

Supporting Password-Security Decisions with Data

Blase Ur

CMU-ISR-16-110

September 2016

School of Computer Science
Institute for Software Research
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:

Lorrie Faith Cranor, Chair
Alessandro Acquisti
Lujo Bauer
Jason Hong
Michael K. Reiter, UNC Chapel Hill

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy*

© 2016 Blase Ur

The research reported in this thesis has been supported in part by NSF grants DGE-0903659 and CNS-1116776, as well as gifts from the PNC Center for Financial Services Innovation and Microsoft Research. It has also been supported by CyLab at Carnegie Mellon under grants DAAD19-02-1-0389 and W911NF-09-1-0273 from the Army Research Office, by Air Force Research Lab Award No. FA87501220139, and by the Facebook graduate fellowship program. In addition, it was conducted with government support awarded by DoD, Air Force OSR, via the NDSEG Fellowship, 32 CFR 168a.

Keywords: Usable security, computer security, passwords, authentication

Abstract

Despite decades of research into developing abstract security advice and improving interfaces, users still struggle to make passwords. Users frequently create passwords that are predictable for attackers or make other decisions (e.g., reusing the same password across accounts) that harm their security. In this thesis, I use data-driven methods to better understand how users choose passwords and how attackers guess passwords. I then combine these insights into a better password-strength meter that provides real-time, data-driven feedback about the user's candidate password.

I first quantify the impact on password security and usability of showing users different password-strength meters that score passwords using basic heuristics. I find in a 2,931-participant online study that meters that score passwords stringently and present their strength estimates visually lead users to create stronger passwords without significantly impacting password memorability. Second, to better understand how attackers guess passwords, I perform comprehensive experiments on password-cracking approaches. I find that simply running these approaches in their default configuration is insufficient, but considering multiple well-configured approaches in parallel can serve as a proxy for guessing by an expert in password forensics. The third and fourth sections of this thesis delve further into how users choose passwords. Through a series of analyses, I pinpoint ways in which users structure semantically significant content in their passwords. I also examine the relationship between users' perceptions of password security and passwords' actual security, finding that while users often correctly judge the security impact of individual password characteristics, wide variance in their understanding of attackers may lead users to judge predictable passwords as sufficiently strong. Finally, I integrate these insights into an open-source password-strength meter that gives users data-driven feedback about their specific password. I evaluate this meter through a ten-participant laboratory study and 4,509-participant online study.

Thesis statement: **The objective of this thesis is to demonstrate how integrating data-driven insights about how users create and how attackers guess passwords into a tool that presents real-time feedback can equip users to make better passwords.**

Acknowledgments

I am deeply indebted to my advisor (Lorrie Cranor), my common-law co-advisor (Lujo Bauer), my other committee members (Alessandro Acquisti, Jason Hong, and Mike Reiter), and my other frequent faculty collaborators (Nicolas Christin and Michael Littman) for guiding my growth as a researcher over the last five years. I particularly want to thank Lorrie for constantly pushing me, always giving very helpful and detailed feedback, and usually inviting me to the kids' guitar recitals.

In addition, I want to thank my teachers from throughout my life, particularly Andy DeNicola, who taught me about resilience and hard work. I am indebted to my prior mentors (Vinod Ganapathy, Rachna Dhamija, and Mike Smith) for setting me on the path to grad school, which I never would have considered otherwise.

I am very grateful for having the opportunity to work with many awesome collaborators over the past few years. My time at CMU has been wonderful because of the people here at Cylab and Societal Computing. For the projects I describe in this thesis, I was fortunate to have excellent collaborators: Felicia Alfieri, Maung Aung, Lujo Bauer, Jonathan Bees, Nicolas Christin, Jessica Colnago, Lorrie Faith Cranor, Henry Dixon, Adam L. Durity, Pardis Emami Naeini, Hana Habib, Phillip (Seyoung) Huh, Noah Johnson, Patrick Gage Kelley, Saranga Komanduri, Darya Kurilova, Joel Lee, Michael Maass, Stephanos Matsumoto, Michelle L. Mazurek, William Melicher, Fumiko Noma, Timothy Passaro, Sean M. Segreti, Richard Shay, and Timothy Vidas. I have learned a huge amount from all of you, and I hope I didn't drive you too crazy in my quest to "Make America Late Again" in the weeks before deadlines. Outside of the lab, I want to thank Carol Frieze, Mary Widom, Kenny Joseph, Manya Sleeper, and everyone else from SCS4ALL, Women@SCS, and BiasBusters. I especially want to thank my awesome SciTech students, some of whom I hope will continue to think computationally.

Had my life over the past five years not continued past the edge of CMU's campus, this thesis never would have been written. For that, I am very grateful to the awesome friends in my life. I wish to thank my wonderful Cedarhousemates, particularly Hillary, for keeping me sane. My Pittsburgh musical collaborators (Molly, Mike, and Chase), the maintainers of the Pittsburgh-area bike trails, and the heroes/villains of greater Bloomfield have made life in Pittsburgh far more tolerable over the last five years. Particular thanks go to Thai Gourmet for their vegetarian-friendly food, convenient location, and sensible prices.

I don't know where I would be without the support of Roxi and Barbie. I am very grateful to my extended family for the support and the shenanigans. Particular thanks go to my sister (Marissa), who will always be my favorite person, as well as the only reasonable one.

This thesis contains text from a number of prior papers [130, 181, 182, 183, 184, 185]. Additional text written for this thesis will be used in subsequent publications.

Contents

1	Introduction	1
2	Related Work	5
2.1	Threats to Password Security	5
2.2	Password-Security Metrics	7
2.3	Types of Guessing Attacks	8
2.3.1	Brute-Force and Mask Attacks	8
2.3.2	Probabilistic Context-Free Grammar	8
2.3.3	Markov Models	9
2.3.4	Neural Networks	9
2.3.5	Mangled Wordlist Attacks	10
2.4	How Users Choose Passwords	10
2.4.1	Structural Characteristics of Human-Chosen Passwords	10
2.4.2	Linguistic and Semantic Properties of Passwords	11
2.4.3	Password Management and Reuse	11
2.5	Helping Users Create Better Passwords	12
2.5.1	Password-Composition Policies	12
2.5.2	Proactive Password Checking	13
2.5.3	Password Meters	13
2.6	Users' Perceptions of Security	14
2.7	Data-Driven Feedback	14
3	The Impact of Password-Strength Meters	17
3.1	Introduction	17
3.2	Password Meters "In the Wild"	18
3.3	Methodology	19
3.3.1	Password-Scoring Algorithms	19
3.3.2	Conditions	21
3.3.3	Mechanical Turk	22
3.3.4	Statistical Tests	22
3.3.5	Calculating Guess Numbers	23
3.4	Results	23
3.4.1	Password Characteristics	24

3.4.2	Password Guessability	25
3.4.3	Password Memorability and Storage	26
3.4.4	Password Creation Process	27
3.4.5	Participant Demographics	30
3.5	Participants' Attitudes and Perceptions	30
3.5.1	Attitudes Toward Password Meters	31
3.5.2	Participant Motivations	31
3.6	Discussion	34
3.6.1	Effective Password Meters	34
3.6.2	Ethical Considerations	35
3.6.3	Limitations	35
3.7	Conclusion	36
4	Understanding Biases in Modeling Password Cracking	37
4.1	Introduction	37
4.2	Methodology	39
4.2.1	Datasets	39
4.2.2	Training Data	40
4.2.3	Simulating Password Cracking	40
4.2.4	Computational Limitations	43
4.3	Results	44
4.3.1	The Importance of Configuration	44
4.3.2	Comparison of Guessing Approaches	46
4.3.3	Differences Across Approaches	50
4.3.4	Robustness of Analyses to Approach	53
4.4	Supplementary Experimental Results	56
4.4.1	Alternate PCFG Configurations	57
4.4.2	Alternate JTR Configurations	58
4.4.3	Alternate Hashcat Configurations	59
4.4.4	Ecological Validity	60
4.5	Conclusion	60
5	The Art of Password Creation: Semantics and Strategies	63
5.1	Introduction	63
5.2	Datasets	64
5.3	Methodology	65
5.3.1	Reverse Engineering Passwords	65
5.3.2	Semantic Analyses	66
5.3.3	The Process of Password Creation	67
5.3.4	Security Analysis	68
5.4	Results	68
5.4.1	General Password Characteristics	68
5.4.2	Character Substitutions in Passwords	69
5.4.3	Passwords Semantics	71

5.4.4	The Process of Password Creation	74
5.5	Design Recommendations and Conclusions	77
6	Do Users' Perceptions of Password Security Match Reality?	79
6.1	Introduction	79
6.2	Methodology	80
6.2.1	Study Structure	80
6.2.2	Recruitment	82
6.2.3	Measuring Real-World Attacks on Passwords	82
6.2.4	Quantitative Analysis	83
6.2.5	Qualitative Analysis	83
6.2.6	Limitations	83
6.3	Results	84
6.3.1	Participants	84
6.3.2	Attacker Model	84
6.3.3	Password Pairs	88
6.3.4	Selected-Password Analysis	92
6.3.5	Password-Creation Strategies	92
6.4	Discussion	93
7	Design and Evaluation of a Data-Driven Password Meter	95
7.1	Introduction	95
7.2	Measuring Password Strength in our Data-Driven Meter	96
7.2.1	The Difficulty of Accurate Client-Side Password-Strength Estimation	96
7.2.2	Neural Networks for Password-Strength Estimation	97
7.2.3	Advanced Heuristics for Password-Strength Estimation	98
7.3	Visual Design and User Experience	103
7.3.1	Translating Scores to a Visual Bar	104
7.3.2	Main Screen	104
7.3.3	Specific-Feedback Modal	108
7.3.4	Generic-Advice Modal	108
7.4	Formative Laboratory Study	110
7.4.1	Methodology	111
7.4.2	Results	112
7.4.3	Takeaways and Changes to the Meter	113
7.5	Summative Online Study	114
7.5.1	Methodology	114
7.5.2	Limitations	118
7.5.3	Participants	119
7.5.4	Security Impact	119
7.5.5	Usability Impact	125
7.5.6	Interface Element Usage and Reactions	127
7.6	Discussion	131

8	Conclusion and Future Work	133
8.1	Conclusions and Lessons Learned	133
8.1.1	Users' Well-Enscenced Approaches	134
8.1.2	The Mismatched Incentives of Professional Password Advice	134
8.1.3	Mismatch Between Reality and Perception	135
8.1.4	Real-World Considerations in Modeling Attackers	135
8.2	Future Work	136
8.2.1	Improve the Ecosystem	136
8.2.2	Evaluating The Role of Password-Composition Policies	137
8.2.3	Natural Language and Passwords	137
8.2.4	Automatically Modeling Targeted Attacks	137
8.2.5	Improvements to the Meter	138
	Appendices	155
A	Surveys from "How Does Your Password Measure Up..."	157
A.1	Survey Questions	157
A.1.1	Day 1 Survey	157
A.1.2	Day 2 Survey	159
B	Surveys and Full Data from "Do Users' Perceptions..."	163
B.1	Survey Questions	163
B.1.1	Part 1	163
B.1.2	Part 2 (Repeated 26 times)	163
B.1.3	Part 3 (Repeated 20 times)	164
B.1.4	Part 4 (Repeated 11 times)	164
B.1.5	Part 5	164
B.2	Full List of Password Pairs	165
C	Scripts/Surveys from "...Data-Driven Password Meter..."	169
C.1	Script for Laboratory Study	169
C.1.1	Introduction	169
C.1.2	Part I – Open, Abstract Discussion of Password Creation	169
C.1.3	Part II – Password Feedback Slideshow	170
C.1.4	Part III – Meter Testing	170
C.1.5	Part IV – Ending	172
C.2	Surveys from Online Study	172
C.2.1	Day 1 Survey	172
C.2.2	Day 2 Survey	175
D	Data-Driven Meter Details	177
D.1	Prioritization of Feedback from Advanced Heuristics	177
D.2	Description of Open-Source Meter Files	178
D.2.1	Files for Displaying the Meter	178

D.2.2	Files Related to Calculating Heuristic Scoring	178
D.2.3	Files Related to Wordlists	179
D.2.4	Files Related to Neural Networks	179

List of Figures

3.1	Password-strength indicators in the wild	18
3.2	The password creation page participants saw	20
3.3	Password guessability by condition	27
3.4	Participants' sentiment results by condition	32
4.1	Hashcat's guessing accuracy using different mangling rules	45
4.2	Automated approaches' successing guessing the different password sets	47
4.3	The overlap between approaches in successful guessing	48
4.4	Complex _{pilot} guessability by trial	49
4.5	Number of automated approaches that guessed a particular password	50
4.6	Guessing accuracy by # character classes for Basic passwords	51
4.7	Accuracy guessing passwords composed entirely of lowercase letters	51
4.8	Accuracy guessing passwords composed entirely of digits	52
4.9	Pros' comparative success guessing each password set	54
4.10	Guessability of all four password sets under Min _{auto}	54
4.11	Relative guessability of password sets using individual automated approaches	55
4.12	Differences in the order of magnitude of guess numbers across approaches	56
4.13	Comparative guessing accuracy of PCFG configurations	57
4.14	Comparative guessing accuracy of JTR rules	58
4.15	Comparative guessing accuracy of Hashcat rules	59
4.16	Comparison of guessing accuracy for Basic, Basic _{rockyou} , and Basic _{yahoo} passwords	61
4.17	Guessing accuracy by # character classes for Basic _{rockyou} and Basic _{yahoo} passwords	61
6.1	Example task for password pairs	80
6.2	Example task for the selected-password analysis	81
6.3	Security and memorability mean ratings for password-creation strategies	94
7.1	Main screen of the data-driven meter prior to meeting requirements	105
7.2	Main screen of the data-driven meter	106
7.3	Example suggested improvements	107
7.4	Modal window providing feedback on a specific password	109
7.5	The generic-advice modal window	110
7.6	Security impact of composition policy and meter	121

7.7 Security impact of 1class8 feedback 122

7.8 Security impact of 3class12 feedback 123

7.9 Security impact of 3class12 stringency 123

A.1 Example meter shown to participants. 158

C.1 Slideshow of example meters 171

C.2 Example colored bar shown to participants. 172

C.3 Example text feedback shown to participants. 173

C.4 Example suggested improvement shown to participants. 174

List of Tables

3.1	Password characteristics by condition	24
3.2	Password guessability at different thresholds by condition	26
3.3	Password-creation characteristics by condition	28
4.1	Password characteristics per set	40
4.2	Four trials of pros guessing Complex _{pilot} passwords	49
5.1	Password length and character class usage by set	69
5.2	The occurrence of character substitutions in passwords	70
5.3	The 20 most frequent substitution mappings	70
5.4	Dictionaries in which alphabetic chunks appeared	72
5.5	The 15 most common Wikipedia categories used in passwords	73
5.6	The natural-language corpora in which n-grams appeared	73
5.7	How words and phrases appeared in passwords	74
5.8	Security impact of complying with a password-composition policy	74
5.9	The impact of modifying a password to comply with a composition policy	75
5.10	Comparison of the character classes the meter suggested adding and those the user added	76
6.1	Perceptions of the types of attackers	85
6.2	Perceptions of why attackers try to guess your password	86
6.3	Perceptions of how attackers try to guess your password	87
6.4	The 25 hypotheses we investigated among password pairs	89
6.5	Password pairs for which perceptions of security differed from reality	90
6.6	How password characteristics correlate with security ratings	91
6.7	How password characteristics correlate with memorability ratings	92
6.8	Perceptions of the security and memorability of password-creation strategies	93
7.1	Regression results correlating advanced heuristics to guessability	102
7.2	List of conditions for online study Experiment 1	117
7.3	List of conditions for online study Experiment 2	117
7.4	Initial security results for Experiment 1	119
7.5	Security results for Experiment 1, 1class8	120
7.6	Security results for Experiment 1, 3class12	120
7.7	Initial security results for Experiment 2	121

7.8	Security results for Experiment 2, 1class8	121
7.9	Security results for Experiment 2, 3class12	121
7.10	Password characteristics by condition	124
7.11	Password creation and password recall by condition	125
7.12	Summary of how meter dimensions impacted key metrics	126
7.13	Participants' sentiment about password creation	128
7.14	Participants' agreement with statements about meter features	129
7.15	Participants' interactions with the colored bar	132
B.1	Full list of password pairs and results	167

Chapter 1

Introduction

Despite decades of research into alternatives, text passwords continue to be used by billions of people because of their comparative advantages [24]. These advantages—passwords’ familiarity, ease of implementation, and lack of anything to carry—make a world without text passwords unlikely in the near future. Two-factor authentication, single-sign-on systems, password managers, and biometrics promise to obviate remembering a distinct password for each account, but passwords will not disappear entirely.

