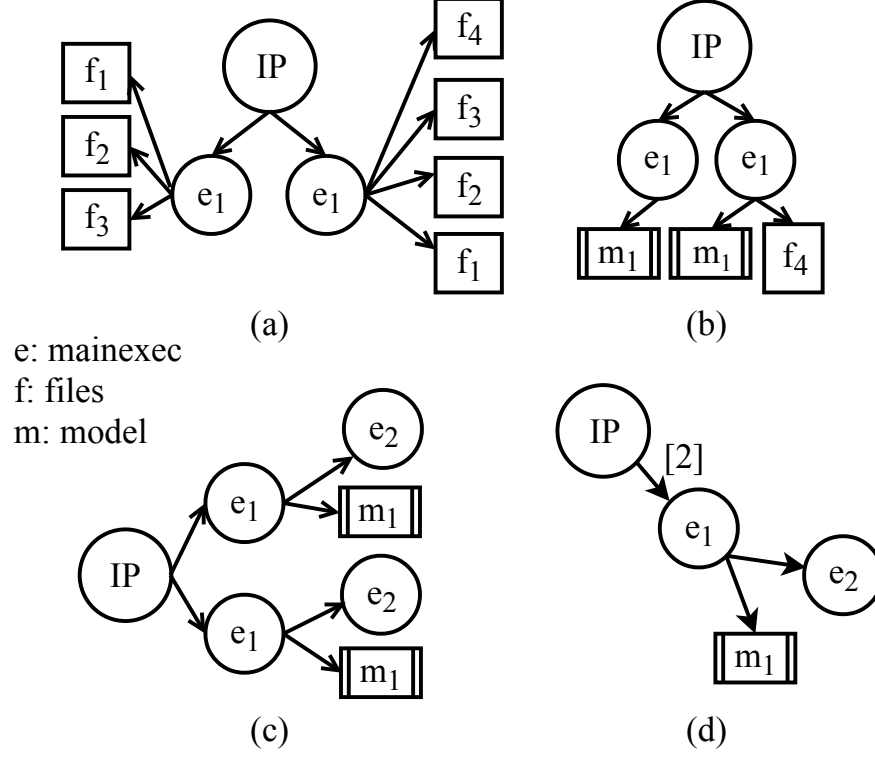


Figure 5.4: Data Modeling

and the number of occurrences.

5.6 Implementation and Evaluation

5.6.1 Experiment Environment

We deploy UTrack on 111 hosts of a real enterprise environment, 21 Linux hosts and 90 Windows hosts. An agent is installed in each host to collect and report system events. UTrack itself is written in Java and contains 8.3K LoC. We evaluate the performance of UTrack based on one month of data. Within this period, more than 4 billion events are generated, where 1.65 billion events come from Windows hosts and 2.41 billion events come from Linux hosts. To facilitate the use of history data, we implement a data replayer to replay the data recorded and stored in the database with their original timestamps. With the assistance of the replayer, we are able to replay the one-month data within 30 hours.

5.6.2 User Tracking

In our one month experiment, we identify 507 user sessions across 111 hosts. Note that the login screen itself is counted as a user session and excluded from our data. Among the total 507 user sessions, only 61 of them are Linux sessions. One reason is that Linux users are less likely to log off or restart their computers than Windows users. Besides, there exist 4 Linux hosts that do not have any user sessions, which means that they are used as servers and no one logs on the hosts through the Linux desktop environment. However, the activities in those servers may be correlated to user sessions in other hosts. On average, each user session lasts 4.6 day. We also observe that Linux sessions are significantly longer (9.1 days) than Windows sessions (3.9 days). More than 100 sessions last beyond the one month period, so they are excluded when we compute the average session lifespans.

For cross-host tracking, we first identify the communication channels. We correlate network events from all hosts by matching 5-tuple attributes, which include local IP, remote IP, local port, remote port, and the network protocol. However, due to port or IP recycling, two network events might be wrongly matched. To avoid such a situation, we add a constraint that two matching events should happen within a small time window. This small window should consider the possible errors caused by asynchronous clocks on different hosts and network resource recycling. In our implementation, we set the time window to 60 seconds, and we recycle the unpaired events after this time window.

In our environment, the number of all ready-to-pair network events stabilizes at around 20,000 to 25,000. We observe that only around 12.3% of network events can be eventually paired, and most of the matched network events (82.4%) are localhost channels. This is reasonable because any communication to the outside world cannot be paired as the other side does not have an agent. Even the internal communication may not be identified, since not all computers host an agent in our environment. Another case is the broadcast network events, which have multiple receivers. When the server is working in the worker-pool mode, it may take a non-negligible time to determine the delegated worker, since it needs to go through a network channel matching process. If a worker is found, a virtual process will be created for the requester. However, before the virtual process

Table 5.1: Servers with the Most Network Connections

ranking	Program Name	Number of Instances	User Instance	Mode	Host Type
1	sshd	134,492	671	Create New	Linux
2	smbd	8,120	428	Create New	Linux
3	Postgres	5,152	559	Create New	Windows&Linux
4	sendmail	1,218	17	Create New	Linux
5	httpd	874	841	Worker Pool	Linux

is created, the network request may have already been partially or entirely handled, because most requests are handled very quickly. Thus, one should record the mapping between the virtual node and the actual node, and migrate the stand-out events to the virtual node once the delegation relation is established.

During the one-month experiment, we observe more than 186 programs that accept network connections, and the top 5 programs are listed in Table 5.1. The “Number of Instances” column shows the total number of request processing instances we observed. In our environment, since a server frequently runs “ss” to localhost for system backup, we observe a large number of ssh events. We also find a Postgres database that constantly stores new data from network connections. An Apache server runs the default pre-fork Multi-Processing Module (MPM) to support a worker pool. The “User Instance” column indicates the instances that belong to a user session. It shows only a small portion of the cross-host activities can be seen in a user session, since most of the virtual process nodes are pruned due to their irrelevance to user activities.

5.6.3 User-centric Activity Tracking

To detect interactive processes, we employ an important feature, the “regularity” of activities, which is measured by the first and second order entropy rates on the inter-arrival time of activity batches in a process. In our implementation, we empirically set the threshold of batching (T) as 350 ms. Figure 5.5 illustrates the CDF of the number of batches a process have when doing the interaction detection, and the number of events a batch has. Both of them are heavily tailed. For clarity, we limit the x axis to be within 100. We observe that 70.7% processes have only one batch, and 99.1% processes have fewer than 100 batches. Similarly, more than 78% of batches have fewer than 5 events, and 97% of batches have fewer than 100 events. When computing the entropy of the process, we round the interval to second-granularity to mitigate noise.

Table 5.2: Classification Results

	Interactive	Non-interactive	Total
Classified as Interactive	447 (TP)	181 (FP)	628
Classified as Non-Interactive	5 (FN)	25,746 (TN)	25,751
Total	452	25,927	

Another important feature we use is the lifespan of a process, which describes the time duration from the time the process is created to the time it is ended (or the current time if it is still alive at the time of decision making). The lifespan is a strong indicator of an interactive process. Due to the communicative nature, interactive processes tend to live longer than other processes. Therefore, a large amount of transient processes, especially in Linux hosts, could be filtered out by inspecting their lifespan of milliseconds. Figure 5.6 illustrates the CDF of process lifespan, which indicates that around 90% of processes have a short life time less than 20 ms.