Unfortunately, users frequently choose passwords that are easy for attackers to guess, either within a handful of guesses in the case of an online attack, or within millions of guesses in offline attacks. For example, celebrities’ private photos were recently obtained through a password-guessing attack on Apple’s iCloud service [36]. Similarly, attackers have successfully guessed many passwords from databases of hashed passwords stolen from companies like LinkedIn [146], eHarmony [178], Ashley Madison [82], Kickstarter [39], Gawker [21], and numerous others.

When asked to make a password, the average user has only a vague, and sometimes incorrect, idea of what characteristics make a password hard to guess [184]. Often, the user does not know how password-guessing attacks work [206], nor how to navigate the greater ecosystem around passwords [27, 70].

In this thesis, I work towards better supporting users’ password-security decisions by using data-driven methods to quantify the impact of password-strength meters on security and usability, model password-guessing attacks, unpack how users create and perceive passwords, and distill these combined insights into an improved password-strength meter that provides data-driven feedback specific to a user’s candidate password. The remainder of this thesis begins by summarizing closely related work on passwords, as well as data-driven user feedback more broadly in security and privacy topic areas, in Chapter 2.

A necessary first step in endeavoring to help users make better passwords is to determine whether users can be successfully encouraged to change their password-creation behavior. To this end, in Chapter 3, I describe an experiment quantifying the impact of password-strength meters on password security and usability. Many web sites have deployed password meters that provide visual feedback on password strength in order to help users create stronger text passwords. Although these meters are in wide use, their effects on password security and usability had not been well studied. In a 2,931-subject study of password creation in the presence of 14 password meters, we found that meters with a variety of visual appearances led users to create longer passwords. However, significant increases in resistance to a password-cracking algorithm were only achieved using meters that scored passwords stringently. These stringent meters also led participants to include more digits, symbols, and uppercase letters. Password meters also affected the act of password

creation. Participants who saw stringent meters spent longer creating their password and were more likely to change their password while entering it, yet they were also more likely to find the password meter annoying. However, the most stringent meter and those without visual bars caused participants to place less importance on satisfying the meter. Participants who saw more lenient meters tried to fill the meter and were averse to choosing passwords a meter deemed “bad” or “poor.”

Although this first study demonstrated that password-strength meters can lead users to create longer and more complex passwords, as well as passwords that are more resistant to guessing by a particular password-guessing algorithm, the study raised additional questions about the potential biases of measuring password strength by simulating only a single password-guessing algorithm. In Chapter 4, I address these questions about metrics by detailing our extensive experiments on *parameterized password guessability*, which is the notion of estimating password strength by measuring how many guesses a particular password-cracking algorithm with particular training data would take to guess that password. Unlike statistical metrics, guessability aims to model real-world attackers and to provide per-password strength estimates. We investigated how cracking approaches often used by researchers compare to real-world cracking by professionals, as well as how the choice of approach biases research conclusions. As a corollary to this work, we created the CMU Password Guessability Service (PGS) [37], enabling the research community to use our recommended configurations of password-cracking approaches for their own analyses of password security.

We found that semi-automated cracking by professionals outperforms popular fully automated approaches, but can be approximated by combining multiple such approaches. These approaches are only effective, however, with careful configuration and tuning; in commonly used default configurations, they underestimate the real-world guessability of passwords. Furthermore, we found that analyses of large password sets are often robust to the algorithm used for guessing as long as it is configured effectively. However, cracking algorithms differ systematically in their effectiveness guessing passwords with certain common features (e.g., character substitutions). This result has important implications for analyzing the security of specific password characteristics or of individual passwords (e.g., in a password meter or security audit). Our results highlight the danger of relying only on a single cracking algorithm as a measure of password strength and constitute the first scientific evidence that automated guessing can often approximate guessing by professionals.

Having found that more nuanced and better configured models of password-guessing attacks could approximate guessing by professionals, the next step towards helping users make better passwords is unpacking how and why users sometimes create predictable passwords. We did so in two parts. First, in Chapter 5, I present a series of analyses of password structures, semantics, and password-creation strategies that collectively provide new insight into password creation and identify ways to help users create better passwords. Combining crowdsourcing with programmatic techniques, we reverse engineered over 45,000 passwords. We quantified character substitutions, which were both infrequent and predictable. We also used natural-language corpora to unpack the semantics of words and phrases contained in passwords. Finally, we delved into password-creation strategies. We found that forcing users to conform to a password-composition policy can result in passwords that are easier to guess, whereas guiding users to make judicious edits may be prudent. We also analyzed the degree to which users heeded suggestions during password creation.

While these results suggested directions for improving the passwords themselves that users create, the second part of unpacking how and why users sometimes create predictable passwords focused on *why* users do so. Building on a previous 49-participant laboratory study of participants’ strategies and misconceptions during password creation [184], I describe in Chapter 6 our study of how users’ *perceptions* of password security compare to reality. Although many users create predictable passwords, the extent to which users

realize these passwords are predictable had not been well understood prior to this work. We investigated the relationship between users' perceptions of the strength of specific passwords and their actual strength, as measured by the aforementioned CMU Password Guessability Service [37]. In a 165-participant online study, we asked participants to rate the comparative security of carefully juxtaposed pairs of passwords, as well as the security and memorability of both existing passwords and common password-creation strategies. Participants had serious misconceptions about the impact of basing passwords on common phrases and including digits and keyboard patterns in passwords. However, in most other cases, participants' perceptions of what characteristics make a password secure were consistent with the performance of current password-cracking tools. We found large variance in participants' understanding of how passwords may be attacked, potentially explaining why users nonetheless make predictable passwords.

Collectively, these previous studies provided extensive insight into precisely how and why users create passwords that are predictable, how to use password-strength meters to encourage users to create stronger passwords, and how to provide more accurate measurements of passwords' resistance to a range of password-guessing attacks. The final step of this thesis involved integrating these insights into a new password-strength meter that gives users data-driven feedback about their password. In Chapter 7, I detail both the design and validation of this meter. Whereas most previous password-strength meters gave highly inaccurate estimates of password strength because they relied on very simple heuristics (e.g., the length of the password and inclusion of different character classes) [51], a major design goal for our meter was to provide more accurate estimates of password strength, inspired by the password-security metrics work described in Chapter 4. We do so by integrating our recent work on using artificial neural networks to model human-chosen passwords [130], as well as by using more advanced heuristics to identify the types of patterns we commonly observed in Chapter 5. Whereas the neural networks provide a principled, statistical model of how humans choose passwords, neural networks do not output human-intelligible explanations for why they have made a particular classification of password strength. In this way, the advanced heuristics are complementary, providing human-intelligible feedback. In our iterative design process, we crafted the meter's visual design to reflect the most successful techniques we studied in Chapter 3 and designed the meter's text feedback to correct users' misconceptions, as identified in Chapter 6.

To evaluate the impact of our improved password-strength meter, we performed a formative ten-participant laboratory study and a summative 4,509-participant online study. From the laboratory study, we found initial evidence that our data-driven meter taught participants new strategies for improving their passwords. We also identified numerous usability improvements and tweaks to correct instances where participants' understanding of meter aspects differed from our intention. In the subsequent online study, we evaluated the different scoring and interaction aspects of our meter. Under the more common password-composition policy we tested, we found that the data-driven meter with detailed feedback led users to create significantly more secure, and no less memorable, passwords than a meter with only a bar as a strength indicator. I also describe the low-level implementation of our meter, whose code I am releasing open-source. I close this thesis with concluding thoughts and an outline of future work in Chapter 8.

Chapter 2

Related Work

I outline related work in seven sections. The first five sections focus specifically on passwords. To explain the greater context of password security, I first outline threats to password security (Section 2.1) with a focus on threats where the strength of a user’s password matters. Then, I detail metrics for password security (Section 2.2), outlining both statistical metrics and metrics focused on modeling adversarial password guessing. Because I take the latter approach in this thesis for reasons I detail in that section, I subsequently detail password-guessing approaches (Section 2.3), laying the groundwork for our analyses of the accuracies and biases of modeling such approaches in Chapter 4.

The subsequent two sections discuss prior work related to how humans choose passwords. In Section 2.4, I describe work examining characteristics of user-chosen passwords, including the way passwords are structured and the extent to which they include semantically meaningful content. I also discuss linguistic analyses of passwords, as well as external decisions related to password choice, such as password reuse and attempts to make passwords memorable. In Section 2.5, I focus on efforts to help users choose better passwords. In particular, I discuss efforts to use password-composition policies, proactive password checking, and password-strength meters to direct users towards stronger passwords.

In the final two sections, I discuss prior work from the broader security and privacy literature on concepts I adopt in this thesis. First, I discuss the prior literature evaluating users’ perceptions of security (Section 2.6), laying the foundation for Chapter 6 of this thesis. Finally, I discuss efforts to provide users with data-driven feedback about their security and privacy outside of the context of passwords (Section 2.7).

2.1 Threats to Password Security

The authentication ecosystem is vulnerable to a number of attacks. For each type, the security of a password has a different impact. In this section, I describe the most important threats.

Sometimes, the security of a password does not matter [25]. If a user is phished, the attacker gets the password in plaintext. If a keylogger has been installed on a user’s machine or the attacker is able to observe the user typing in his or her password (an attack known as “shoulder surfing”), password strength also does not matter.

In some other cases, it is most important that a password not be trivially predictable. In what is known as an online attack, the attacker attempts to authenticate to a running system using guesses of what the user’s password might be. After a few incorrect guesses, often 3–10, best practices dictate the system rate-limit

subsequent attempts or lock the account and require alternate authentication [69]. To be protected against an online attack, a password should not be among the million most common passwords [69], nor should it include the user's personal information (e.g., family member's name, birthdate) in case the attacker targets the attack to the potential victim. This chain of reasoning assumes that rate-limiting is properly implemented. As the recent Apple iCloud hack regrettably demonstrated [36, 122], this is not always the case. In other words, even for online attacks, large-scale guessing may be possible if the system's implementation is flawed.

Another threat, an offline attack, usually involves large-scale guessing. Best practices dictate that systems store passwords hashed using a cryptographically secure one-way function. When a user attempts to authenticate, the system hashes the submitted password and verifies that its hash matches the value in its database. Sadly, numerous hashed password databases have been compromised in recent years [21, 31, 79, 178], enabling offline attacks.

While it is efficient to compute $H(\textit{password})$ given a password, it is very inefficient to find the password given $H(\textit{password})$. Attackers thus proceed in an offline attack by guessing likely candidate passwords, hashing those candidates, and seeing if the hash matches any user's hashed password contained in the database. A password's resistance to an offline attack is highly situational. It depends on the way passwords are stored, including the type of hash function used to store the password, as well as the attacker's resources and motivation. Precomputed mappings of hashes to the passwords that produce them ("rainbow tables") enable many passwords to be discovered trivially. However, combining each user's password with a unique, random string ("salting") both prevents the use of rainbow tables and forces the attacker to try each guess on a per-user basis, rather than for all users at once.

The hash function itself is also a major consideration. General-purpose hash functions like MD5 were designed for efficiency, which makes them poor for storing passwords due to their speed. Modern hardware can try billions of MD5 guesses per second [166, 171]. Unfortunately, numerous services [21, 31, 178] have hashed passwords with MD5. Best practices dictate the use of intentionally slow hash functions like bcrypt [143], scrypt [138], or Argon2 [19]. To be more expensive to compute, they require many sequential rounds of computation or large amounts of memory during computations. Attackers can therefore only try hundreds of guesses per second [82, 166]. The 2015 Ashley Madison breach [12] was the first major compromise involving bcrypt. Unfortunately, they retained a legacy database that stored passwords using MD5 [82], reiterating that best practices are not always followed.

The main security threat of an offline attack derives from password reuse [25]. Once attackers have learned a particular user's password in an offline attack, they will try the same username and same password, or close variants [205], on other sites. Users often reuse passwords [49, 101, 184, 191], which can cause serious harm. For instance, attackers recently infiltrated Mozilla's Bugzilla database and learned of crucial zero-day vulnerabilities because one Mozilla administrator had reused his password on another, compromised site [75]. Similarly, recent estimates attribute the compromise of 20 million accounts on Taobao, a Chinese online shopping website similar to eBay, to password reuse [58].

Although offline attacks against online accounts, as described above, are among the most widely discussed types of large-scale guessing, there are a number of other common scenarios where large-scale guessing is plausible. These cases occur when cryptographic key material is derived from, or protected by, a password. For instance, for password managers that sync across devices [89] or privacy-preserving cloud backup tools (e.g., SpiderOak [164]), the security of files stored in the cloud depends directly on password strength. Furthermore, cryptographic keys used for asymmetric secure messaging (e.g., GPG private keys), encrypted hard drives, and Windows Domain Kerberos Tickets [44] are protected by passwords. If an adversary obtains

the relevant file containing key material, the strength of the password is critical for security because the adversary can perform large-scale guessing, limited only by his or her resources. The importance of this final type of attack is likely to grow with the wider adoption of password managers and encryption tools.

2.2 Password-Security Metrics

In light of the many potential attacks against passwords, being able to measure the strength of a particular password becomes crucial. Gauging the strength of an individual password, however, is a complex and nuanced problem [23, 76, 185]. Historically, Shannon entropy was used to rate the strength of a password [22]. At a high level, Shannon entropy measures the unpredictability of elements in a set. Unfortunately, as outlined by recent work [22], Shannon entropy is inappropriate for measuring password strength for a handful of reasons. First, it measures the entirety of a set. If a set of passwords contains some extremely predictable passwords, this predictability will not necessarily be captured by the entropy estimate [22, 104, 196]. In other words, entropy will not indicate what proportion of a set an attacker will guess, nor will it directly indicate how strong a particular password in a set is. Furthermore, accurately calculating entropy requires an extremely large set of passwords. If multiple passwords in a set are very closely related in structure, yet not perfectly identical, entropy will not capture that an intelligent adversary might be able to leverage these similarities in guessing passwords. As a result, for password sets of the size of even a large company's userbase, Shannon entropy is inappropriate as a measure of password strength [22]. Methods to estimate the entropy of a password without requiring a huge corpus [35] often do not reflect a password's actual resistance to guessing [104, 199].

Two main classes of metrics have emerged in place of entropy: statistical metrics and parameterized metrics. Both classes focus on *guessability*, the number of guesses needed by an adversary to guess a given password or a fraction of a set.

Statistical metrics are particularly valuable for examining password sets as a whole. For example, Bonneau introduced partial guessing metrics [22] for estimating the number of guesses required for an idealized attacker, who can perfectly order guesses, to guess a given fraction of a set. Since password distributions are heavy-tailed, very large samples are required to determine a set's guessability accurately. Furthermore, many statistical metrics do not calculate the strength of an individual password, though a handful do [23]. Regardless, all of these statistical metrics require a very large corpus of training data (e.g., previously released passwords) that is closely matched [104] to the passwords that are being modeled. Because very few sets of passwords that one would encounter are sufficiently large to provide accurate training for statistical models, even these more advanced statistical methods are often not appropriate for researchers or for use in client-side password checking [185].

Parameterized metrics instead investigate guessability under a cracking algorithm and training data [23, 104, 196]. These metrics thus model an adversary using existing tools, rather than an idealized attack, though the metric is only as good as the chosen algorithm and training data. Parameterized metrics can also be used to compare password sets without fully running the algorithm [123]. Furthermore, because estimated guess numbers (rather than precise guess numbers) are often sufficient for evaluating the security of a password in the context of proactive password checking, Dell'Amico and Filippone recently proposed using Monte Carlo methods to estimate guess numbers, enabling probabilistic password-guessing attacks to be modeled very quickly to a large number of guesses [53].

In contrast to statistical metrics, parameterized metrics have two important properties. First, they estimate

the guessability of each password individually. Estimating guessability per-password is important for security audits and to provide feedback to a user about a password he or she has created. This latter promises to become more widespread as proactive feedback tools move from length-and-character-class heuristics [51] to data-driven feedback [38, 107]. Second, parameterized metrics aim to estimate security against real-world, rather than idealized, attacks. Researchers previously assumed automated techniques approximate real-world attackers [104, 196]. Chapter 4 of this thesis evaluates the accuracies and biases of this assumption, and it is also the first to compare this assumption to attacks by human experts in password forensics.

Parameterized metrics have been used to measure password strength in a number of previous studies [38, 49, 54, 67, 71, 104, 108, 123, 126, 145, 157, 182, 188, 196, 205]. While there are many different methods for cracking passwords, time and resource constraints lead many researchers to run only a single algorithm per study. However, prior to the research described in Chapter 4 of this thesis [185], it was an open question whether this strategy accurately models real-world attackers, or whether choosing a different algorithm would change a study's results.

Measuring guessability entails computing a *guess number* for each password indicating how many guesses a particular password-cracking approach configured and trained in a particular way would take to guess that password. Because guessability estimators can only run for finite time, there is necessarily a guess cutoff at which remaining passwords are labeled “unguessed.”

2.3 Types of Guessing Attacks

The selection of a guessing attack to model is crucial to analyzing password guessability. Researchers and security practitioners have long investigated how to guess passwords. A handful of studies [43, 54, 145] have compared the aggregate results of running different cracking approaches. Other studies have compared results of running different cracking approaches based on guess numbers [42, 60, 123]. In this section, I highlight four major types of attacks.

2.3.1 Brute-Force and Mask Attacks

Brute-force attacks are conceptually the simplest. They are also inefficient and therefore used in practice only when targeting very short or randomly generated, system-assigned passwords.

Mask attacks are directed brute-force attacks in which password character-class structures, such as “seven lowercase letters followed by one digit,” are exhausted in an attacker-defined order [167]. While this strategy may make many guesses without success, mask attacks can be effective for short passwords as many users craft passwords matching popular structures [111, 177]. Real-world attackers also turn to mask attacks after more efficient methods exhaust their guesses.

2.3.2 Probabilistic Context-Free Grammar

In 2009, Weir et al. proposed using a probabilistic context-free grammar (PCFG) with a large training set of passwords from major password breaches [187] to model passwords and generate guesses [197]. The intuition behind PCFGs is that passwords are built with template structures (e.g., 6 letters followed by 2 digits) and terminals that fit into those structures. A password's probability is the probability of its structure multiplied by those of its terminals. In more detail, Weir et al. use training data to create a context-free grammar in

which non-terminals represent contiguous strings of a single character class. From the passwords observed in its training data, PCFG assigns probabilities to both the structure of a password (e.g., *monkey99* has the structure $\{\textit{six letters}\}\{\textit{two digits}\}$) and the component strings (e.g., “99” will be added to the list of two-digit strings it has seen) [197]. A number of research studies [42, 54, 61, 104, 123, 126, 157, 182, 196, 205] have used PCFG or a close variant to compute guessability.

Kelley et al. proposed improvements to Weir et al.’s PCFG algorithm, like treating uppercase and lowercase letters separately and training with structures and component strings from separate sources [104]. Because they found these modifications improved guessing effectiveness, we incorporate their improvements in our tests. In addition, multiple groups of researchers have proposed using grammatical structures and semantic tokens as PCFG non-terminals [145, 188]. Researchers have found that using separate training sources for structures and terminals improves guessing [104]. It is also beneficial to assign probabilities to unseen terminals by smoothing, as well as to augment guesses generated by the grammar with passwords taken verbatim from the training data without abstracting them into the grammar [106].

2.3.3 Markov Models

Narayanan and Shmatikov first proposed using a Markov model of letters in natural language with finite automata representing password structures [135]. Castelluccia et al. used a similar algorithm for password meters [38]. John the Ripper and Hashcat offer simple Markov modes in their cracking toolkits as well.

Conceptually, Markov models predict the probability of the next character in a password based on the previous characters, or context characters. Using more context characters can allow for better guesses, yet risks overfitting. Smoothing and backoff methods compensate for overfitting.

Recently, Dürmuth et al. [60] and Ma et al. [123] independently evaluated many variations of Markov models and types of smoothing in cracking passwords, using large sets of leaked passwords for training. Both groups compared their model with other probabilistic attacks, including Weir et al.’s original PCFG code, finding particular configurations (e.g., using 6-grams) of a Markov model to be more efficient at guessing passwords for some datasets [60, 123].

2.3.4 Neural Networks

Recently, our group proposed and evaluated the use of artificial neural networks (henceforth termed “neural networks”) for guessing passwords [130]. Building on neural networks’ documented success at classification problems and generating novel text, we show how password guessing can be reenvisioned as a process that starts with modeling subsequent characters of a password based on the previous (context) characters [130]. We tested a number of aspects of the potential design space for neural networks, including the application of transference learning, augmenting sets of passwords with natural-language corpora as training data, and varying the model size.

In our comparative tests, we found that neural networks can often guess passwords more effectively than other password-guessing approaches, including as probabilistic context-free grammars and Markov models. We also found that our neural networks can be compressed to as little as hundreds of kilobytes without substantially worsening guessing effectiveness.

2.3.5 Mangled Wordlist Attacks

Perhaps the most popular strategy in real-world password cracking is the dictionary attack and its variants [79]. First proposed by Morris and Thompson in 1979 [132], modern-day dictionary attacks often combine *wordlists* with *mangling rules*, string transformations that modify wordlist entries to create additional guesses. Wordlists usually contain both natural-language dictionaries and stolen password sets. Typical mangling rules perform transformations like appending digits and substituting characters [139, 168]. Unlike probabilistic password-guessing approaches (e.g., PCFG, Markov models, and neural networks), mangled wordlist attacks often do not directly rely on statistical analysis of password sets, yet often guess both relatively accurately and very efficiently in terms of wall-clock time and disk space [43, 185].

Many modern cracking tools, including Hashcat [166], John the Ripper [139], and PasswordsPro [96], support these attacks, which we term *mangled wordlist attacks*. The popularity of this category of attack is evident from these tools' wide use and success in password-cracking competitions [109, 140]. Furthermore, a number of research papers have used John the Ripper, often with the default mangling rules [42, 49, 51, 67, 71, 86, 108, 204] or additional mangling rules [54, 61, 205].

Expert password crackers, such as those offering forensic password-recovery services, frequently perform a variant of the mangled wordlist attack in which humans manually write, prioritize, and dynamically update rules [79]. We term these manual updates to mangling rules *freestyle rules*. We evaluated guessability using off-the-shelf tools relying on publicly available wordlists and mangling rules. We also contracted a password-recovery industry leader to do the same using their proprietary wordlists and freestyle rules.

2.4 How Users Choose Passwords

While the discussion of related work to this point has focused on attacks against passwords, understanding tendencies in how humans choose passwords is essential to helping users avoid common patterns along the way to creating better passwords. In this section, we discuss how humans structure passwords, how they integrate natural-language content into passwords, and how they make decisions about reusing, storing, or writing down passwords.

2.4.1 Structural Characteristics of Human-Chosen Passwords

Many users make passwords that are quite predictable [124, 191] even for relatively important accounts [67, 126]. Passwords are generally too short to provide much resistance to attacks [22, 118, 126, 187], and users tend to put digits and symbols at the end of the password [22, 183, 184] and capital letters at the beginning [183, 184].

Users tend to base passwords around predictable words and phrases, including names [94], dates [189], song lyrics [115, 183], and other concepts or objects they like [184]. Furthermore, when a password contains multiple words, those words tend to be semantically related [29, 188]. Keyboard patterns (e.g., “1qaz2wsx”) are common [184, 188], and passwords sometimes contain character substitutions (e.g., replacing “a” with “@”) [102]. These characteristics vary somewhat for passwords created on touchscreen devices [129].

A number of researchers have surveyed users about password creation. Unsurprisingly, most passwords contain meaningful elements [207], such as the name of the user or a relative, geographic locations, and names of sports teams [114, 127].

Users' passwords appear to be becoming longer and more complex over time. In 1997, Zviran and Haga found an average password length of six characters, with 14% including digits and less than 1% including symbols [207]. By the mid-2000s, the average length had increased to eight characters, at least 40% used digits, and 3-16% used symbols [32]. A more recent survey asking users to compare passwords they used at various times similarly found that users now choose stronger passwords than in the past [191].

A number of researchers have analyzed passwords obtained from public leaks of password data, including sets from RockYou (2009), Sony (2011), Gawker (2011), LinkedIn (2012), and Adobe (2013). Common passwords in these sets include *password*, *password1*, and *123456* [56, 124, 146]. For Sony and Gawker, 14% of passwords included people's names, 25% included dictionary words, and 8% included place names, with the most common modification being the addition of digits [93]. Names, dictionary words, keyboard patterns, and numbers were similarly prevalent in RockYou [56, 189].

Several studies have considered factors affecting password selection. Users choose passwords more carefully (and reuse them less frequently) for accounts they perceive to have higher value [68, 136]. Others find correlations with demographics [22, 126] and with users' annoyance with the password creation process [126]. Zhang et al. found that up to 17% of passwords created to replace an expiring prior password can be broken in under five guesses if the old password is known. Common transformations included incrementing a number and replacing one symbol with another [205].

2.4.2 Linguistic and Semantic Properties of Passwords

Some recent work has examined passwords explicitly from a linguistic perspective. Bonneau and Shutova searched for predictable Amazon payphrases, which must contain two or more words, but no digits or symbols [29]. They found that many expected phrases from the arts, sports, and geography had been chosen, and that noun phrases were common. Rao et al. investigated grammatical structures within long passwords as an aid to cracking [145]. However, their mostly manual analysis examined a sample of only 144 passwords and did not consider substitutions or interstitial digits and symbols.

Jakobsson and Dhiman built an automated parser to study how dictionary words are modified as part of passwords, with particular focus on concatenated words, "leet" substitutions, and misspellings [102]. Veras et al. automatically separated passwords into linguistic chunks (words) based on large corpora of text, tagged each word with its part of speech, and further classified nouns and verbs semantically using a lexical database of English-language concepts [188]. They found nouns to be overwhelmingly popular among parts of speech, while top semantic categories included names, cities, and words related to love.

In contrast to this past work, in Chapter 5, we instead use a combination of crowdsourcing and automated techniques to reverse engineer passwords, providing greater accuracy in creating the abstract semantic representation of a password. We then perform analyses on a number of different levels to provide a holistic understanding of user behavior in constructing passwords.

2.4.3 Password Management and Reuse

The overall password ecosystem has also been a major area of investigation. Many users view passwords as a burden [59] and exhibit potentially insecure behaviors when managing passwords [74, 86, 136, 169]. However, many of these behaviors are likely rational coping strategies for users who are asked to make far more distinct, complex passwords than they could possibly remember [3, 70, 169].

A number of studies have investigated password-management practices [3, 74, 84, 95, 160] and how users respond to password-creation requirements [142, 193]. Researchers have studied how users recall multiple passwords, including text passwords and graphical passwords [40]. Researchers have also explored automatically increasing password strength by adding random characters, which study participants could shuffle until arriving at a configuration they liked. The authors found that inserting two random characters increased security, yet adding more characters hurt usability [71].

More recently, Stobert and Biddle interviewed 27 participants about their strategies for password management and usage. Participants had an average of 27 accounts and five passwords. They often made tradeoffs between following password advice and expending too much effort [169].

A key coping mechanism is password reuse. Users often reuse passwords across accounts [49, 68, 101, 191]. Even when they do not reuse a password verbatim, they frequently make only small, predictable modifications [49, 184, 205]. Other studies have focused on economic analyses of the password ecosystem [27], internationalization issues in password systems [30], and designing systems that make offline attacks more easily detectable [103]. Finally, a handful of studies have examined the memorability of system-assigned passwords, finding that humans can learn long, random secrets through spaced repetition [28] and that system-assigned passphrases are not significantly more memorable than system-assigned passwords [154].

2.5 Helping Users Create Better Passwords

Three major efforts attempt to help users avoid the predictable password-creation tendencies described in the previous section and therefore create stronger passwords. First, password-composition policies dictate required characteristics a password must include, such as a minimum length and the mandatory inclusion of certain character classes (e.g., digits or symbols). Second, proactive password checking aims to model a password's security and only permit users to select a password the model deems sufficiently strong. Proactive password checking is a general approach that can either mandate that passwords pass particular strength checks or simply encourage users towards creating stronger passwords by showing them the results of the proactive strength checks. The latter encompasses the third approach, using password-strength meters to provide visual displays of password strength. Below, we detail each of these three approaches.

2.5.1 Password-Composition Policies

Without intervention, users tend to create simple passwords [68, 116, 172, 202]. Many organizations use password-composition policies that force users to select more complex passwords to increase password strength. However, users may conform to these policies in predictable ways, reducing password strength [35].

Although prior work has shown that password-composition policies requiring more characters or more character classes can improve resistance to automated guessing attacks, many passwords that meet common policies remain vulnerable [104, 142, 193, 196]. Furthermore, strict policies can frustrate users, inhibit their productivity, and lead users to write their passwords down [4, 90, 95, 108, 165]. Some password-composition policies, particularly those that emphasize a balance of length and character-class complexity [157], lead users to create stronger passwords than other password-composition policies.

2.5.2 Proactive Password Checking

Rather than using a password-composition policy to mandate particular password characteristics in the (often false) hope that users will fulfill those requirements in unpredictable ways, a second approach is to proactively check passwords based on a given model of how humans choose passwords. Current real-time password checkers can be categorized based on whether they run entirely client-side. Historically, checkers with a server-side component were more accurate because they could directly leverage large amounts of data. For instance, researchers proposed using server-side Markov models to gauge password strength [38]. Others have studied using large-scale training data from leaked passwords and natural-language corpora to show users predictions about what they will type next in their password [107], requiring that users make multiple hard-to-predict choices within their password for it to be permitted.

Unfortunately, a server-side component introduces substantial disadvantages for security. In some cases, sending a password to a server for password checking destroys all security guarantees. For instance, passwords that protect an encrypted volume (e.g., TrueCrypt) or cryptographic keys (e.g., GPG), as well as the master password for a password manager, should never leave the user's device, even for proactive password checking. As a result, accurate password checking is often missing from these security-critical applications. In cases when a password is eventually sent to the server (e.g., for an online account), a real-time, server-side component both adds latency and opens password meters to powerful side-channel attacks based on keyboard timing, message size, and caching [162].

Prior client-side password checkers, such as those running entirely in a web browser, rely on heuristics that can be easily encoded. Many common meters rate passwords based on their length or inclusion of different character classes [51, 182]. Unfortunately, in comprehensive tests of both client- and server-side password meters, the password scoring of all but one meter was highly inaccurate [51].

2.5.3 Password Meters

While the general category of proactive password checking simply means that a candidate password is evaluated against some model of how passwords are chosen, users most frequently encounter these proactive estimates of password strength as visualized by password meters [65, 182]. Generally, password meters display estimated password strength through some visual metaphor, such as a bar that fills up as a password increases in strength. Many password meters guide users toward, but do not strictly require, complex passwords. This approach reflects the behavioral economics concept of nudging, or soft paternalism [121, 175].

Most widely deployed meters are based on basic heuristics, such as estimating a password's strength based on its length and the number of character classes used [182]. Unfortunately, these basic heuristics frequently do not reflect the actual strength of a password [2, 51]. Among prior password meters based on heuristics, only zxcvbn [198, 199] uses more advanced heuristics, which is why it is the only accurate password-strength meter that has been widely deployed [51, 130].

As we show in Chapter 3, password-strength meters can successfully encourage users to create stronger passwords [182], though perhaps only for higher-value accounts [65]. Researchers have previously proposed a number of password-strength visualizations other than the typical bar metaphor [51, 182]. For example, Sotirakopoulos et al. investigated a password meter that compares the strength of a user's password with those of other users [163]. While visualizations can help users understand password-guessing attacks [206] and some password-strength meters can give detailed feedback about predictability [107], alternative visualizations have yet to be widely adopted.

2.6 Users' Perceptions of Security

In Chapter 6, I describe users' perceptions of password security and how these perceptions compare to reality. Hundreds of research studies have been conducted at the general intersection of usability and security [73], but surprisingly few have specifically investigated users' *perceptions* of security. One stream of qualitative work has examined users' mental models [8, 190] and "folk models" [144, 194] of security, finding that non-expert users' mental models often differ from those of experts. For instance, non-experts perceive losing a password as similar to losing a key, whereas experts perceive the same event as more akin to losing a credit-card number [8]. Likewise, a study that examined users' perceptions of the most important computer-security practices [98] again showed a disconnect between non-technical users and experts. Notably, users' perceptions of the efficacy of different security practices impact the adoption of security technologies [18, 52, 97, 173].