Our model also considers the context of a process, including the parent process, the number of child processes, and the nature of the parent, as a set of important features. If a process is created by a window manager (e.g., compiz is the default window manager in Ubuntu 16.04), the process is more likely to be an interactive process. We also blacklist 16 types of commonly seen non-interactive processes (e.g., “/etc/update” periodically runs on Ubuntu OSes) to remove unnecessary distractions. It is hard to maintain a white-list since a process could be sometimes interactive and sometimes non-interactive depending on user operations.

In total, we extract 11 features to build a random forest model based on Weka [53] to predict if a process is interactive. In the training stage, we manually examine and label processes in 50 user sessions (20 Linux sessions and 30 Windows sessions) in a one-day period. An advantage with manual effort is that we can deliberately search the mainexec of a process online and better understand what the process is used for. The machine learning module is triggered when a process’s ending event is observed or the process has too many activities, including batches, network connections, and child processes.

More than 26,000 processes are labeled after filtering out the processes on the blacklist. We apply 10-fold cross-validation on all the processes, and the evaluation results are shown in Table 5.2. Our machine learning module has a high accuracy and recall of 99.3%. However, the module has a fairly

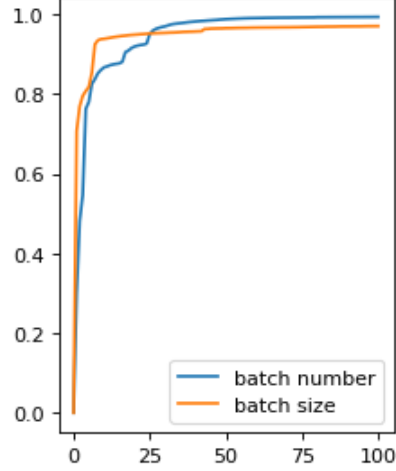


Figure 5.5: Batches in Processes

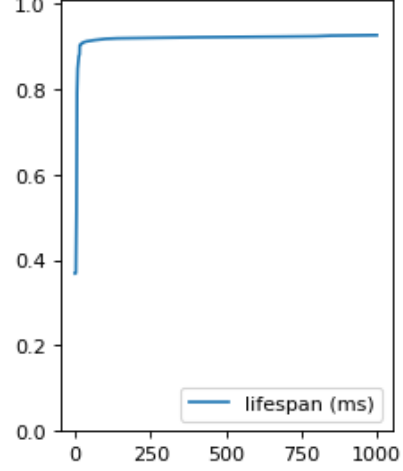


Figure 5.6: Lifespan of Processes

low precision, which is only 71.1%. Therefore, in a user profile, there are a non-negligible portion of processes that do not really interact with the user. However, even with those false positives, the user profile has been largely reduced since the dominant factors of non-interactive processes are mostly identified and pruned off. We argue that having some wrongly classified processes in the profile is acceptable since the amount of noise created is limited and can be easily identified by the security auditors.

5.6.4 Data Modeling

We model files accessed by both processes and sub-process branches, and compress them by only recording the deviations from the model. This modeling process is done when the process is ended. In most cases, the interaction detection module also kicks in at this moment. It produces the same results no matter which module runs first, since the modeled processes will be pruned if they or their ancestors are later decided to be non-interactive. On the other hand, pruned processes do not go through the interaction detection stage. In fact, most processes do not live more than 20 ms (as shown in Figure 5.6), and will be immediately pruned or modeled. In our implementation, we apply data pruning, if applicable, before modeling for higher efficiency. As such, the evaluation results are only applied on the interactive processes and their offspring. In contrast, non-interactive processes are pruned off before any modeling can be done.

We build an FP-Tree [54] to model the commonly accessed files of the same mainexec on each

host. The FP-Tree is frequently used to mine association rules from a growing data. We set the Minimum Support Threshold (MST) to 0.3, so that files with frequency less than 0.3 are discarded from the tree. To identify the common models, we record 100 instances or the number of instances when UTrack has processed all events within the requested time period. The FP-Tree no longer changes after the training period. At this stage, new processes with the same mainexec can be modeled using the Tree.

Since many processes may have the same process branches that exhibit exactly the same system behaviors, we compress the same branches into one and record the number of occurrences. Some processes may have a random token in the mainexec, such as the Chrome renderer processes. We handle them in a case-by-case manner. Similar to data pruning, our online branch modeling algorithm adopts a bottom up approach, which starts modeling from the leaf processes and propagates back to the parent process if no leaf process is alive. When a parent process notices that multiple child processes have the same model, it merges these child processes and records the occurrence of the model. The model of each process is represented in an XML-styled structure, which stores the information of files, remote IPs, and mainexec of itself and its offspring. Note that the backward propagation in our user session model stops at the interactive processes. This is because the model becomes increasingly large in lower-depth process nodes due to the large number of child processes. Also, it does not provide any help on compressing the data, because the process models at these levels are rarely identical and thus hardly compressible.

In the 507 user sessions, 8,382 interactive processes and 176,822 other processes (i.e. the child processes of the interactive processes) are identified. After modeling the branches, more than 71% of the processes are compressed, leaving us 8,382 interactive processes and 50,394 of their child processes. All 58,776 processes access more than 1.2 million files. After data modeling, the number of files reduces to 502,446 (around 60% reduction), where a file model is counted as one file.

On average, each user session has about 116 processes, which is not a large number considering that a user session can last for several days. However, the number of accessed files is large, almost 1,000 files per session profile. We observe that the majority of files come from process initialization, since a new process can easily read hundreds of files during initialization. Although this initialization

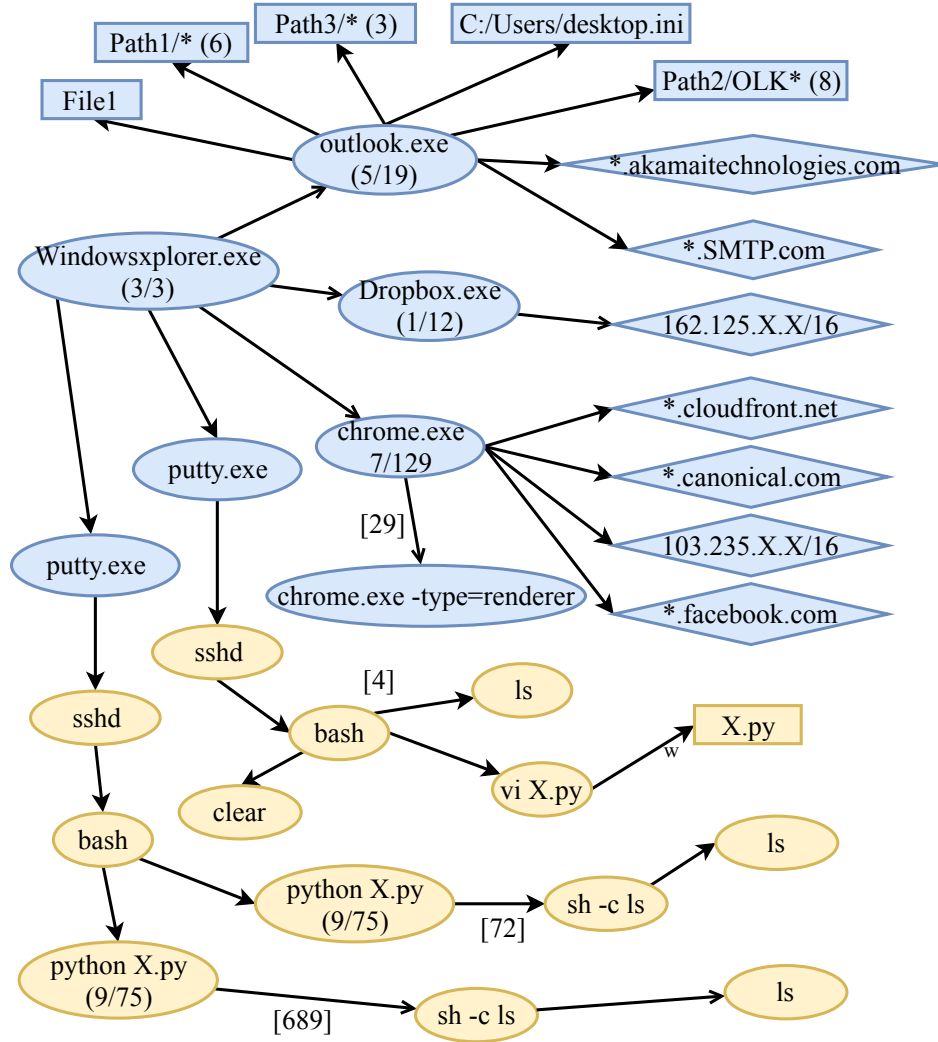


Figure 5.7: Example User Profile

Path1: C:/Users/X/appdata/local/microsoft/windows/temporaryInternetfiles/content.IE5

Path2: C:/Users/X/appdata/local/microsoft/outlook

Path3: C:/Users/X/appdata/local/TEMP

File1: C:/program files/common files/system/ado/msadox.dll

process can usually be modeled, our experiment period may not cover enough instances of the processes, so all the files are preserved. When UTrack runs for a sufficiently long time, these processes can also be modeled to reduce the number of files that a profile has.

5.6.5 Graph Presentation

A graph presentation of a user session profile visualizes the activities of a user, and it can provide security auditors with system insight to make informed decision. However, it is challenging to

present the session profile on a single graph, since the graph could be very large due to processes accessing a large number of files or network connections in a long session. To alleviate this issue, we abstract similar files and network connections when visualizing the session profile graph. The abstraction is applied to files and network events. For files, we build a trie (a.k.a prefix tree) to abstract the prefix of a set of files. Due to loss of information, we should carefully gauge the degree of abstraction in different scenarios. In our case, the gain is the number of files that can be abstracted when folding all nodes under a single node in the trie, and the cost is the loss on the file path levels and filename characters. For network events, remote IP addresses are abstracted by using the top-level and secondary-level domain names, e.g., “*.google.com”. When the domain names cannot be found through a reverse DNS lookup, we adopt a network mask approach similar to [55]. Specifically, class B and class C subnets are abstracted in a similar manner to that of file abstraction.

Figure 5.7 depicts a simplified user session profile identified from our network environment. Two different colors indicate two hosts. Processes, files, and remote IPs are represented by ovals, squares, and diamonds, respectively. The complete graph has a total of 323 nodes. For simplicity, we omit most unimportant files from the graph. We illustrate the 5 abstracted files read by “outlook.exe” in the graph to give a basic idea of how the files look like after the abstraction step. For other processes, we put the number of abstracted files and the number of total files inside the process node. The case of a process without a number indicates that the files are completely modeled. Meaningful files are preserved as nodes on the graph. The number of abstracted branches is shown in brackets on edges.

From the figure, we can easily find the user’s activities in the network. The user logs in the system on a Windows host and the session lasts for six hours. The session spans two hosts through interactive ssh connections using “putty.exe.” On the Windows host, the user browses the Internet via Chrome and uses Outlook for emailing. The user then logs on a remote Linux host to edit and run a python program “X.py” (the file name is anonymized), which further runs the “ls” program a couple of times. In general, a graph-based session profile presentation can be easily understood by a human auditor, and provides insightful visibility on the activities of a user.

5.6.6 Use Cases

Many more UBA features can be directly applied to UTrack for anomaly detection. For example, one can audit the roles (user accounts) that a user has been playing in the network from the user profiles and identify higher-level inconsistencies. For instance, one cannot be both “Alice” and “Bob” in the same session profile. Besides providing a foundation of UBA systems, there are many other use cases that can be built on top of UTrack. For example, it can be used in forensics analysis to study the behavior of attackers (such that the attacker becomes the POI) and reveal more seemingly benign behaviors which are in fact part of the cyber kill chain. UTrack can also be used to determine the value of files (by the amount of time an employee spent on a file) for backing up digital asset. This is particularly useful in fighting ransomware.

5.7 Conclusion

This chapter presents UTrack, a novel user tracking system that connects events under different user accounts and from different hosts to form a more holistic user session profile. With the help of UTrack, a system auditor can easily find out the activities of users inside enterprise networks. UTrack works by identifying a session root and then following both the local process lineage and the network control flow of the session root to associate other activities of a user. For scalability and salient description, UTrack employs an interaction detection module to sift out the most relevant events that result from users’ interactions, and models common file and activity patterns. We deploy UTrack in a real enterprise environment that contains 111 hosts for a period of one month. Our evaluation shows that UTrack is capable of producing accurate and concise user session profiles for system auditors to use.

Chapter 6

Related Work

6.1 Password Study

Many works on password have been done through decades of password-based authentication. In one of the earliest works [86], Morris and Thompson found that passwords are quite simple and thus are vulnerable to guessing attacks. Nowadays, passwords studies are done on a much larger scale, and thus they reveal more insightful and accurate characteristics of passwords. For example, Mazurek et al. [84] measured 25,000 passwords from a university and revealed correlation between demographics or other factors, such as gender and field of study. Li et al. [76] conducted a large-scale measurement study on Chinese passwords, in which more than 100 million real-life passwords are studied and differences between passwords in Chinese and other languages are presented. Bonneau [19] studied the language effect on user passwords from more than 70 million passwords. Bonneau et al. also found that a user's birthdate appears extensively in 4-digit PINs [23]. Malone et al. [81] studied the distribution of passwords on several large leaked datasets and found that user passwords fit Zipf distribution well.

There are also many works investigating more specific aspects of passwords. For instance, Yan et al. [123] and Kuo et al. [73] investigated the mnemonic passwords. Veras et al. [115] showed the importance of a date in passwords. Das et al. [36] studied how users mangle one password for different sites. Schweitzer et al. [100] studied the keyboard pattern in passwords. Apart from the password itself, human habits and psychology toward password security are also being extensively

investigated [44, 57].