In the password domain, a recent interview study of password creation using a think-aloud protocol implicitly rested on users' perceptions of password security [184]. Security perceptions, however, were not the focus of that qualitative study.

Most closely related to our work in Chapter 6, a 384-participant study examined users' perceptions of the security and usability of Android graphical unlock patterns [9]. Participants compared two unlock patterns and assessed their relative security and memorability. Unlike our study, that study did not compare the actual security of these patterns.

2.7 Data-Driven Feedback

The final part of this thesis, Chapter 7, builds the insights from the other chapters of this thesis into a data-driven password-strength meter that uses personalized examples from the user's candidate password to teach the user about creating a secure password. More broadly within the security and privacy research literature, a handful of other researchers have also evaluated the effectiveness of using personalized examples derived from users' own data in helping users make privacy and security decisions. These studies rely on different techniques and focus on different application areas than our work. Furthermore, a number of these techniques do not present this information to users "just in time" as they are about to make a security or privacy decision, in contrast to the focus of my thesis.

Overall, providing users information about privacy or security can impact their decisions. For example, Tsai et al. found that consumers will pay a premium price to make purchases from more privacy-protective businesses when information about privacy is made accessible to consumers [180].

Much of the research on data-driven support for users in making security and privacy decisions has centered on smartphones. For example, Almuhiemedi et al. found that showing users how frequently different smartphone apps access sensitive data can nudge users to restrict apps' access to this information [6]. That project builds on work by Harbach et al., who demonstrated that personal examples of the data accessible to smartphone apps help users understand otherwise abstract smartphone permission requests [87]. Similarly, Balebako et al. demonstrated that summary visualizations and just-in-time notices of smartphone privacy leakages help correct users' misconceptions about data sharing [11]. Their tool collects data about these leaks using the Taintdroid platform [66]. In a slightly different focus area, Consolvo et al. proposed bringing transparency to the information that is leaked unencrypted over wi-fi networks by showing this unencrypted data to users [46].

Some researchers have investigated the use of data-driven methods and feedback during the password-creation process, but using different types of data than I do, and often using this data in a less directed manner. For decades, researchers have suggested providing users proactive feedback based on the password they are typing [15, 20]. Two studies have investigated the underlying premise of whether meters impact user behavior in creating passwords. Complementary to my own online study of password-strength meters [182] (Chapter 3), Egelman et al. conducted a laboratory study of password-strength meters, also finding that the presence of a meter resulted in stronger passwords, yet only for high-value accounts [65].

The amount of data that drives this feedback on passwords has varied, however. Most websites currently rate passwords using length and character-class heuristics [51]. In the academic community, researchers have suggested using peer pressure by ranking passwords comparatively to other users [163] or estimating password strength using Markov models [38]. In this same vein, the Telepathwords project [107] uses large corpora of leaked passwords and dictionaries to guess what the user might type next. In contrast to my work, neither the scheme based on Markov models nor the Telepathwords scheme provides directed feedback based on the semantic characteristics or explicit guessability of the passwords. Furthermore, neither scheme directly takes into account users' misperceptions of password security.

A few projects have also applied data-driven methods to privacy decision-making. Wills and Zeljkovic examined data-driven decision support in privacy decisions related to online behavioral advertising. They prototyped a tool that examines browser history to help a user understand when their data has been tracked [201]. Their prototype presents a table of websites a user has visited, along with a list of the third parties tracking users on each site. Angulo et al. proposed using a plugin to visualize similar privacy leaks, such as the user's name and email address, in the browser [7]. Cranor et al. automatically parsed 6,191 standard-format privacy notices from the U.S. financial industry, building a comparative Bank Privacy Website [47].

Other projects have used semi-automated methods or crowdsourcing to gather data about privacy and then present this data to users. For instance, Liu et al.'s Privacy Grade project crowdsources analysis of the permissions requested by different smartphone apps [119, 120]. They assign each app a "privacy grade" based on the appropriateness of the permissions the app requests, as judged by crowdworkers. Researchers aim to provide users with similarly succinct summaries of privacy information, but for full-length privacy policies rather than smartphone app permissions [148]. They aim to do so by relying on both natural-language processing and crowdsourcing. Projects like "Terms of Service; Didn't Read" (TOS;DR) have already used crowdsourcing on a small scale to put information about companies' privacy policies and terms of service into a standardized, usable format [174]. In the rare cases when machine-readable privacy information is available, as was the case for some websites using the P3P standard, crowdsourcing is not necessary [63].

In a number of other application domains within privacy and security, collected data drives decision making by software programs, rather than informing a user's decision. For instance, data-driven techniques are popular in the detection of phishing messages [1, 16, 128], intrusion detection, and malware identification.

Chapter 3

The Impact of Password-Strength Meters

3.1 Introduction

While the premature obituary of passwords has been written time and again [113, 131], text passwords remain ubiquitous [24]. Unfortunately, users often create passwords that are memorable but easy to guess [20, 131, 142]. To combat this behavior, system administrators employ a number of measures, including system-assigned passwords and stringent password-composition policies. System-assigned passwords can easily be made difficult to guess, but users often struggle to remember them [71] or write them down [154]. Password-composition policies, sets of requirements that every password on a system must meet, can also make passwords more difficult to guess [35, 196]. However, strict policies can lead to user frustration [158], and users may fulfill requirements in ways that are simple and predictable [35].

Another measure for encouraging users to create stronger passwords is the use of password meters. A password meter is a visual representation of password strength, often presented as a colored bar on screen. Password meters employ suggestions to assist users in creating stronger passwords. Many popular websites, from Google to Twitter, employ password meters.

Despite their widespread use, password meters have not been well studied. This experiment contributes what we believe to be the first large-scale study of what effect, if any, password meters with different scoring algorithms and visual components, such as color and size, have on the security and usability of passwords users create.

We begin by surveying password meters in use on popular websites. Drawing from our observations, we create a control condition without a meter and 14 conditions with meters varying in visual features or scoring algorithm. The only policy enforced is that passwords contain at least eight characters. However, the meter nudges the user toward more complex or longer passwords.

We found that using any of the tested password meters led users to create passwords that were statistically significantly longer than those created without a meter. Meters that scored passwords more stringently led to even longer passwords than a baseline password meter. These stringent meters also led participants to include a greater number of digits, symbols, and uppercase letters.

We also simulated a probabilistic context-free grammar password-cracking algorithm [196] and compared

Previously published as Blase Ur, Patrick Gage Kelley, Saranga Komanduri, Joel Lee, Michael Maass, Michelle L. Mazurek, Timothy Passaro, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor. How Does Your Password Measure Up? The Effect of Strength Meters on Password Creation. In Proc. USENIX Security Symposium, 2012.

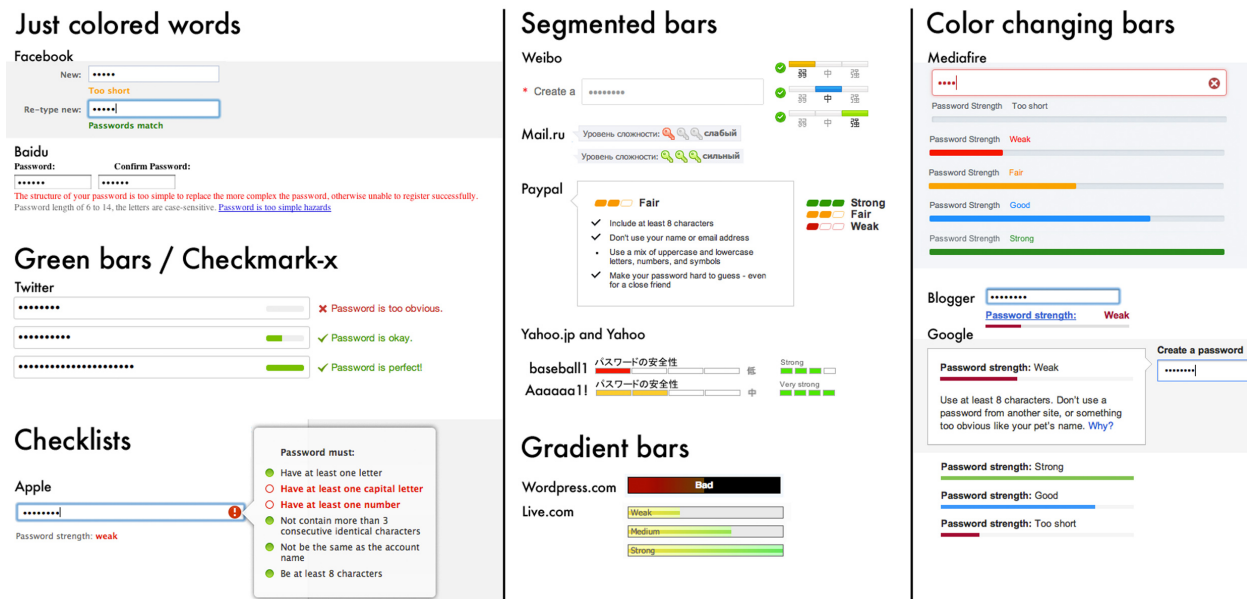


Figure 3.1: A categorized assortment of the 46 unique indicators we found across Alexa’s 100 most visited global sites.

the percentage of passwords cracked in each condition by adversaries making 500 million, 50 billion, and 5 trillion guesses. Passwords created without a meter were cracked at a higher rate than passwords in any of the 14 conditions with meters, although most differences were not statistically significant. Only passwords created in the presence of the two stringent meters with visual bars were cracked at a significantly lower rate than those created without a meter. None of the conditions approximating meters we observed in the wild significantly increased cracking resistance, suggesting that currently deployed meters are not sufficiently aggressive. However, we also found that users have expectations about good passwords and can only be pushed so far before aggressive meters seem to annoy users rather than improve security.

We proceed by first surveying popular websites’ password meters in Section 3.2, enabling us to base our meter experiments on widely used practices. We then present our methodology in Section 3.3. Section 3.4 contains results related to password composition, cracking, and creation, while Section 3.5 summarizes participants’ attitudes. We discuss these findings in Section 3.6 and conclude in Section 3.7.

3.2 Password Meters “In the Wild”

To understand how password meters are currently used, we examined Alexa’s 100 most visited global sites in January 2012. Among these 100 sites, 96 allowed users to register and create a password. Of these 96, 70 sites (73%) gave feedback on a user’s password based either on its length or using a set of heuristics. The remaining 26 sites (27%) provided no feedback. In some cases, all sites owned by the same company used the same meter; for example, Google used the same meter on all 27 of its affiliates that we examined. In other cases, the meters varied; for example, ebay.de used a different mechanism than ebay.com. We observed 46 unique indicators, examples of which are shown in Figure 3.1.

Indicators included bar-like meters that displayed strength (23, 50%); checkmark-or-x systems (19, 41.3%); and text, often in red, indicating invalid characters and too-short passwords (10, 21.2%). Sites with bar-like meters used either a progress-bar metaphor (13, 56.5%) or a segmented-box metaphor (8, 34.8%). Two sites presented a bar that was always completely filled but changed color (from red to green or blue) as password complexity increased. Three other sites used meters colored with a continuous gradient that was revealed as users typed. Sites commonly warned about insecure passwords using the words “weak” and “bad.”

We examined scoring mechanisms both by reading the Javascript source of the page, when available, and by testing sample passwords in each meter. Across all meters, general scoring categories included password length, the use of numbers, uppercase letters, and special characters, and the use of blacklisted words. Most meters updated dynamically as characters were typed.

Some meters had unique visual characteristics. Twitter’s bar was always green, while the warning text changed from red to green. Twitter offered phrases such as “Password could be more secure” and “Password is Perfect.” The site `mail.ru` had a three-segment bar with key-shaped segments, while `rakuten.co.jp` had a meter with a spring-like animation.

We found some inconsistencies across domains. Both `yahoo.com` and `yahoo.co.jp` used a meter with four segments; however, the scoring algorithm differed, as shown in Figure 3.1. Google used the same meter across all affiliated sites, yet its meter on `blogger.com` scored passwords more stringently.

3.3 Methodology

We conducted a two-part online study of password-strength meters, recruiting participants through Amazon’s Mechanical Turk crowdsourcing service (MTurk). Participants, who were paid 55 cents, needed to indicate that they were at least 18 years old and use a web browser with JavaScript enabled. Participants were assigned round-robin to one of 15 conditions, detailed in Section 3.3.2. We asked each participant to imagine that his or her main email provider had changed its password requirements, and that he or she needed to create a new password. We then asked the participant to create a password using the interface shown in Figure 3.2.

Passwords needed to contain at least eight characters, but there were no other requirements. The participant was told he or she would be asked to return in a few days to log in with the password. He or she then completed a survey about the password-creation experience and was asked to re-enter his or her password at the end.

Two days later, participants received an email through MTurk inviting them to return for a bonus payment of 70 cents. Participants were asked to log in again with their password and to take another survey about how they handled their password.

3.3.1 Password-Scoring Algorithms

Password-strength meters utilize a scoring function to judge the strength of a password, displaying this score through visual elements. We assigned passwords a score using heuristics including the password’s length and the character classes it contained. While alternative approaches to scoring have been proposed, as discussed in Chapter 2, judging a password only on heuristics obviates the need for a large, existing dataset of passwords and can be implemented quickly in Javascript. These heuristics were based on those we observed in the wild.

In our scoring system, a score of 0 points represented a blank password field, while a score of 100 points filled the meter and displayed the text “excellent.” We announced our only password-composition policy in

Figure 3.2: The password creation page. The password meter’s appearance and scoring varied by condition.

bold text to the participant as an “8-character minimum” requirement. However, we designed our scoring algorithm to assign passwords containing eight lowercase letters a score of 32, displaying “bad.” To receive a score of 100 in most conditions, participants needed to meet one of two policies identified as stronger in the literature [35, 108], which we term *Basic16* and *Comprehensive8*. Unless otherwise specified by the condition, passwords were assigned the larger of their *Basic16* and *Comprehensive8* scores. Thus, a password meeting either policy would fill the meter. Each keystroke resulted in a recalculation of the score and update of the meter.

The *Basic16* policy specifies that a password contain at least 16 characters, with no further restrictions. In our scoring system, the first 8 characters entered each received 4 points, while all subsequent characters received 8 points. Thus, passwords such as *aaaaaaaaaaaaaaaa*, *WdH5\$87T5c#hgfd&*, and *passwordpassword* would all fill the meter with scores of exactly 100 points.

The second policy, *Comprehensive8*, specifies that a password contain at least eight characters, including an uppercase letter, a lowercase letter, a digit, and a symbol. Furthermore, this password must not be in the OpenWall Mangled Wordlists, which is a cracking dictionary.¹ In our scoring system, 4 points were awarded for each character in the password, and an additional 17 points were awarded each for the inclusion of an uppercase character, a digit, and a symbol; 17 points were deducted if the password contained no lowercase letters. A second unique digit, symbol, or uppercase character would add an additional 8 points, while a third would add an additional 4 points. Passing the dictionary check conferred 17 points. Therefore, passwords such as *P4\$sword*, *gT7fas#g*, and *N!ck1ebk* would fill the meter with a score of exactly 100. In addition, passwords that were hybrids of the two policies, such as a 13-character password meeting *Comprehensive8* except containing no symbols, could also fill the meter.

¹<http://www.openwall.com/wordlists/>

3.3.2 Conditions

Our 15 conditions fall into four main categories. The first category contains the two conditions to which we compared the others: having no password meter and having a baseline password meter. Conditions in the next category differ from the baseline meter in only one aspect of visual presentation, but the scoring remains the same. In contrast, conditions in the third category have the same visual presentation as the baseline meter, but are scored differently. Finally, we group together three conditions that differ in multiple dimensions from the baseline meter. In addition, we collectively refer to *half-score*, *one-third-score*, *text-only half-score*, and *text-only half-score* as the *stringent* conditions. Each participant was assigned round-robin to one condition.

Control Conditions

- **No meter.** This condition, our control, uses no visual feedback mechanism. 26 of the Alexa Top 100 websites provided no feedback on password strength, and this condition allows us to isolate the effect of the visual feedback in our other conditions.
- **Baseline meter.** This condition represents our default password meter. The score is the higher of the scores derived from comparing the password to the Basic16 and Comprehensive8 policies, where a password meeting either policy fills the bar. The color changes from red to yellow to green as the score increases. We also provide a suggestion, such as “Consider adding a digit or making your password longer.” This condition is a synthesis of meters we observed in the wild.