6.2 Password Strength Measurement

Password Strength measurement still remains a challenge. It has been shown that Shannon entropy can hardly accurately describe the security level of passwords [26, 67, 95, 120]. Thus, a number of metrics to measure passwords is introduced. Massey [82] proposed guessing entropy, which shows the expected number of guesses needed to make a correct guess. Several other most commonly used metrics include marginal guesswork μ_α [95], which measures the number of expected guess needed to succeed with probability α , and the marginal success rate λ_β [24], which is the probability to succeed in β guesses.

6.3 Password Cracking

Password-cracking methods have been discussed for more than 30 years. Attackers usually try to recover plain-text passwords from a hashed password database. While reverse hashing function is infeasible, dictionary attacks are found effective [86]. Reducing time-memory trade-off in passwords [56] even made dictionary attacks much more efficient. Rainbow table [89] further reduces the table number in [56] using multiple reduction functions. However, in recent years as the password policy has become strict, simple dictionary passwords are less common. More powerful attacks are then created. Narayanan and Shmatikov [87] used the Markov model to generate guesses based on the fact that passwords are phonetically similar to users' native languages. OMEN [29] improves [87] to crack passwords by using a more optimal guessing order. Another advanced attack is by using PCFG [121], on which Personal-PCFG is built. Veras et al. [114] tried to leverage semantic patterns in passwords. Besides, while attacking a hashed password database remains the main attacking scenario, there are other attacks on different scenarios, such as video eavesdropping [15].

6.4 Password Manager

Facing the dilemma of not being able to replace passwords, many works focus on helping users manage and remember their passwords, which indirectly enhance password strength due to decreased memorability requirement. In consequence, password manager earns its prosperity. Despite ubiquitous “memorize and fetch” type of password managers such as browser built-in password managers or LastPass, researchers also proposed password managers that can enhance password security in addition to usability [52, 85, 97, 119].

Password manager significantly reduce the memory burden on users. However, it has its own usability and security problems [77]. Severe security issues may also be introduced due to the fact that users failed to capture the correct mental model [31]. Silver et al. [102] demonstrated that careless auto-filling policy on non-https websites could make passwords be extracted directly from the web form by an attacker.

6.5 Enhancing Password Security

Facing the fact that passwords are a vulnerable authentication scheme, alternatives such as graphics-based [37, 60] or biometrics-based [59] authentication methods have been proposed. However, text-based passwords are expected to continue to dominate [21].

Instead of trying to replacing passwords, there are many works focusing on enhancing password security, including password strength feedback [30, 71, 120], multiple factor authentication [25, 94, 99], and security enhancement tools [85, 97, 105, 119]. However, an ideal solution to enhance password security without sacrificing usability, if existed, is yet to be found.

6.6 Multi-factor Authentication

Multi-Factor Authentication enhances authentication security by requiring two or even more factors. Although there are cases that many factors are considered, such as Bank of America account recovery [2], two-factor authentication (2FA) is generally considered as achieving a satisfactory security level. Despite password as one factor, the other factor ranges from additional knowledges [25, 94],

biometrics [61], to hardware tokens [11]. However, enabling 2FA sacrifices usability since it takes considerably more time and effort to complete user authentication. Thus, 2FA has a limited adoption rate [93].

6.7 Password Recovery

As an important component of password authentication, account recovery is increasingly important, especially when users own increasing number of online accounts. Garfinkel [46] proposed Email-based Identification and Authentication (EBIA), which authenticates a user based on the ability to access a certain email address. Although it cannot universally replace password authentication, EBIA has been a primary way for account recovery. Similarly, receiving calls or SMS on cell phones is another de facto recovery scheme. One more previously popular recovery scheme is through security question [63]. However, it has been shown that secret questions are weak [20, 99] because the entropy is low and hence can be easily cracked through guessing or social engineering. Based on large datasets from Google, Bonneau et al. [20] also demonstrated that secret questions have a low recall rate and an easily constructible distribution. They also found that users try to supply fake answers to make the questions harder to answer, which however, yields the opposite outcome.

6.8 User Tracking and UBA

User or user activity tracking has been extensively studied in different contexts and various techniques have been proposed. One typical scenario is web user tracking through different measures [9, 13, 83]. User behavior tracking for the purpose of security drives UBA, where user accounts are no longer the single indicator of who an incident is performed by. Nowadays, many security companies have announced UBA tool integration or plan to develop UBA in their systems [14, 58, 62, 104, 113].

UBA consists of two steps. The first is to model normal user behaviors, and the second is to detect abnormal users by examining how deviated they are from normal users. There can be many metrics, algorithms, or machine learning models being used to identify an abnormal user [101, 104].

Contemporary UBA mostly models users based on basic patterns or statistics, for example, several basic statistics, such as total upload bytes and total download bytes of a user [101].

However, to detect more sophisticated attacks, it is vital to ensure high accuracy and descriptiveness of user activities.

6.9 Log Audit

Log audit has been used in many fields of security research, such as forensics analysis [69, 70, 79], intrusion recovery [49, 68], and intrusion detection [39]. One of the most widely adopted log levels is the OS level, where the basic units are process, files, sockets, etc. The reason is that the OS level maintains high fidelity of states of the entire system, as well as incurring acceptable CPU and storage overhead [69]. There are previous works focusing on the reduction of the storage overhead while not losing much information [75, 122]. Besides, there are also previous works that attempt to increase data granularity based on OS level logs [74, 78].

One important use of log audit is to understand an attack, especially more sophisticated attacks (APT attacks) or unknown attacks. Security experts rely on the logs to determine how an attack happens [69, 74, 79], as well as its impact on the system [70]. They capture the causal relationship among processes, files, or sockets, and reconstruct the provenance of an attack and its ramification. Another broad genre of attack provenance is to use tainting techniques [64, 124]. However, they all rely on an detected attack as its start point so that backtracking and forward tracking are possible to apply from the point. By contrast, our scheme has a different goal. We aim to build the input used as part of the detection system.

HERCULE [92] leverages community discovery algorithms to identify an attack based on the fact that the attack activities belong to the same community in a graph. [17] logs events at the proxy and mainly focuses on parsing traffic from common application protocols, such as SQL and SOAP.

6.10 User Interaction Detection

The detection of bot generated data (system-triggered) from human-generated data (user-triggered) is a long-studied subject that has applications in many fields. Generally there are two types of detection. One is the active detection, such as CAPTCHA [117], which is easy to implement, (arguably) more accurate, but intrusive. The other type is the passive detection. The passive detection relies on processing log events to detect abnormal behaviors. The related previous works include detecting game cheaters through Human Observational Proof [48], bots in online social networks [28,32,35,116], detecting malicious web bots/crawlers, Google reCaptcha [103], and malicious crawler detection [118]. There are some significant differences between these techniques and ours. A major one is that they have specially tailored data input. For example, user agent, cookie lifetime in Google’s reCaptcha [103], a user account favored access log system in [48,118], or side information such as social graph [28,35,48,116]. However, in our setting, there is only a generic-purposed log system, and the events are harder to interpret.

Chapter 7

Conclusion and Future Work

In this dissertation, we present four research projects on enhancing the security of user authentication, particularly, the password-based authentication. In the first project, we perform a study on the use of personal information in passwords, aiming to understand how users create their passwords with their personal information. We develop a novel metric — Coverage — to quantify the correlation between a password and the personal information of a user. Then, we build an efficient password cracker that is capable of generating personalized guesses. Our evaluation shows that our password cracker is much faster than the state-of-the-art crackers with the knowledge of personal information. The password study in this project is fueled up by the availability of numerous large-sized leaked datasets. Although the structure of passwords has been well understood, the study on password semantics still falls short. It is well-known that people from different culture and/or with different native languages behave differently when constructing a password. One future research direction is to collect more data that cover more users from different regions, so that we can better understand how those users differ in choosing passwords. On the other hand, using leaked datasets for research study raises ethical and privacy concerns, no matter how carefully researchers process the data. Therefore, a future research topic could target at how to release password datasets with minimum privacy leakage under different data sharing constraints.

The second project designs and develops BluePass, a distributed password manager that emphasizes to achieve both security and usability. On the security side, it requires a dual possession (i.e., a master password and a mobile device) to access all site passwords with a two-factor authentication.

On the usability side, it requires no user involvement for the authentication via the mobile device, since it is done automatically using the short-ranged Bluetooth communication. In the future, we envision user authentication could become more heterogeneous to compensate the security limits of text-based passwords, especially with the rise of mobile devices that are equipped with a variety of biometric input sources, such as fingerprints, iris, and facial recognition. Biometric-based user authentication has been replacing passwords in certain applications, such as unlocking the smartphone with fingerprints. Thus, mobile-assisted authentication may be a promising direction on user authentication in the future. The powerful mobile devices also provide a template to support continuous user authentication and behavior monitoring.

The third project examines hundreds of top websites to unveil the contemporary implementation of password recovery protocols. One critical observation is that most websites rely on emails to reset the user passwords, which could lead to a potential single point of failure — if an email account is compromised, all other accounts are also likely to be compromised by recovering the account passwords through the email account. We estimate the likelihood and potential damage of such an account recovery attack, and find that password recovery attacks are real and can cause serious damages. Furthermore, many major email providers fail to properly protect users' email accounts. Finally, we propose a security enhancement protocol to secure the password recovery process in a compatible and deployable fashion.

The password serves as a token to pass a gatekeeper, but it cannot guarantee an authenticated user will not perform malicious activities after entering the system. Therefore, it is critical to keep monitoring the user behaviors in order to identify malicious users that manage to conduct certain levels of system breaches. In the fourth project, we design a user tracking system called UTrack that tracks user behaviors with associated identities in an enterprise network system. It is based on OS-level logs to construct complete and concise user profiles inside a network. Beyond one-time password authentication, behavior-based user authenticity monitoring is an important trend. Many security companies start to use UBA to identify suspicious users. The ability to continuously track user behaviors largely complement the one-time nature of password-based authentication. Based on UTrack, a number of end-solutions could be developed. For instance, it can empower intrusion

detection systems on accurately detecting and identifying intruders based on a complete user profile. A user tracking system could also be useful on determining the value of digital assets, which can be done by assessing the amount of time spent on a file by employees. This is particularly useful for fighting ransomware. Furthermore, it can be used to ensure that an employee follows the security policy while operating on sensitive data.

UBA has been introduced only for several years, and it has become a very hot topic in both academia and industry. We believe that there would be more accurate user modeling and more effective detection techniques on various layers. Meanwhile, many questions arise to be answered. For instance, for log-based UBA, it is challenging to collect and fuse data from different sources, since they may have very diverse syntax and semantics. It is also non-trivial to instrument existing software without significant performance degradation. Furthermore, the “needle in a haystack” problem that widely exists in many log systems also needs to be addressed.

Bibliography

- [1] Amazon web service. <https://aws.amazon.com/>.
- [2] Bank of america forgot passcode. <https://secure.bankofamerica.com/login/reset/-entry/forgotPwdScreen.go>.
- [3] Google's downtime caused a 40% drop in global traffic. <https://engineering.gosquared.com/googles-downtime-40-drop-in-traffic>.
- [4] Hold security recovers 272 million stolen credentials from a collector. http://holdsecurity.com/news/the_collector_breach/.
- [5] Lastpass suffers data breach again. <http://www.csoononline.com/article/2936105/data-breach/lastpass-suffers-data-breach-again.html>.
- [6] opendkim. <http://www.opendkim.org/>.
- [7] Postfix. <http://www.postfix.org/>.
- [8] An update on our war against account hijackers. <https://googleblog.blogspot.com/2013/02/an-update-on-our-war-against-account.html>.
- [9] GUNES ACAR, CHRISTIAN EUBANK, STEVEN ENGLEHARDT, MARC JUAREZ, ARVIND NARAYANAN, AND CLAUDIA DIAZ. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM CCS*, 2014.
- [10] FURKAN ALACA AND PC VAN OORSCHOT. Device fingerprinting for augmenting web authentication: classification and analysis of methods. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 289–301. ACM, 2016.
- [11] FADI ALOUL, SYED ZAHIDI, AND WASSIM EL-HAJJ. Two factor authentication using mobile phones. In *AICCSA 2009*. IEEE, 2009.
- [12] ANIMASHREE ANANDKUMAR, CHATSHIK BISDIKIAN, AND DAKSHI AGRAWAL. Tracking in a spaghetti bowl: monitoring transactions using footprints. In *ACM SIGMETRICS Performance Evaluation Review*, volume 36, pages 133–144. ACM, 2008.
- [13] RICHARD ATTERER, MONIKA WNUK, AND ALBRECHT SCHMIDT. Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. In *WWW*, 2006.

- [14] BALABIT. Privileged account analytics - user behavior analytics security solution. <https://www.balabit.com/privileged-account-analytics>, 2015.
- [15] DAVIDE BALZAROTTI, MARCO COVA, AND GIOVANNI VIGNA. Clearshot: Eavesdropping on keyboard input from video. In *IEEE Security & Privacy*, 2008.
- [16] PAUL BARHAM, AUSTIN DONNELLY, REBECCA ISAACS, AND RICHARD MORTIER. Using magpie for request extraction and workload modelling. In *OSDI*, 2004.
- [17] ADAM BATES, WAJIB UL HASSAN, KEVIN BUTLER, ALIN DOBRA, BRADLEY REAVES, PATRICK CABLE, THOMAS MOYER, AND NABIL SCHEAR. Transparent web service auditing via network provenance functions. In *Proceedings of the 26th International Conference on World Wide Web*, pages 887–895, 2017.
- [18] ADAM BEAUTEMENT, M ANGELA SASSE, AND MIKE WONHAM. The compliance budget: managing security behaviour in organisations. In *NSPW*. ACM, 2008.
- [19] JOSEPH BONNEAU. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *IEEE Security & Privacy*, 2012.
- [20] JOSEPH BONNEAU, ELIE BURSZTEIN, ILAN CARON, ROB JACKSON, AND MIKE WILLIAMSON. Secrets, lies, and account recovery: Lessons from the use of personal knowledge questions at google. In *WWW*, 2015.
- [21] JOSEPH BONNEAU, CORMAC HERLEY, PAUL C VAN OORSCHOT, AND FRANK STAJANO. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *IEEE Security & Privacy*, 2012.
- [22] JOSEPH BONNEAU, CORMAC HERLEY, PAUL C VAN OORSCHOT, AND FRANK STAJANO. Passwords and the evolution of imperfect authentication. *Communications of the ACM*, 2015.
- [23] JOSEPH BONNEAU, SÖREN PREIBUSCH, AND ROSS ANDERSON. A birthday present every eleven wallets? the security of customer-chosen banking pins. In *Financial Cryptography and Data Security*. Springer, 2012.
- [24] S BOZTAS. Entropies, guessing, and cryptography. *Department of Mathematics, Royal Melbourne Institute of Technology, Tech. Rep*, 1999.
- [25] JOHN BRAINARD, ARI JUELS, RONALD L RIVEST, MICHAEL SZYDLO, AND MOTI YUNG. Fourth-factor authentication: somebody you know. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 168–178. ACM, 2006.
- [26] CHRISTIAN CACHIN. *Entropy measures and unconditional security in cryptography*. PhD thesis, Swiss Federal Institute of Technology Zurich, 1997.
- [27] PHUONG CAO, HONGYANG LI, KLARA NAHRSTEDT, ZBIGNIEW KALBARCZYK, RAVISHANKAR IYER, AND ADAM J SLAGELL. Personalized password guessing: a new security threat. In *ACM Proceedings of the 2014 Symposium and Bootcamp on the Science of Security*, 2014.