Conditions Differing in Appearance

- **Three-segment.** This condition is similar to *baseline meter*, except the continuously increasing bar is replaced with a bar with three distinct segments, similar to meters from Google and Mediafire.
- **Green.** This condition is similar to *baseline meter*, except instead of changing color as the password score increases, the bar is always green, like Twitter’s meter.
- **Tiny.** This condition is similar to *baseline meter*, but with the meter’s size decreased by 50% horizontally and 60% vertically, similar to the size of Google’s meter.
- **Huge.** This condition is similar to *baseline meter*, but with the size of the meter increased by 50% horizontally and 120% vertically.
- **No suggestions.** This condition is similar to *baseline meter*, but does not offer suggestions for improvement.
- **Text-only.** This condition contains all of the text of *baseline meter*, but has no visual bar graphic.

Conditions Differing in Scoring

- **Half-score.** This condition is similar to *baseline meter*, except that the password’s strength is displayed as if it had received half the rating. A password that would fill the *baseline meter* meter only fills this condition’s meter half way, allowing us to study nudging the participant toward a stronger password. A

password with 28 characters, or one with 21 characters that included five different uppercase letters, five different digits, and five different symbols, would fill this meter.

- **One-third-score.** This condition is similar to *half-score*, except that the password's strength is displayed as if it had received one-third the rating. A password that would fill the *baseline meter* meter only fills one-third of this condition's meter. A password containing 40 characters would fill this meter.
- **Nudge-16.** This condition is similar to *baseline meter*, except that only the password score for the Basic16 policy is calculated, allowing us to examine nudging the user toward a specific policy.
- **Nudge-comp8.** As with *nudge-16*, this condition is similar to *baseline meter*, except that only the password score for Comprehensive8 is calculated.

Conditions Differing in Multiple Ways

- **Text-only half-score.** As with *text-only*, this condition contains all of the text of *baseline meter*, yet has no bar. Furthermore, like *half-score*, the password's strength is displayed as if it had received only half the score.
- **Bold text-only half-score.** This condition mirrors *text-only half-score*, except the text is bold.
- **Bunny.** In place of a bar, the password score is reflected in the speed at which an animated Bugs Bunny dances. When the score is 0, he stands still. His speed increases with the score; at a score of 100, he dances at 20 frames per second; at a score of 200, he reaches his maximum of 50 frames per second. This condition explores a visual feedback mechanism other than a traditional bar.

3.3.3 Mechanical Turk

Many researchers have examined using MTurk workers for human-subjects research and found it to be a convenient source of high-quality data [33, 57, 105, 176]. MTurk enables us to have a high volume of participants create passwords, on a web site we control, with better population diversity than would be available in an on-campus laboratory environment [33]. MTurk workers are also more educated, more technical, and younger than the general population [99].

3.3.4 Statistical Tests

All statistical tests use a significance level of $\alpha = .05$. For each variable, we ran an omnibus test across all conditions. We ran pairwise contrasts comparing each condition to our two control conditions, *no meter* and *baseline meter*. In addition, to investigate hypotheses about the ways in which conditions varied, we ran planned contrasts comparing *tiny* to *huge*, *nudge-16* to *nudge-comp8*, *half-score* to *one-third-score*, *text-only* to *text-only half-score*, *half-score* to *text-only half-score*, and *text-only half-score* to *bold text-only half-score*. If a pairwise contrast is not noted as significant in the results section, it was not found to be statistically significant. To control for Type I error, we ran contrasts only where the omnibus test was significant. Further, we corrected contrasts for multiple testing, accounting for the previous contrasts. We applied multiple testing correction to the p-values of the omnibus tests when multiple tests were run on similar variables, such as the Likert response variables measuring user attitudes.

We analyzed quantitative data using Kruskal-Wallis for the omnibus cases and Mann-Whitney U for the pairwise cases. These tests, identified in our results as K-W and MWU, respectively, are analogues of the ANOVA and t -tests without the assumption of normality. We analyze categorical data for equality of proportions with χ^2 tests for both the omnibus and pairwise cases. All multiple testing correction used the Holm-Bonferroni method, indicated as HC.

3.3.5 Calculating Guess Numbers

We evaluated the strength of passwords created in each condition by simulating a guessing attack using a probabilistic context-free grammar [197], which had been considered to be the most accurate password-guessing approach at the time of our experiment [54, 205]. This approach allowed us to approximate passwords' resistance to automated cracking. Using a password guess calculator similar to that used by Kelley et al. [104], we calculate the guessability of passwords in three different attack scenarios. This calculator simulates the password-cracking algorithm devised by Weir et al. [197], which makes guesses based on the structures, digits, symbols, and alphabetic strings in its training data. The calculator was set to only consider guesses with minimum length 8. For training, we used several "public" datasets, including leaked sets of cracked passwords. In Section 3.6.2, we discuss ethical issues of using leaked data.

Training data included 40 million passwords from the OpenWall Mangled Wordlist [137], 32 million leaked passwords from the website RockYou [187], and about 47,000 passwords leaked from MySpace [150]. We augmented the training data with all strings harvested from the Google Web Corpus [83], resulting in a dictionary of 14 million alphabetic strings.

In the *weak attacker* scenario, we consider an attacker with limited computational resources who can make 500 million (5×10^8) guesses. In the *medium attacker* scenario, we consider an attacker with greater resources who can make 50 billion (5×10^{10}) guesses. Finally, in the *strong attacker* scenario, we examine what percentage of passwords would have been guessed within the first 5 trillion (5×10^{12}) guesses. As discussed in Section 2.1, mapping a number of guesses to wall-clock time depends heavily on the hash function used, as well as the attacker's resources. For passwords hashed using bcrypt [143] with a large number of iterations, even our model of the weak attacker might represent an infeasible attack. For passwords hashed using MD5, however, our model of the strong attacker is likely possible on a single commodity laptop [166, 171].

3.4 Results

From January to April 2012, 2,931 people completed the initial task, and 2,016 of these subjects returned for the second part of the study. We begin our evaluation by comparing characteristics of passwords created in each condition, including their length and the character classes used. Next, we simulate a cracking algorithm to evaluate what proportion of passwords in each condition would be cracked by adversaries of varying strength. We then examine the usability of these passwords, followed by data about the process of password creation. Finally, we discuss participant demographics and potential interaction effects. In Section 3.5, we provide additional results on participants' attitudes and reactions.

Table 3.1: A comparison across conditions of the characteristics of passwords created: the *length*, number of *digits*, number of *uppercase* letters, and number of *symbols*. For each metric, we present the mean, the standard deviation (SD), and the median. Conditions that differ significantly from *no meter* are indicated with an asterisk (*). Conditions that differ significantly from *baseline meter* are indicated with a dagger (†).

Metric	no meter (*)	baseline meter (†)	three-segment	green	tiny	huge	no suggestion	text-only	half-score	one-third-score	nudge-16	nudge-comp8	text-only half	bold text-only half	bunny
Length		*	*	*	*	*	*	†	*,†	*	*,†	*	*	*	*
Mean	10.4	12.0	11.5	11.3	11.4	11.6	11.4	10.9	14.9	14.3	13.0	11.6	12.3	13.0	11.2
SD	2.9	3.7	3.8	3.6	3.2	3.3	3.5	3.2	7.3	8.1	3.7	3.5	6.1	5.5	3.1
Median	9	11	10	10	11	11	11	10	12.5	12	12	11	10.5	11	10
Digits									*	*		*		*	*
Mean	2.4	2.7	2.8	2.6	2.7	2.5	3.0	2.5	3.3	3.4	3.2	3.3	3.1	3.2	3.3
SD	2.8	2.6	2.6	2.5	2.3	2.2	2.8	2.3	3.0	3.2	3.4	2.8	3.5	3.0	3.0
Median	2	2	2	2	3	2	2	2	3	3	3	3	2	3	3
Uppercase									*	*				*,†	
Mean	0.8	0.8	0.9	0.8	0.6	1.0	0.7	0.9	1.5	1.4	0.5	0.8	1.2	1.5	0.8
SD	2.0	1.8	1.7	2.0	1.4	2.3	1.5	1.7	3.4	3.2	1.3	1.5	2.2	2.5	1.5
Median	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5
Symbols									*	*			*	*	
Mean	0.3	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.8	1.0	0.5	0.5	0.6	0.9	0.4
SD	0.7	1.0	0.8	1.1	0.7	0.8	0.8	0.7	1.6	2.7	1.3	1.0	1.2	1.7	0.7
Median	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3.4.1 Password Characteristics

The presence of almost any password meter significantly increased password length. In conditions that scored passwords stringently, the meter also increased the use of digits, uppercase letters, and symbols. The length of the passwords varied significantly across conditions, as did the number of digits, uppercase characters, and symbols contained in each password (HC K-W, $p < .001$). Table 3.1 details password characteristics.

Length The presence of any password meter except *text-only* resulted in significantly longer passwords. Passwords created with *no meter* had a mean length of 10.4, and passwords created in the *text-only* condition had a mean length of 10.9, which was not significantly different. Passwords created in the thirteen other conditions with meters, with mean length ranging from 11.3 to 14.9 characters, were significantly longer than in *no meter* (HC MWU, $p \leq .014$).

Furthermore, passwords created in *half-score*, with mean length 14.9, and in *nudge-16*, with mean length 13.0, were significantly longer than those created in *baseline meter*, which had mean length 12.0 (HC MWU, $p \leq .017$). On the other hand, passwords created in , with mean length 10.9, were significantly shorter than in *baseline meter* (HC MWU, $p = .015$). Although passwords created in *one-third-score* had mean length 14.3, they had a high standard deviation (8.1) and did not differ significantly from *baseline meter*.

Digits, Uppercase Characters, and Symbols Compared to *no meter*, passwords in five conditions contained significantly more digits: *half-score*, *one-third-score*, *nudge-comp8*, *bold text-only half-score*, and *bunny* (HC MWU, $p < .028$). In each of these five conditions, passwords contained a mean of 3.2 to 3.4 digits, compared to 2.4 digits in *no meter*. The mean number of digits in all other conditions ranged from 2.5 to 3.1.

In three of these conditions, *half-score*, *one-third-score*, and *bold text-only half-score*, passwords on average contained both more uppercase letters and more symbols (HC MWU, $p < .019$) than in *no meter*. In these three conditions, the mean number of uppercase characters ranged from 1.4 to 1.5 and the mean number of symbols ranged from 0.8 to 1.0, whereas passwords created in *no meter* contained a mean of 0.8 uppercase characters and 0.3 symbols. Furthermore, passwords created in *text-only half-score* had significantly more symbols, 0.6 on average, than *no meter*, although the mean number of digits did not differ significantly.

While most participants used digits in their passwords, uppercase characters and symbols were not as common. In nearly all conditions, the majority of participants did not use any uppercase characters in their password despite the meter's prompts to do so. Less than half of participants in any condition used symbols.

3.4.2 Password Guessability

We evaluated the strength of passwords based on their “guessability,” which is the number of guesses an adversary would need to guess that password. As described in Section 3.3.5, We considered three adversaries: a *weak attacker* with limited resources who makes 500 million (5×10^8) guesses, a *medium attacker* who makes 50 billion (5×10^{10}) guesses, and a *strong attacker* who makes 5 trillion (5×10^{12}) guesses. Table 3.2 and Figure 3.3 present the proportion of passwords cracked by condition.

We found that all conditions with password meters appeared to provide a small advantage against attackers of all three strengths. In all fourteen conditions with meters, the percentage of passwords cracked by all three adversaries was always smaller than in *no meter*, although most of these differences were not statistically significant. The only substantial increases in resistance to cracking were provided by the two stringent meters with visual bars, *half-score* and *one-third-score*.

A *weak adversary* cracked 21.0% of passwords in the *no meter* condition, which was significantly larger than the 5.8% of passwords cracked in the *half-score* condition and the 4.7% of passwords cracked in *one-third-score* (HC χ^2 , $p < 0.001$). Only 7.8% of passwords were cracked in *bunny*, which was also significantly less than in *no meter* (HC χ^2 , $p = 0.008$). Between 9.5% and 15.3% of passwords were cracked in all other conditions with meters, none of which were statistically significantly different than *no meter*.

In the *medium adversary* scenario, significantly more passwords were cracked in the *no meter* condition than in the *half-score* and *one-third-score* conditions (HC χ^2 , $p \leq 0.017$). 35.4% of the passwords in the *no meter* condition were cracked, compared with 19.5% of passwords in *half-score* and 16.8% of passwords in *one-third-score*. None of the other conditions differed significantly from *no meter*; between 23.7% and 34.4% of passwords were cracked in these conditions.

The *half-score* and *one-third-score* meters were significantly better than *no meter* against a *strong adversary*. In *no meter*, 46.7% of passwords were cracked, compared with 26.3% in *half-score* and 27.9% in *one-third-score* (HC χ^2 , $p \leq 0.005$). Between 33.7% and 46.2% of passwords in all other sets were cracked.

After the completion of the experiment, we ran additional conditions to explore how meters consisting of only a visual bar, without accompanying text, would compare to text-only conditions and conditions containing both text and visual features. Since this data was collected two months after the rest of our data, we do not include it in our main analyses. However, passwords created in these conditions performed similarly

Table 3.2: A comparison of the percentage of passwords in each condition cracked by weak (5×10^8 guesses), medium (5×10^{10} guesses), and strong adversaries (5×10^{12} guesses). Each cell contains the percentage of passwords cracked in that threat model. Conditions that differ significantly from *no meter* are indicated with an asterisk (*).

Adversary	no meter (*)	baseline meter (†)	three-segment	green	tiny	huge	no suggestion	text-only	half-score	one-third-score	nudge-16	nudge-comp8	text-only half	bold text-only half	bunny
Weak									*	*					*
% Cracked	21.0	11.1	10.3	12.0	10.7	9.6	11.0	15.1	5.8	4.7	15.3	10.3	9.5	11.4	7.8
Medium									*	*					
% Cracked	35.4	27.2	26.6	30.0	30.0	31.0	25.9	34.4	19.5	16.8	25.0	23.7	24.2	25.7	28.1
Strong									*	*					
% Cracked	46.7	39.4	39.4	45.5	42.1	41.6	39.3	46.2	26.3	27.9	33.7	39.2	34.7	35.6	40.1

to equivalent text-only conditions and strictly worse than equivalent conditions containing both a bar and text. For instance, a strong adversary cracked 48.3% of passwords created with the *baseline meter* bar without its accompanying text and 33.0% of passwords created with the *half-score* bar without its accompanying text.

3.4.3 Password Memorability and Storage

To gauge the memorability of the passwords subjects created, we considered the proportion of subjects who returned for the second day of our study, the ability of participants to enter their password both minutes after creation and a few days after creation, and the number of participants who either reported or were observed storing or writing down their password.

Of our participants, 2,016 (68.8%) returned and completed the second part of the study. The proportion of participants who returned did not differ significantly across conditions (χ^2 , $p=0.241$).

Between the 68.8% of participants who returned for the second part of the study and the 31.2% of participants who did not, there were no significant differences in the length of the passwords created, the number of digits their password contained, or the percentage of passwords cracked by a *medium* or *strong* attacker. However, the *weak* attacker cracked a significantly higher percentage of passwords created by subjects who did not return for the second part of the study than passwords created by participants who did return (HC χ^2 , $p<.001$). 14.5% of passwords created by subjects who did not return and 9.5% of passwords created by subjects who did return were cracked. Participants who returned for the second part of the study also had more uppercase letters and more symbols in their passwords (K-W, $p<.001$). Participants who returned had a mean of 1.0 uppercase letters and 0.6 symbols in their passwords, while those who did not had a mean of 0.8 uppercase letters and 0.5 symbols.

Participants' ability to recall their password also did not differ significantly between conditions, either minutes after creating their password (χ^2 , $p=0.236$) or at least two days later (χ^2 , $p=0.250$). In each condition, 93% or more of participants were able to enter their password correctly within three attempts minutes after

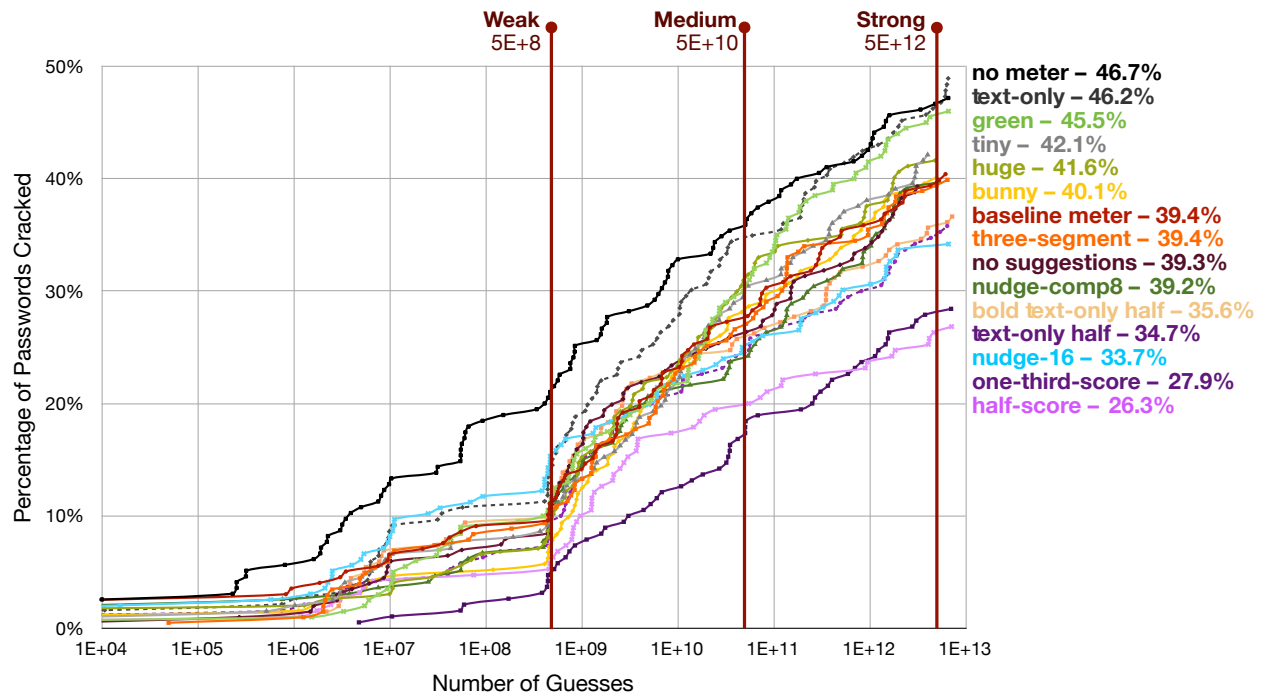


Figure 3.3: The percentage of passwords that were cracked in each condition. The x-axis, which is logarithmically scaled, indicates the number of guesses made by an adversary. The y-axis indicates the percentage of passwords in that condition cracked by that particular guess number.

creating the password. When they received an email two days later to return and log in with their password, between 77% and 89% of subjects in each condition were able to log in successfully within three attempts.

As an additional test of password memorability, we asked participants if they had written their password down, either electronically or on paper, or if they had stored their password in their browser. Furthermore, we captured keystroke data as they entered their password, which we examined for evidence of pasting in the password. If a participant answered affirmatively to either question or pasted the password into the password field, he or she was considered as having stored the password. Overall, 767 participants (38.0% of those who returned) reported that they had stored or written down their password. 78 of these 767 participants were also observed to have pasted in their password. An additional 32 participants (1.6%) were observed pasting in their password even though they had said they had not stored it.

The proportion of participants storing their passwords did not differ across conditions (χ^2 , $p=0.364$). In each condition, between 33% and 44% of participants were observed pasting in a password or reported writing down or storing their password.

3.4.4 Password Creation Process

Based on analysis of participants' keystrokes during password creation, we found that participants behaved differently in the presence of different password meters. Password meters seemed to encourage participants to reach milestones, such as filling the meter or no longer having a "bad" or "poor" password. The majority

Table 3.3: A comparison across conditions of password creation: the percentage of participants who completely *filled the password meter* or equivalently scored “excellent” in text-only conditions, the percentage of participants whose password received a score of “*bad*” or “*poor*”, the *time* of password creation (first to last keystroke), the number of *deletions* (characters deleted after being entered) in the password creation process, the percentage of participants who *changed their password* (initially entering a valid password containing at least 8 characters before completely deleting it and entering a different password), and the *edit distance* between the initial password entered and the final password saved, normalized by the length of the final password. Conditions differing significantly from *no meter* are indicated with an asterisk (*), while those differing significantly from *baseline meter* are marked with a dagger (†).

Metric	no meter (*)	baseline meter (†)	three-segment	green	tiny	huge	no suggestion	text-only	half-score	one-third-score	nudge-16	nudge-comp8	text-only half	bold text-only half	bunny
Filled Meter		*	*	*	*	*	*	*	*,†	*,†	†	*	*,†	*,†	*
% participants	(25.1)	48.5	53.2	42.5	48.2	52.8	37.3	46.2	9.0	1.6	24.5	46.9	3.2	5.0	48.4
“Bad” / “Poor”		*	*	*	*	*	*	*	*,†	*,†	†	*	*,†	*,†	*
% participants	(24.1)	9.1	10.3	12.0	9.6	8.1	7.5	13.4	58.4	93.7	37.2	9.8	76.3	67.8	8.3
Time (seconds)									*,†	*,†	*		*	*,†	*
Mean	19.9	23.5	22.7	21.0	21.5	25.8	24.7	24.8	60.8	59.8	33.1	26.6	38.5	57.1	30.4
SD	28.4	22.7	23.6	22.2	23.2	28.9	36.6	29.4	75.7	84.9	33.2	30.2	49.8	150.0	36.9
Median	10.6	15.6	14.0	13.7	13.1	14.7	13.0	14.0	39.1	34.2	23.2	13.8	23.5	32.8	19.8
Deletions									*,†	*,†	*,†		*,†	*,†	*
Mean	5.3	6.2	7.5	5.8	6.2	7.8	5.5	7.8	23.8	22.9	12.1	8.1	14.6	23.1	10.7
SD	10.7	10.2	13.7	12.4	10.8	11.3	8.4	11.9	29.0	26.6	16.2	13.3	19.3	26.9	17.2
Median	0	0	0	0	1	2	0	0	13.5	13	8	1	8	13.5	5
Changed PW									*,†	*,†	*,†		*,†	*,†	*,†
% participants	14.4	18.7	25.6	16.5	23.9	23.4	25.9	25.8	52.6	52.6	40.3	24.7	35.8	51.0	34.9
Edit Dist.									*,†	*,†	*,†		*,†	*,†	*,†
Mean	0.10	0.09	0.47	0.09	0.14	0.12	0.15	0.17	0.37	0.45	0.27	0.15	0.27	0.35	0.28
SD	0.29	0.23	4.84	0.28	0.30	0.31	0.37	0.36	0.42	1.22	0.38	0.36	0.43	0.47	0.70
Median	0	0	0	0	0	0	0	0	0.15	0.11	0	0	0	0.08	0

of participants who saw the most stringent meters changed their mind partway into password creation, erasing what they had typed and creating a different password. Table 3.3 details this data.

Most participants created a new password for this study, although some participants reused or modified an existing password. Between 57% and 71% of subjects in each condition (63% overall) reported creating an entirely new password, between 15% and 26% (21% overall) reported modifying an existing password, between 9% and 19% (14% overall) reported reusing an existing password, and fewer than 4% (2% overall) used some other strategy. The proportion of participants reporting each behavior did not vary significantly across conditions (χ^2 , $p=.876$).

Participants in *nudge-16*, *bunny*, and all four stringent conditions took longer to create their password than those in *no meter* (HC χ^2 , $p<.001$). The mean password creation time, measured from the first to the last keystroke in the password box, was 19.9 seconds in the *no meter* condition. It was 60.8 seconds for *half-score*, 59.8 seconds for *one-third-score*, 57.1 seconds for *bold text-only half-score*, 38.5 seconds for

text-only half-score, 33.1 seconds for *nudge-16*, and 30.4 seconds for *bunny*. Compared also to the *baseline meter* meter, where mean password creation time was 23.5 seconds, participants took significantly longer in the *half-score*, *one-third-score*, and *bold text-only half-score* conditions (HC χ^2 , $p < .008$). The mean time of password creation ranged from 21.0 to 26.6 seconds in all other conditions.

Password meters encouraged participants both to avoid passwords that the meter rated “bad” or “poor” and to create passwords that filled the meter. Had there been a password meter, 24.1% of passwords created in *no meter* would have scored “bad” or “poor,” which was significantly higher than the 12.0% or fewer of passwords in all non-stringent conditions other than *no suggestions* and *nudge-16* rated “bad” or “poor” (HC χ^2 , $p \leq 0.035$). Had *no meter* contained a password meter, 25.1% of passwords created would have filled the meter. A larger proportion of passwords in all non-stringent conditions other than *no suggestions* and *nudge-16* filled the meter (HC χ^2 , $p \leq 0.006$). In each of these conditions, 42.5% or more of the passwords filled the meter. While the proportion of passwords in *nudge-16* and the four stringent conditions reaching these thresholds was significantly lower than *baseline meter*, the proportions would have been higher than *baseline meter* were the *baseline meter* scoring algorithm used in those conditions.

During the password creation process, participants in all four stringent conditions, as well as in *nudge-16*, made more changes to their password than in *no meter* or *baseline meter*. We considered the number of deletions a participant made, which we defined as the number of characters that were inserted into the password and then later deleted. In the four stringent conditions and in *nudge-16*, the mean number of deletions by each participant ranged from 12.1 to 23.8 characters. In contrast, significantly fewer deletions were made in *no meter*, with a mean of 5.3 deletions, and *baseline meter*, with a mean of 6.2 deletions (HC MWU, $p < 0.001$). The *bunny* condition, with a mean of 10.7, also had significantly more deletions than *no meter* (HC MWU, $p = 0.004$).

We further analyzed the proportion of participants who changed their password, finding significantly more changes occurring in the stringent conditions, as well as in *nudge-16* and *bunny*. Some participants entered a password containing eight or more characters, meeting the stated requirements, and then completely erased the password creation box to start over. We define the *initial password* to be the longest such password containing eight or more characters that a participant created before starting over. Similarly, we define the *final password* to be the password the participant eventually saved. We considered participants to have changed their password if they created an initial password, completely erased the password field, and saved a final password that differed by one edit or more from their initial password.

More than half of the participants in *half-score*, *one-third-score*, and *bold text-only half-score* changed their password during creation. Similarly, between 34.9% and 40.3% of *nudge-16*, *text-only half-score*, and *bunny* participants changed their password. The proportion of participants in these six conditions who changed their password was greater than the 14.4% of *no meter* participants and 18.7% of *baseline meter* participants who did so (HC χ^2 , $p \leq .010$). Across all conditions, only 7.7% of final passwords consisted of the initial password with additional characters added to the end; in a particular condition, this percentage never exceeded 16%.

These changes in the password participants were creating resulted in final passwords that differed considerably from the initial password. We assigned an edit distance of 0 to all participants who did not change their password. For all other participants, we computed the Levenshtein distance between the initial and final password, normalized by the length of the final password. The mean normalized edit distance between initial and final passwords ranged from 0.27 to 0.45 in the six aforementioned conditions, significantly greater than *no meter*, with a mean of 0.10, and *baseline meter*, with a mean of 0.09 (HC MWU, $p < .003$).

We also compared the guessability of the initial and final passwords for participants whose initial password, final password, or both were guessed by the strong adversary. 86.1% of the 43 such changes in *half-score* resulted in a password that was harder to guess, as did 83.8% of 37 such changes in *text-only half-score*. In contrast, 50% of 18 such changes in *baseline meter* and between 56.7% and 76.7% such changes in all other conditions resulted in passwords that were harder to guess. However, these differences were not statistically significant.

3.4.5 Participant Demographics

Participants ranged in age from 18 to 74 years old, and 63% percent reported being male and 37% female.² 40% percent reported majoring in or having a degree or job in computer science, computer engineering, information technology, or a related field; 55% said they did not. Participants lived in 96 different countries, with most from India (42%) and the United States (32%). Because many of our password meters used a color scheme that includes red and green, we asked about color-blindness; 3% of participants reported being red-green color-blind, while 92% said they were not, consistent with the general population [159].

The number of subjects in each condition ranged from 184 to 202, since conditions were not reassigned if a participant did not complete the study. There were no statistically significant differences in the distribution of participants' gender, age, technology background, or country of residence across experimental conditions.

However, participants who lived in different countries created different types of passwords. We separated participants into three groups based on location: United States, India, and "the rest of the world." Indian subjects' passwords had mean length 12.2, U.S. subjects' passwords had mean length 11.9, and all other subjects' passwords had mean length 12.1 (HC K-W, $p=0.002$). Furthermore, Indian subjects' passwords had a mean of 0.9 uppercase letters, and both U.S. subjects' and all other subjects' passwords had a mean of 1.0 uppercase letters (HC K-W, $p<0.001$). While the percentage of passwords cracked by a *weak* or *medium* attacker did not differ significantly between the three groups, a lower percentage of the passwords created by Indian participants than those created by American participants was cracked by a *strong* adversary (HC χ^2 , $p=.032$). 42.3% of passwords created by subjects from the U.S., 35.5% of passwords created by subjects from India, and 38.8% of passwords created by subjects from neither country were cracked by a *strong* adversary. However, the guessing algorithm was trained on sets of leaked passwords from sites based in the U.S., which may have biased its guesses.

3.5 Participants' Attitudes and Perceptions

We asked participants to rate their agreement on a Likert scale with fourteen statements about the password creation process, such as whether it was fun or annoying, as well as their beliefs about the password meter they saw. We also asked participants to respond to an open-ended prompt about how the password meter did or did not help. We begin by reporting participants' survey responses, which reveal annoyance among participants in the stringent conditions. The *one-third-score* condition and text-only stringent conditions also led participants to believe the meter gave an incorrect score and to place less importance on the meter's rating. The distribution of responses to select survey questions is shown in Figure 3.4. We then present participants' open-ended responses, which illuminate strategies for receiving high scores from the meter.

²We offered the option not to answer demographic questions, so percentages may sum to less than 100.

3.5.1 Attitudes Toward Password Meters

In a survey immediately following password creation, a higher percentage of participants in the stringent conditions found password creation to be annoying or difficult than those in *baseline meter*. A larger proportion of subjects in the four stringent conditions than in either the *no meter* or *baseline meter* conditions agreed that creating a password in this study was annoying (HC χ^2 , $p \leq .022$). Similarly, a higher percentage of subjects in the *half-score* and *bold text-only half-score* found creating a password difficult than in either the *no meter* or *baseline meter* conditions (HC χ^2 , $p \leq .012$). Creating a password was also considered difficult by a higher percentage of subjects in *one-third-score* and *text-only half-score* than in *baseline meter* (HC χ^2 , $p \leq .003$), although these conditions did not differ significantly from *no meter*.

Participants in the stringent conditions also found the password meter itself to be annoying at a higher rate. A higher percentage of subjects in all four stringent conditions than in *baseline meter* agreed that the password-strength meter was annoying (HC χ^2 , $p \leq .007$). Between 27% and 40% of participants in the four stringent conditions, compared with 13% of *baseline meter* participants, found the meter annoying.

Participants in the two stringent conditions without a visual bar felt that they did not understand how the meter rated their password. 38% of *text-only half-score* and 39% of *bold text-only half-score* participants agreed with the statement, "I do not understand how the password strength meter rates my password," which was significantly greater than the 22% of participants in *baseline meter* who felt similarly (HC χ^2 , $p \leq .015$). 32% of *half-score* participants and 34% of *one-third-score* participants also agreed, although these conditions were not statistically significantly different than *baseline meter*.

The *one-third-score* condition and both text-only stringent conditions led participants to place less importance on the meter. A smaller proportion of *one-third-score*, *text-only half-score*, and *bold text-only half-score* participants than *baseline meter* subjects agreed, "It's important to me that the password-strength meter gives my password a high score" (HC χ^2 , $p \leq .021$). 72% of *baseline meter* participants, yet only between 49% and 56% of participants in those three conditions, agreed. In all other conditions, between 64% and 78% of participants agreed. Among these conditions was *half-score*, in which 68% of participants agreed, significantly more than in *one-third-score* (HC χ^2 , $p = .005$).

More participants in those same three conditions felt the meter's score was incorrect. 42-47% of *one-third-score*, *text-only half-score*, and *bold text-only half-score* participants felt the meter gave their password an incorrect score, significantly more than the 21% of *baseline meter* participants who felt similarly (HC χ^2 , $p \leq .001$). Between 12% and 33% of participants in all other conditions, including *half-score*, agreed; these conditions did not differ significantly from *baseline meter*.