- [28] QIANG CAO, MICHAEL SIRIVIANOS, XIAOWEI YANG, AND TIAGO PREGUEIRO. Aiding the detection of fake accounts in large scale social online services. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 15–15. USENIX Association, 2012.
- [29] CLAUDE CASTELLUCCIA, ABDELBERI CHAABANE, MARKUS DÜRMUTH, AND DANIELE PERITO. When privacy meets security: Leveraging personal information for password cracking. *arXiv preprint arXiv:1304.6584*, 2013.
- [30] CLAUDE CASTELLUCCIA, MARKUS DÜRMUTH, AND DANIELE PERITO. Adaptive password-strength meters from markov models. In *NDSS*, 2012.
- [31] SONIA CHIASSON, PAUL C VAN OORSCHOT, AND ROBERT BIDDLE. A usability study and critique of two password managers. In *Usenix Security*, 2006.
- [32] ZI CHU, STEVEN GIANVECCHIO, HAINING WANG, AND SUSHIL JAJODIA. Who is tweeting on twitter: human, bot, or cyborg? In *Proceedings of the 26th annual computer security applications conference*, pages 21–30. ACM, 2010.
- [33] DAVE CROCKER, TONY HANSEN, AND MURRAY KUCHERAWY. Domainkeys identified mail (dkim) signatures. no. rfc 6376. Technical report, 2011.
- [34] ALEXEI CZESKIS, MICHAEL DIETZ, TADAYOSHI KOHNO, DAN WALLACH, AND DIRK BALFANZ. Strengthening user authentication through opportunistic cryptographic identity assertions. In *ACM CCS*, 2012.
- [35] GEORGE DANEZIS AND PRATEEK MITTAL. Sybilinfer: Detecting sybil nodes using social networks. In *NDSS*. San Diego, CA, 2009.
- [36] ANUPAM DAS, JOSEPH BONNEAU, MATTHEW CAESAR, NIKITA BORISOV, AND XIAOFENG WANG. The tangled web of password reuse. In *NDSS*, 2014.
- [37] DARREN DAVIS, FABIAN MONROSE, AND MICHAEL K REITER. On user choice in graphical password schemes. In *USENIX Security*, 2004.
- [38] X DE CARNÉ DE CARNAVALET AND MOHAMMAD MANNAN. From very weak to very strong: Analyzing password-strength meters. In *NDSS*, 2014.
- [39] DOROTHY E DENNING. An intrusion-detection model. *IEEE Transactions on software engineering*, (2):222–232, 1987.
- [40] DAVID DEVECSERY, MICHAEL CHOW, XIANZHENG DOU, JASON FLINN, AND PETER M CHEN. Eidetic systems. In *USENIX OSDI*, pages 525–540, 2014.
- [41] ZAKIR DURUMERIC, DAVID ADRIAN, ARIANA MIRIAN, JAMES KASTEN, ELIE BURSZTEIN, NICOLAS LIDZBORSKI, KURT THOMAS, VIJAY ERANTI, MICHAEL BAILEY, AND J ALEX HALDERMAN. Neither snow nor rain nor mitm...: An empirical analysis of email delivery security. In *ACM IMC*, 2015.
- [42] SERGE EGELMAN, SAKSHI JAIN, REBECCA S PORTNOFF, KERWELL LIAO, SUNNY CONSOLVO, AND DAVID WAGNER. Are you ready to lock? In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 750–761. ACM, 2014.

- [43] SERGE EGELMAN, ANDREAS SOTIRAKOPOULOS, ILDAR MUSLUKHOV, KONSTANTIN BEZNOSOV, AND CORMAC HERLEY. Does my password go up to eleven?: the impact of password meters on password selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2379–2388. ACM, 2013.
- [44] DINEI FLORENCIO AND CORMAC HERLEY. A large-scale study of web password habits. In *ACM WWW*, 2007.
- [45] CHERRY-PY. A MINIMALIST PYTHON WEB FRAMEWORK. <http://www.cherrypy.org/>, 2016.
- [46] SIMSON L GARFINKEL. Email-based identification and authentication: An alternative to pki? *IEEE Security & Privacy Magazine*, 2003.
- [47] STEVEN GIANVECCHIO AND HAINING WANG. Detecting covert timing channels: an entropy-based approach. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pages 307–316, 2007.
- [48] STEVEN GIANVECCHIO, ZHENYU WU, MENGJUN XIE, AND HAINING WANG. Battle of botcraft: fighting bots in online games with human observational proofs. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 256–268. ACM, 2009.
- [49] ASHVIN GOEL, KENNETH PO, KAMRAN FARHADI, ZHENG LI, AND EYAL DE LARA. The taser intrusion recovery system. In *SOSP*, volume 39, pages 163–176. ACM, 2005.
- [50] RALPH GROSS AND ALESSANDRO ACQUISTI. Information revelation and privacy in online social networks. In *ACM WPES*, 2005.
- [51] ERIC GROSSE AND MAYANK UPADHYAY. Authentication at scale. *IEEE Security & Privacy Magazine*, 11(1):15–22, 2013.
- [52] J ALEX HALDERMAN, BRENT WATERS, AND EDWARD W FELTEN. A convenient method for securely managing passwords. In *WWW*. ACM, 2005.
- [53] MARK HALL, EIBE FRANK, GEOFFREY HOLMES, BERNHARD PFAHRINGER, PETER REUTEMANN, AND IAN H WITTEN. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 2009.
- [54] JIAWEI HAN, JIAN PEI, AND YIWEN YIN. Mining frequent patterns without candidate generation. In *ACM SIGMOD record*, volume 29, pages 1–12, 2000.
- [55] WAJIB UL HASSAN, MARK LEMAY, NURAINI AGUSE, ADAM BATES, AND THOMAS MOYER. Towards scalable cluster auditing through grammatical inference over provenance graphs. In *NDSS*, 2018.
- [56] MARTIN E HELLMAN. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 1980.
- [57] ADELE E HOWE, INDRAJIT RAY, MARK ROBERTS, MALGORZATA URBANSKA, AND ZINTA BYRNE. The psychology of security for the home computer user. In *IEEE Security & Privacy*, 2012.