3.5.2 Participant Motivations

Participants' open-ended responses to the prompt, "Please explain how the password strength meter helped you create a better password, or explain why it was not helpful," allowed participants to explain their thought process in reaction to the meter and their impressions of what makes a good password.

Reactions to the Password Meter

Some participants noted that they changed their behavior in response to the meter, most commonly adding a different character class to the end of the password. One participant said the meter "motivated [him] to use symbols," while another "just started adding numbers and letters to the end of it until the high score

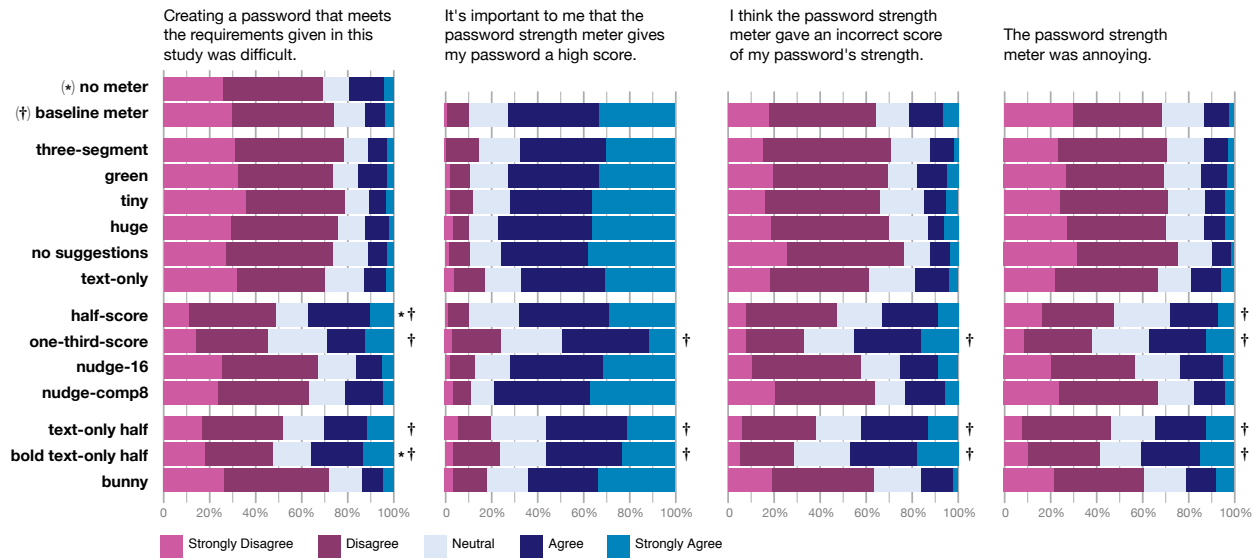


Figure 3.4: Participants’ agreement or disagreement with the statement above each chart. Each color represents the proportion of participants in that condition who expressed a particular level of agreement or disagreement with the statement. Conditions in which the proportion of participants agreeing with a statement differed significantly from *no meter* are indicated with an asterisk (*), while those that differed significantly from *baseline meter* are marked with a dagger (†). Participants in *no meter* did not respond to questions about password meters.

was reached.” Participants also said that the meter encouraged or reminded them to use a more secure password. One representative participant explained, “It kept me from being lazy when creating my password. [I] probably would not have capitalized any letters if not for the meter.”

Other participants chose a password before seeing the meter, yet expressed comfort in receiving validation. For instance, one representative participant noted, “The password I ultimately used was decided on before hand. However, whilst I was typing and I saw the strength of my password increase and in turn felt reassured.”

However, a substantial minority of participants explained that they ignore password meters, often because they believe these meters discourage passwords they can remember. One representative participant said, “No matter what the meter says, I will just use the password I chose because it’s the password I can remember. I do not want to get a high score for the meter and in the end have to lose or change my password.” Some participants expressed frustration with meters for not understanding this behavior. For instance, one participant explained, “I have certain passwords that I use because I can remember them easily. I hate when the meter says my password is not good enough— it’s good enough for me!”

Participants also reported embarrassment at poor scores, fear of the consequences of having a weak password, or simply a desire to succeed at all tasks. One participant who exemplifies the final approach said, “I wanted to make my password better than just ‘fair,’ so I began to add more numbers until the password-strength meter displayed that my password was ‘good.’ I wanted to create a strong password because I’m a highly competitive perfectionist who enjoys positive feedback.” In contrast, another participant stated, “Seeing a password strength meter telling me my password is weak is scary.”

Impressions of Password Strength

Participants noted impressions of password strength that were often based on past experiences. However, the stringent conditions seemed to violate their expectations.

Most commonly, subjects identified a password containing different character classes as strong. One representative participant said, "I am pretty familiar with password strength meters, so I knew that creating a password with at least 1 number/symbol and a mixture of upper and lower case letters would be considered strong." Participants also had expectations for the detailed algorithm with which passwords were scored, as exemplified by a participant who thought the meter "includes only English words as predictable; I could have used the Croatian for 'password123' if I wanted."

The stringent conditions elicited complaints from participants who disagreed with the meter. For example, one participant was unsure how to receive a good score, saying, "No matter what I typed, i.e. how long or what characters, it still told me it was poor or fair." Another participant lamented, "Nothing was good enough for it!" Some participants questioned the veracity of the stringent meters. For instance, a *one-third-score* participant said, "I have numbers, upper/lower case, and several symbols. It's 13 characters long. It still said it was poor. No way that it's poor." Other participants reused passwords that had received high scores from meters in the wild, noting surprise at the stringent meters' low scores. Some participants became frustrated, including one who said *one-third-score* "was extremely annoying and made me want to punch my computer."

The *bunny* received mixed feedback from participants. Some respondents thought that it sufficed as a feedback mechanism for passwords. For instance, one subject said, "I think it was just as helpful as any other method I have seen for judging a password's strength...I do think the dancing bunny is much more light-hearted and fun." However, other participants found the more traditional bar to be more appropriate, including one who said *bunny* "was annoying, I am not five [years old]."

Goals for the Password Meter

Participants stated two primary goals they adopted while using the password meter. Some participants aimed to fill the bar, while others hoped simply to reach a point the meter considered not to be poor. Those participants who aimed to fill the bar noted that they continued to modify their password until the bar was full, citing as motivation the validation of having completed their goal or their belief that a full bar indicated high security.

Participants employing the latter strategy increased the complexity of their password until the text "poor" disappeared. One participant noted, "It gave me a fair score, so I went ahead with the password, but if it would have given me a low score I would not have used this password." A number of participants noted that they didn't want to receive a poor rating. One representative participant said, "I didn't want to have poor strength, while I didn't feel I needed something crazy."

Some participants also identified the bar's color as a factor in determining when a password was good enough. Some participants hoped to reach a green color, while others simply wanted the display not to be red. One participant aiming towards a green color said, "I already chose a fairly long password, but I changed a letter in it to an uppercase one to make it turn green." Another participant expressed, "I knew that I didn't want to be in the red, but being in the yellow I thought was ok."

3.6 Discussion

We discuss our major findings relating to the design of effective password meters. We also address our study's ethical considerations, limitations, and future work.

3.6.1 Effective Password Meters

At a high level, we found that users do change their behavior in the presence of a password-strength meter. Seeing a password meter, even one consisting of a dancing bunny, led users to create passwords that were longer. Although the differences were generally not statistically significant, passwords created in all 14 conditions with password meters were cracked at a lower rate by adversarial models of different strengths.

However, the most substantial changes in user behavior were elicited by stringent meters. These meters led users to add additional character classes and make their password longer, leading to significantly increased resistance to a guessing attack. Furthermore, more users who saw stringent meters changed the password they were creating, erasing a valid password they had typed and replacing it with one that was usually harder to crack.

Unfortunately, the scoring systems of meters we observed in the wild were most similar to our non-stringent meters. This result suggests that meters currently in use on popular websites are not aggressive enough in encouraging users to create strong passwords. However, if all meters a user encountered were stringent, he or she might habituate to receiving low scores and ignore the meter, negating any potential security benefits.

There seems to be a limit to the stringency that a user will tolerate. In particular, the *one-third-score* meter seemed to push users too hard; *one-third-score* participants found the meter important at a lower rate and thought the meter to be incorrect at a higher rate, yet their passwords were comparable in complexity and cracking-resistance to those made by *half-score* participants. If meters are too stringent, users just give up.

Tweaks to the password meter's visual display did not lead to significant differences in password composition or user sentiment. Whether the meter was tiny, monochromatic, or a dancing bunny did not seem to matter. However, an important factor seemed to be the combination of text and a visual indicator, rather than only having text or only having a visual bar. Conditions containing text without visual indicators, run as part of our experiment, and conditions containing a visual bar without text, run subsequently to the experiment we focus on here, were cracked at a higher rate and led to less favorable user sentiment than conditions containing a combination of text and a visual indicator.

In the presence of password-strength meters, participants changed the way they created a password. For instance, the majority of participants in the stringent conditions changed their password during creation. Meters seemed to encourage participants to create a password that filled the meter. If that goal seemed impossible, participants seemed content to avoid passwords that were rated "bad" or "poor." In essence, the password meter functions as a progress meter, and participants' behavior echoed prior results on the effects progress meters had on survey completion [45]. Meters whose estimates of password strength mirrored participants' expectations seemed to encourage the creation of secure passwords, whereas very stringent meters whose scores diverged from expectations led to less favorable user sentiment and an increased likelihood that a participant would abandon the task of creating a strong password.

We also found many users to have beliefs regarding how to compose a strong password, such as including different character classes. Because users' understanding of password strength appears at least partially based

on experience with real-world password-strength meters and password-composition policies, our results suggest that wide-scale deployment of stringent meters may train users to create stronger passwords.

3.6.2 Ethical Considerations

We calculated our guessability results by training a guess-number calculator on sets of passwords that are publicly and widely available, but that were originally gathered through illegal cracking and phishing attacks. It can be argued that data acquired illegally should not be used at all by researchers, and so we want to address the ethical implications of our work. We use the passwords alone, excluding usernames and email addresses. We neither further propagate the data, nor does our work call significantly greater attention to the data sets, which have been used in several scientific studies [26, 54, 104, 196, 197]. As a result, we believe our work causes no additional harm to the victims, while offering potential benefits.

3.6.3 Limitations

One potential limitation of our study is its ecological validity. Subjects created passwords for an online study, and they were not actually protecting anything valuable with those passwords. Furthermore, one of the primary motivations for part of the MTurk population is financial compensation [99], which differs from real-world motivations for password creation. Outside of a study, users would create passwords on web pages with the logos and insignia of companies they might trust, perhaps making them more likely to heed a password meter's suggestions. On the other hand, subjects who realize they are participating in a password study may be more likely to think carefully about their passwords and pay closer attention to the password meter than they otherwise would. We did ask participants to imagine that they were creating passwords for their real email accounts, which prior work has shown to result in stronger passwords [108]. Because our results are based on comparing passwords between conditions, we believe our findings about how meters compare to one another can be applied outside our study.

Our study used a password-cracking algorithm developed by Weir et al. [197] in a guess-number calculator implemented by Kelley et al. [104] to determine a password's guessability. We did not experiment with a wide variety of cracking algorithms since prior work [104, 196, 205] has found that this algorithm outperformed alternatives including John the Ripper. Nevertheless, the relative resistance to cracking of the passwords we collected may differ depending on the choice of cracking algorithm. We investigated this question further in Chapter 4, finding that heterogeneous password sets' relative resistance to cracking is fairly robust to the password-cracking algorithm chosen.

Furthermore, the data we used to train our cracking algorithm was not optimized to crack passwords of particular provenance. For instance, passwords created by participants from India were the most difficult to crack. The data with which we trained our guessing algorithm was not optimized for participants creating passwords in languages other than English, which may have led to fewer of these passwords being cracked; prior work by Kelley et al. [104] found that the training set has a substantial effect on the success of the guessing algorithm we used.

3.7 Conclusion

We conducted the first large-scale study of password-strength meters, finding that meters did affect user behavior and security. Meters led users to create longer passwords. However, unless the meter scored stringently, the resulting passwords were only marginally more resistant to password cracking attacks.

Meters that rated passwords stringently led users to make significantly longer passwords that included more digits, symbols, and uppercase letters. These passwords were not observed to be less memorable or usable, yet they were cracked at a lower rate by simulated adversaries making 500 million, 50 billion, and 5 trillion guesses. The most stringent meter annoyed users, yet did not provide security benefits beyond those provided by slightly less stringent meters. The combination of a visual indicator and text outperformed either in isolation. However, the visual indicator's appearance did not appear to have a substantial impact.

Despite the added strength that these more stringent meters convey, we observed many more lenient meters deployed in practice. Our findings suggest that, so long as they are not overly onerous, employing more rigorous meters would increase security. Improving password meters, such as through the data-driven password meter presented in Chapter 7, therefore has the potential to help users create better passwords on a large scale.

Chapter 4

Understanding Biases in Modeling Password Cracking

4.1 Introduction

While I demonstrated in the previous chapter that password-strength meters using even very basic heuristics to score passwords can lead users to create passwords that better resist one particular password-guessing attack, there are many such password-guessing attacks. In this chapter, I report on our investigation of the accuracies and biases of modeling different conceptual approaches to guessing passwords, each in multiple configurations, to better understand how researchers can model a password's resistance to adversarial guessing. In addition, we compare the types of models a researcher can create to password guessing by a human expert in password forensics, finding that the more nuanced and better-configured models we propose can serve as a conservative proxy for a guessing attack by a professional.

A key aspect of improving password security is making passwords more computationally expensive to guess during offline attacks. Cracking tools like the GPU-based oclHashcat [166] and distributed cracking botnets [48, 55] enable attackers to make 10^{14} guesses in hours if passwords are hashed using fast hash functions like MD5 or NTLM. These advances are offset by the development of hash functions like bcrypt [143] and scrypt [138], which make attacks more difficult by requiring many iterations or consuming lots of memory.

Unfortunately, users often create predictable passwords [31, 94], which attackers can guess quickly even if the passwords are protected by a computationally expensive hash function. In some cases, predictable passwords are a rational coping strategy [150, 169]; in other cases, users are simply unsure whether a password is secure [184]. System administrators encourage strong passwords through password-composition policies and password-strength meters. The design and effectiveness of such mechanisms hinges on robust metrics to measure how difficult passwords are to guess.

In recent years, traditional entropy metrics have fallen out of favor because they do not reflect how easily a password can be cracked in practice [22, 104, 196]. It has instead become common to measure password strength by running or simulating a particular cracking algorithm, parameterized by a set of training data [23, 104, 196]. This approach has two main advantages. First, it calculates the guessability of each

Previously published as Blase Ur, Sean M. Segreti, Lujjo Bauer, Nicolas Christin, Lorrie Faith Cranor, Saranga Komanduri, Darya Kurilova, Michelle L. Mazurek, William Melicher, Richard Shay. Measuring Real-World Accuracies and Biases in Modeling Password Guessability. In Proc. USENIX Security Symposium, 2015.

password individually, enabling data-driven strength estimates during password creation [38, 107]. Second, it estimates real-world security against existing, rather than idealized, adversarial techniques. A disadvantage of this approach is that the (simulated) cracking algorithm may not be configured or trained as effectively as by a real attacker, leading to inaccurate estimates of password strength.

We report on the first study of how various cracking approaches used by researchers compare to real-world cracking by professionals, as well as how the choice of approach biases research conclusions. We contracted a computer security firm specializing in password recovery to crack a set of passwords chosen for their diversity in password-composition policies. We then computed the guessability of these passwords using four popular approaches. We tested many configurations of two well-known password-cracking toolkits: John the Ripper [139] and oclHashcat [166]. We also tested two approaches popular in academia: Weir et al.’s probabilistic context-free grammar (PCFG) [197] and Ma et al.’s Markov models [123].

Unsurprisingly, a professional attacker updating his strategy dynamically during cracking outperformed fully automated, “fire-and-forget” approaches (henceforth simply referred to as *automated*), yet often only once billions or trillions of guesses had been made. We found that relying on a single automated approach to calculate guessability underestimates a password’s vulnerability to an experienced attacker, but using the earliest each password is guessed by any automated approach provides a realistic and conservative approximation.