- [58] IBM. Ibm qradar user behavior analytics. <https://www.ibm.com/cz-en/marketplace/qradar-user-behavior-analytics>, 2016.
- [59] ANIL K JAIN, ARUN ROSS, AND SHARATH PANKANTI. Biometrics: a tool for information security. *IEEE Transactions on Information Forensics and Security*, 2006.
- [60] IAN JERMYN, ALAIN J MAYER, FABIAN MONROSE, MICHAEL K REITER, AVIEL D RUBIN, ET AL. The design and analysis of graphical passwords. In *Usenix Security*, 1999.
- [61] ANDREW TEOH BENG JIN, DAVID NGO CHEK LING, AND ALWYN GOH. Biohashing: two factor authentication featuring fingerprint data and tokenised random number. *Pattern recognition*, 2004.
- [62] JOHNA TILL JOHNSONS. User behavioral analytics tools can thwart security attacks, 2015.
- [63] MIKE JUST. Designing and evaluating challenge-question systems. *IEEE Security & Privacy Magazine*, (5):32–39, 2004.
- [64] MIN GYUNG KANG, STEPHEN MCCAMANT, PONGSIN POOSANKAM, AND DAWN SONG. Dta++: dynamic taint analysis with targeted control-flow propagation. In *NDSS*, 2011.
- [65] AMBARISH KAROLE, NITESH SAXENA, AND NICOLAS CHRISTIN. A comparative usability evaluation of traditional password managers. In *ICISC 2010*. Springer, 2010.
- [66] CHARLIE KAUFMAN, RADIA PERLMAN, AND MIKE SPECINER. *Network Security: Private Communication in a Public World*. Prentice-Hall, Inc., 1995.
- [67] PATRICK GAGE KELLEY, SARANGA KOMANDURI, MICHELLE L MAZUREK, RICHARD SHAY, TIMOTHY VIDAS, LUJO BAUER, NICOLAS CHRISTIN, LORRIE FAITH CRANOR, AND JULIO LOPEZ. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *IEEE Security & Privacy*, 2012.
- [68] TAESOO KIM, XI WANG, NICKOLAI ZELDOVICH, M FRANS KAASHOEK, ET AL. Intrusion recovery using selective re-execution. In *OSDI*, pages 89–104, 2010.
- [69] SAMUEL T KING AND PETER M CHEN. Backtracking intrusions. *SOSP*, 37(5):223–236, 2003.
- [70] SAMUEL T KING, ZHUOQING MORLEY MAO, DOMINIC G LUCCHETTI, AND PETER M CHEN. Enriching intrusion alerts through multi-host causality. In *NDSS*, 2005.
- [71] SARANGA KOMANDURI, RICHARD SHAY, LORRIE FAITH CRANOR, CORMAC HERLEY, AND STUART SCHECHTER. Telepathwords: Preventing weak passwords by reading users’ minds. In *USENIX Security*, 2014.
- [72] BALACHANDER KRISHNAMURTHY AND CRAIG E WILLS. On the leakage of personally identifiable information via online social networks. In *ACM COSN*, 2009.
- [73] CYNTHIA KUO, SASHA ROMANOSKY, AND LORRIE FAITH CRANOR. Human selection of mnemonic phrase-based passwords. In *ACM SOUPS*, 2006.

- [74] KYU HYUNG LEE, XIANGYU ZHANG, AND DONGYAN XU. High accuracy attack provenance via binary-based execution partition. In *NDSS*, 2013.
- [75] KYU HYUNG LEE, XIANGYU ZHANG, AND DONGYAN XU. Loggc: garbage collecting audit log. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1005–1016. ACM, 2013.
- [76] ZHIGONG LI, WEILI HAN, AND WENYUAN XU. A large-scale empirical analysis of chinese web passwords. In *Proc. USENIX Security*, 2014.
- [77] ZHIWEI LI, WARREN HE, DEVDATTA AKHAWA, AND DAWN SONG. The emperor’s new password manager: Security analysis of web-based password managers. In *USENIX Security*, 2014.
- [78] SHIQING MA, KYU HYUNG LEE, CHUNG HWAN KIM, JUNGHWAN RHEE, XIANGYU ZHANG, AND DONGYAN XU. Accurate, low cost and instrumentation-free security audit logging for windows. In *Proceedings of the 31st Annual Computer Security Applications Conference*, pages 401–410. ACM, 2015.
- [79] SHIQING MA, XIANGYU ZHANG, AND DONGYAN XU. Protracer: towards practical provenance tracing by alternating between logging and tainting. In *Proceedings of NDSS*, volume 16, 2016.
- [80] THE MUTT E MAIL CLIENT. <http://www.mutt.org/>, 2019.
- [81] DAVID MALONE AND KEVIN MAHER. Investigating the distribution of password choices. In *ACM WWW*, 2012.
- [82] JAMES L MASSEY. Guessing and entropy. In *IEEE International Symposium on Information Theory*, 1994.
- [83] JONATHAN R MAYER AND JOHN C MITCHELL. Third-party web tracking: Policy and technology. In *IEEE S&P 2012*, pages 413–427, 2012.
- [84] MICHELLE L MAZUREK, SARANGA KOMANDURI, TIMOTHY VIDAS, LUJO BAUER, NICOLAS CHRISTIN, LORRIE FAITH CRANOR, PATRICK GAGE KELLEY, RICHARD SHAY, AND BLASE UR. Measuring password guessability for an entire university. In *ACM CCS*, 2013.
- [85] DANIEL MCCARNEY, DAVID BARRERA, JEREMY CLARK, SONIA CHIASSON, AND PAUL C VAN OORSCHOT. Tapas: design, implementation, and usability evaluation of a password manager. In *ACSAC*. ACM, 2012.
- [86] ROBERT MORRIS AND KEN THOMPSON. Password security: A case history. *Communications of the ACM*, 1979.
- [87] ARVIND NARAYANAN AND VITALY SHMATIKOV. Fast dictionary attacks on passwords using time-space tradeoff. In *ACM CCS*, 2005.
- [88] CLAUDIO SORIENTE NIKOLAOS KARAPANOS, CLAUDIO MARFORIO AND SRDJAN CAPKUN. Sound-proof: Usable two-factor authentication based on ambient sound. In *Proc. USENIX Security*, 2015.

- [89] PHILIPPE OECHSLIN. Making a faster cryptanalytic time-memory trade-off. In *Advances in Cryptology-CRYPTO 2003*. 2003.
- [90] HOW OFTEN DO YOU NEED TO CHARGE YOUR SMARTPHONE? <http://lifel hacker.com/how-often-do-you-need-to-charge-your-smartphone-1441051270>, 2016.
- [91] BRYAN PARNO, CYNTHIA KUO, AND ADRIAN PERRIG. *Phoolproof phishing prevention*. Springer, 2006.
- [92] KEXIN PEI, ZHONGSHU GU, BRENDAN SALTAFORMAGGIO, SHIQING MA, FEI WANG, ZHIWEI ZHANG, LUO SI, XIANGYU ZHANG, AND DONGYAN XU. Hercule: Attack story reconstruction via community discovery on correlated log graph. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 583–595. ACM, 2016.
- [93] THANASIS PETSAS, GIORGOS TSIRANTONAKIS, ELIAS ATHANASOPOULOS, AND SOTIRIS IOANNIDIS. Two-factor authentication: is the world ready?: quantifying 2fa adoption. In *Proceedings of the Eighth European Workshop on System Security*, page 4. ACM, 2015.
- [94] BENNY PINKAS AND TOMAS SANDER. Securing passwords against dictionary attacks. In *ACM CCS*, 2002.
- [95] JOHN O PLIAM. On the incomparability of entropy and marginal guesswork in brute-force attacks. In *Progress in Cryptology-INDOCRYPT*. 2000.
- [96] PATRICK REYNOLDS, JANET L WIENER, JEFFREY C MOGUL, MARCOS K AGUILERA, AND AMIN VAHDAT. Wap5: black-box performance debugging for wide-area systems. In *Proceedings of the 15th international conference on World Wide Web*, pages 347–356. ACM, 2006.
- [97] BLAKE ROSS, COLLIN JACKSON, NICK MIYAKE, DAN BONEH, AND JOHN C MITCHELL. Stronger password authentication using browser extensions. In *Usenix security*, 2005.
- [98] BO SANG, JIANFENG ZHAN, GANG LU, HAINING WANG, DONGYAN XU, LEI WANG, ZHIHONG ZHANG, AND ZHEN JIA. Precise, scalable, and online request tracing for multitier services of black boxes. *IEEE Transactions on Parallel and Distributed Systems*, 23(6):1159–1167, 2012.
- [99] STUART SCHECHTER, SERGE EGELMAN, ET AL. It’s no secret. measuring the security and reliability of authentication via “secret” questions. In *IEEE Security & Privacy*, 2009.
- [100] DINO SCHWEITZER, JEFF BOLENG, COLIN HUGHES, AND LOUIS MURPHY. Visualizing keyboard pattern passwords. In *IEEE VizSec*, 2009.
- [101] MADHU SHASHANKA, MIN-YI SHEN, AND JISHENG WANG. User and entity behavior analytics for enterprise security. In *2016 IEEE Big Data*, pages 1867–1874, 2016.
- [102] DAVID SILVER, SUMAN JANA, ERIC CHEN, COLLIN JACKSON, AND DAN BONEH. Password managers: Attacks and defenses. In *Usenix Security*, 2014.
- [103] SUPHANNEE SIVAKORN, JASON POLAKIS, AND ANGELOS D KEROMYTIS. I’m not a human: Breaking the google recaptcha. *Black Hat, (i)*, pages 1–12, 2016.

- [104] SPLUNK. Splunk user behavior analytics, 2015.
- [105] BENJAMIN STRAHS, CHUAN YUE, AND HAINING WANG. Secure passwords through enhanced hashing. In *USENIX LISA*, 2009.
- [106] BYUNG-CHUL TAK, CHUNQIANG TANG, CHUN ZHANG, SRIRAM GOVINDAN, BHUVAN URGONKAR, AND RONG N CHANG. vpath: Precise discovery of request processing paths from black-box observations of thread and network activities. In *USENIX ATC*, 2009.
- [107] ALEXA THE TOP 500 SITES ON THE WEB. <http://www.alexa.com/topsites>, 2016.
- [108] ENO THERESKA, BRANDON SALMON, JOHN STRUNK, MATTHEW WACHS, MICHAEL ABDEL-MALEK, JULIO LOPEZ, AND GREGORY R GANGER. Stardust: tracking activity in a distributed storage system. In *ACM SIGMETRICS Performance Evaluation Review*, volume 34, pages 3–14. ACM, 2006.
- [109] ROY HODGMAN TOD BEARDSLEY. Rapid 7 research report: Understanding user behavior analytics, 2015.
- [110] TRUSTWAVE. Trustwave global security report. https://www2.trustwave.com/rs/815-RFM-693/images/2015_TrustwaveGlobalSecurityReport.pdf, 2015.
- [111] MELISSA TURCOTTE AND JUSTON SHANE MOORE. Technical report la-ur-17-21663: User behavior analytics, 2017.
- [112] BLASE UR, PATRICK GAGE KELLEY, SARANGA KOMANDURI, JOEL LEE, MICHAEL MAASS, MICHELLE L MAZUREK, TIMOTHY PASSARO, RICHARD SHAY, TIMOTHY VIDAS, LUJO BAUER, ET AL. How does your password measure up? the effect of strength meters on password creation. In *The 21st USENIX Security Symposium*, pages 65–80, 2012.
- [113] VARONIS. User behavior analytics. <https://www.varonis.com/user-behavior-analytics/>, 2016.
- [114] RAFAEL VERAS, CHRISTOPHER COLLINS, AND JULIE THORPE. On the semantic patterns of passwords and their security impact. In *NDSS*, 2014.
- [115] RAFAEL VERAS, JULIE THORPE, AND CHRISTOPHER COLLINS. Visualizing semantics in passwords: The role of dates. In *IEEE VizSec*, 2012.
- [116] BIMAL VISWANATH, ANSLEY POST, KRISHNA P GUMMADI, AND ALAN MISLOVE. An analysis of social network-based sybil defenses. *ACM SIGCOMM Computer Communication Review*, 40(4):363–374, 2010.
- [117] LUIS VON AHN, BENJAMIN MAURER, COLIN McMILLEN, DAVID ABRAHAM, AND MANUEL BLUM. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.
- [118] SHENGYE WAN, YUE LI, AND KUN SUN. Protecting web contents against persistent distributed crawlers. In *IEEE ICC*, 2017.
- [119] LUREN WANG, YUE LI, AND KUN SUN. Amnesia: A bilateral generative password manager. In *ICDCS*, 2016.

- [120] MATT WEIR, SUDHIR AGGARWAL, MICHAEL COLLINS, AND HENRY STERN. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *ACM CCS*, 2010.
- [121] MATT WEIR, SUDHIR AGGARWAL, BRENO DE MEDEIROS, AND BILL GLODEK. Password cracking using probabilistic context-free grammars. In *IEEE Security & Privacy*, 2009.
- [122] ZHANG XU, ZHENYU WU, ZHICHUN LI, KANGKOOK JEE, JUNGHWAN RHEE, XUSHENG XIAO, FENGYUAN XU, HAINING WANG, AND GUOFEI JIANG. High fidelity data reduction for big data security dependency analyses. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 504–516. ACM, 2016.
- [123] JEFF YAN, ALAN BLACKWELL, ROSS ANDERSON, AND ALASDAIR GRANT. Password memorability and security: Empirical results. *IEEE Security & Privacy Magazine*, 2004.
- [124] HENG YIN, DAWN SONG, MANUEL EGELE, CHRISTOPHER KRUEGEL, AND ENGIN KIRDA. Panorama: capturing system-wide information flow for malware detection and analysis. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 116–127. ACM, 2007.