We found that each approach was highly sensitive to its configuration. Using more sophisticated configurations than those traditionally used in academic research, our comparative analysis produced far more nuanced results than prior work. These prior studies found that Markov models substantially outperform the PCFG approach [60, 123], which in turn substantially outperforms tools like John the Ripper [54, 196, 205]. We found that while Markov was marginally more successful at first, it was eventually surpassed by PCFG for passwords created under typical requirements. Furthermore, the most effective configurations of John the Ripper and Hashcat were frequently comparable to, and sometimes even more effective than, the probabilistic approaches we tested.

Both the differences across algorithms and the sensitivity to configuration choices are particularly notable because most researchers use only a single approach as a security metric [38, 43, 61, 126, 157, 182, 196]. In addition, many researchers use adversarial cracking tools in their default configuration [42, 49, 51, 67, 71, 86, 108, 204]. Such a decision is understandable since each algorithm is very resource- and time-intensive to configure and run. This raises the question of whether considering only a single approach biases research studies and security analyses. For instance, would substituting a different cracking algorithm change the conclusions of a study?

We investigate these concerns and find that for comparative analyses of large password sets (e.g., the effect of password-composition policies on guessability), choosing one cracking algorithm can reasonably be expected to yield similar results as choosing another.

However, more fine-grained analyses—e.g., examining what characteristics make a password easy to guess—prove very sensitive to the algorithm used. We find that per-password guessability results often vary by orders of magnitude, even when two approaches are similarly effective against large password sets as a whole. This has particular significance for efforts to help system administrators ban weak passwords or provide customized guidance during password creation [38, 107]. To facilitate the analysis of password guessability across many password-cracking approaches and to further systematize passwords research, we introduce a Password Guessability Service [37] for researchers.

In summary, we make the following main contributions: We show that while running a single cracking

algorithm or tool relatively out-of-the-box produces only a poor estimate of password guessability, using multiple well-configured algorithms or tools in parallel can approximate passwords' vulnerability to an expert, real-world attacker. Furthermore, while comparative analyses of large password sets may be able to rely on a single cracking approach, any analysis of the strength of individual passwords (e.g., a tool to reject weak ones) or the security impact of particular characteristics (e.g., the use of digits, multiple character classes, or character substitutions) must consider many approaches in parallel.

4.2 Methodology

We analyze four automated guessing algorithms and one manual cracking approach (together, our five *cracking approaches*). We first describe the password sets for which we calculated guessability, then explain the training data we used. Afterwards, we discuss our five cracking approaches. Finally, we discuss computational limitations of our analyses.

4.2.1 Datasets

We examine 13,345 passwords from four sets created under composition policies ranging from the typical to the currently less common to understand the success of password-guessing approaches against passwords of different characteristics. Since no major password leaks contain passwords created under strict composition policies, we leverage passwords that our group collected for prior studies of password-composition policies [104, 126, 157]. This choice of data also enables us to contract with a professional computer security firm to crack these unfamiliar passwords. Had we used any major password leak, their analysts would have already been familiar with most or all of the passwords contained in the leak, biasing results.

The passwords in these sets were collected using Amazon's Mechanical Turk crowdsourcing service. Two recent studies have demonstrated that passwords collected for research studies, while not perfect proxies for real data, are in many ways very representative of real passwords from high-value accounts [67, 126].

Despite these claims, we were also curious how real passwords would differ in our analyses from those collected on Mechanical Turk. Therefore, we repeated our analyses of Basic passwords (see below) with 15,000 plaintext passwords sampled from the RockYou gaming site leak [187] and another 15,000 sampled from a Yahoo! Voices leak [77]. As we detail in our supplementary results (Section 4.4.4), our Basic passwords and comparable passwords from these two real leaks yielded approximately the same results.

Next, we detail our datasets, summarized in Table 4.1. The **Basic** set comprises 3,062 passwords collected for a research study requiring a minimum length of 8 characters [104]. As we discuss in Section 4.3, the vast majority of 8-character passwords can be guessed using off-the-shelf, automated approaches. We give particular attention to longer and more complex passwords, which will likely represent future best practices.

System administrators commonly require passwords to contain multiple character classes (lowercase letters, uppercase letters, digits, and symbols). The **Complex** set comprises passwords required to contain 8+ characters, include all 4 character classes, and not be in a cracking wordlist [137] after removing digits and symbols. They were also collected for research [126].

Recent increases in hashing speeds have made passwords of length 8 or less increasingly susceptible to offline guessing [80, 104]. We therefore examine 2,054 **LongBasic** passwords collected for research [104] that required a minimum length of 16 characters. Finally, we examine 990 **LongComplex** passwords, also

Table 4.1: Characteristics of passwords per set, including the percentage of characters that were lowercase (LC) or uppercase (UC) letters, digits, or symbols (Sym).

Set	#	Length	% of Characters			
		Mean (σ)	LC	UC	Digit	Sym
Basic	3,062	9.6 (2.2)	68	4	26	1
Complex	3,000	10.7 (3.2)	51	14	25	11
LongBasic	2,054	18.1 (3.1)	73	4	20	2
LongComplex	990	13.8 (2.6)	57	12	22	8

collected for research [157], that needed to contain 12+ characters, including characters from 3 or more character classes.

4.2.2 Training Data

To compare cracking approaches as directly as possible, we used the same training data for each. That said, each algorithm uses training data differently, making perfectly equivalent comparisons impossible.

Our training data comprised leaked passwords and dictionaries. The passwords were from breaches of MySpace, RockYou, and Yahoo! (excluding the aforementioned 30,000 passwords analyzed in Section 4.4.4). Using leaked passwords raises ethical concerns. We believe our use of such sets in this research is justifiable because the password sets are already available publicly and we exclude personally identifiable information, such as usernames. Furthermore, malicious agents use these sets in attacks [79]; failure to consider them in our analyses may give attackers an advantage over those who work in defensive security.

Prior research has found including natural-language dictionaries to work better than using just passwords [104, 196]. We used the dictionaries previously found most effective: all single words in the Google Web corpus [83], the UNIX dictionary [13], and a 250,000-word inflection dictionary [153]. The combined set of passwords and dictionaries contained 19.4 million unique entries. For cracking approaches that take only a wordlist, without frequency information, we ordered the wordlist by descending frequency and removed duplicates. We included frequency information for the other approaches.

4.2.3 Simulating Password Cracking

To investigate the degree to which research results can be biased by the choice of cracking algorithm, as well as how automated approaches compare to real attacks, we investigated two cracking tools and two probabilistic algorithms. We selected approaches based on their popularity in the academic literature or the password-cracking community, as well as their conceptual distinctness. We also contracted a computer security firm specializing in password cracking for the real-world attack.

Most cracking approaches do not natively provide guess numbers, and instrumenting them to calculate guessability was typically far from trivial. Because this instrumentation enabled our comparisons and can similarly support future research, we include many details in this section about this instrumentation. Furthermore, in Section 4.5, we introduce a Password Guessability Service so that other researchers can leverage our instrumentation and computational resources.

For each approach, we analyze as many guesses as computationally feasible, making 100 trillion (10^{14})

guesses for some approaches and ten billion (10^{10}) guesses for the most resource-intensive approach. With the exception of Hashcat, as explained below, we filter out guesses that do not comply with a password set’s composition policy. For example, a LongComplex password’s guess number excludes guesses with under 12 characters or fewer than 3 character classes.

We define Min_{auto} as the minimum guess number (and therefore the most conservative security result) for a given password across our automated cracking approaches. This number approximates the best researchers can expect with well-configured automation.

In the following subsections, we detail the configuration (and terminology) of the five approaches we tested. We ran CPU-based approaches (JTR, PCFG, Markov) on a 64-core server. Each processor on this server was an AMD Opteron 6274 running at 1.4Ghz. The machine had 256 GB of RAM and 15 TB of disk. Its market value is over \$10,000, yet we still faced steep resource limitations generating Markov guesses. We ran Hashcat (more precisely, oclHashcat) on a machine with six AMD R9 270 GPUs, 2 GB of RAM, and a dual-core processor.

Probabilistic context-free grammar Weir et al.’s probabilistic context-free grammar (termed **PCFG**) [197] has been widely discussed in recent years. We use Komanduri’s implementation of PCFG [106], which improves upon the guessing efficiency of Weir et al.’s work [197] by assigning letter strings probabilities based on their frequency in the training data and assigning unseen strings a non-zero probability. This implementation is a newer version of Kelley et al.’s implementation of PCFG as a lookup table for quickly computing guess numbers, rather than enumerating guesses [104].

Based on our initial testing, discussed further in Section 4.3.1, we prepend our training data, ordered by frequency, before PCFG’s first guess to improve performance. As a result, we do not use Komanduri’s hybrid structures [106], which serve a similar purpose. We weight passwords $10\times$ as heavily as dictionary entries. We were able to simulate 10^{12} guesses for Complex passwords and 10^{14} guesses for the other three sets.

Markov model Second, we evaluated the **Markov**-model password guesser presented by Ma et al. [123], which implemented a number of variants differing by order and approaches to smoothing. We use the order-5 Markov-chain model, which they found most effective for English-language test sets. We tried using both our combined training data (dictionaries and passwords) using the same weighting as with PCFG, as well as only the passwords from our training data. The combined training data and passwords-only training data performed nearly identically. We report only on the combined training data, which was slightly more effective for Basic passwords and is most consistent with the other approaches.

We used Ma et al.’s code [123], which they shared with us, to enumerate a list of guesses in descending probability. We used a separate program to remove guesses that did not conform to the given password-composition policy. Because this approach is extremely resource-intensive, both conceptually (traversing a very large tree) and in its current implementation, we were not able to analyze as many guesses as for other approaches. As with PCFG, we found prepending the training data improved performance, albeit only marginally for Markov. Therefore, we used this tweak. We simulated over 10^{10} guesses for Basic passwords, similar to Ma et al. [123].

John the Ripper We also tested variants of a mangled wordlist attack implemented in two popular software tools. The first tool, John the Ripper (termed **JTR**), has been used in a number of prior studies as a security metric, as described in Section 2.3. In most cases, these prior studies used JTR with its stock mangling

rules. However, pairing the stock rules with our 19.4-million-word wordlist produced only 10^8 guesses for Basic passwords. To generate more guesses, we augment the stock rules with 5,146 rules released for DEF CON’s “Crack Me If You Can” (CMIYC) password-cracking contest in 2010 [110]. Specifically, we use Trustwave SpiderLabs’ reordering of these rules for guessing efficiency [179]. Our JTR tests therefore use the stock mangling rules followed by the Spiderlabs rules. For completeness, Section 4.4.2 presents these rules separately.

Instrumenting JTR to calculate precise guess numbers was complex. We used `john-1.7.9-jumbo` with the `--stdout` flag to output guesses to standard out. We piped these guesses into a program we wrote to perform a regular expression check filtering out guesses that do not conform to the given password policy. This program then does a fast hash table lookup with GNU `gperf` [85] to quickly evaluate whether a guess matches a password in our dataset. Using this method, we achieved a throughput speed of 3 million guesses per second and made more than 10^{13} guesses for Basic passwords.

Hashcat While Hashcat is conceptually similar to JTR, we chose to also include it in our tests for two reasons. First, we discovered in our testing that JTR and Hashcat iterate through guesses in a very different order, leading to significant differences in the efficacy of guessing specific passwords. JTR iterates through the entire wordlist using one mangling rule before proceeding to the subsequent mangling rule. Hashcat, in contrast, iterates over all mangling rules for the first wordlist entry before continuing to the subsequent wordlist entry.

Second, the GPU-based `oclHashcat`, which is often used in practice [79, 80, 109, 140], does not permit users to filter guesses that do not meet password-composition requirements except for computationally expensive hash functions. We accept this limitation both because it represents the actual behavior of a popular closed-source tool and because, for fast hashes like MD5 or NTLM, guessing without filtering cracks passwords faster in practice than applying filtering.

Unlike JTR, Hashcat does not have a default set of mangling rules, so we evaluated several. We generally report on only the most effective set, but detail our tests of four different rule sets in Section 4.4.3. This most effective rule set, which we term **Hashcat** throughout, resulted from our collaboration with a Hashcat user and password researcher from MWR InfoSecurity [81, 134], who shared his mangling rules for the purpose of this analysis. We believe such a configuration represents a typical expert configuration of Hashcat.

We used `oclHashcat-1.21`. While, like JTR, Hashcat provides a debugging feature that streams guesses to standard output, we found it extremely slow in practice relative to Hashcat’s very efficient GPU implementation. In support of this study, Hashcat’s developers generously added a feature to `oclHashcat` to count how many guesses it took to arrive at each password it cracked. This feature is activated using the flag `--outfile-format=11` in `oclHashcat-1.20` and above. We therefore hashed the passwords in our datasets using the NTLM hash function, which was the fastest for Hashcat to guess in our benchmarks. We then used Hashcat to actually crack these passwords while counting guesses, with throughput of roughly 10 billion guesses per second on our system. We made more than 10^{13} guesses for Basic passwords, along with nearly 10^{15} guesses for some alternate configurations reported in Section 4.4.3.

Professional cracker An open question in measuring password guessability using off-the-shelf, automated tools is how these attacks compare to an experienced, real-world attacker. Such attackers manually customize and dynamically update their attacks based on a target set’s characteristics and initial successful cracks.

To this end, we contracted an industry leader in professional password recovery services, KoreLogic (termed **Pros**), to attack the password sets we study. We believe KoreLogic is representative of expert password crackers because they have organized the DEF CON “Crack Me If You Can” password-cracking contest since 2010 [109] and perform password-recovery services for many Fortune-500 companies [112]. For this study, they instrumented their distributed cracking infrastructure to count guesses.

Like most experienced crackers, the KoreLogic analysts used tools including JTR and Hashcat with proprietary wordlists, mangling rules, mask lists, and Markov models optimized over 10 years of password auditing. Similarly, they dynamically update their mangling rules (termed *freestyle rules*) as additional passwords are cracked. To unpack which aspects of a professional attack (e.g., proprietary wordlists and mangling rules, freestyle rules, etc.) give experienced crackers an advantage, we first had KoreLogic attack a set of 4,239 Complex passwords (distinct from those reported in our other tests) in artificially limited configurations.

We then had the professionals attack the Complex, LongBasic, and LongComplex passwords with no artificial limitations. An experienced password analyst wrote freestyle rules for each set before cracking began, and again after 10^{13} guesses based on the passwords guessed to that point. They made more than 10^{14} guesses per set.

LongBasic and LongComplex approaches are rare in corporate environments and thus relatively unfamiliar to real-world attackers. To mitigate this unfamiliarity, we randomly split each set in two and designated half for training and half for testing. We provided analysts with the training half (in plaintext) to familiarize them with common patterns in these sets. Because we found that automated approaches can already crack most Basic passwords, rendering them insecure, we chose not to have the professionals attack Basic passwords.

4.2.4 Computational Limitations

As expected, the computational cost of generating guesses in each approach proved a crucial limiting factor in our tests. In three days, oclHashcat, the fastest of our approaches, produced 10^{15} guesses using a single AMD R9 290X GPU (roughly a \$500 value). In contrast, the Markov approach (our slowest) required three days on a roughly \$10,000 server (64 AMD Opteron 6274 CPU cores and 256 GB of RAM) to generate 10^{10} guesses without computing a single hash. In three days on the same machine as Markov, PCFG simulated 10^{13} guesses.

The inefficiency of the Markov approach stems partially from our use of a research implementation. Even the most efficient implementation, however, would still face substantial conceptual barriers. Whereas Hashcat and JTR incur the same performance cost generating the quadrillionth guess as the first guess, Markov must maintain a tree of substring probabilities. As more guesses are desired, the tree must grow, increasing the cost of both storing and traversing it. While the Markov approach produced a high rate of successful guesses per guess made (see Section 4.3.2), the cost of generating guesses makes it a poor choice for computing guessability beyond billions of guesses.

Further, our automated approaches differ significantly in how well they handle complex password-composition policies. For PCFG, non-terminal structures can be pruned before guessing starts, so only compliant passwords are ever generated. As a result, it takes about equal time for PCFG to generate Basic passwords as LongComplex passwords. In contrast, Markov must first generate all passwords in a probability range and then filter out non-compliant passwords, adding additional overhead per guess. JTR has a similar generate-then-filter mechanism, while Hashcat (as discussed above) does not allow this post-hoc filtering

at all for fast hashes. This means that Markov and JTR take much longer to generate valid LongComplex guesses than Basic guesses, and Hashcat wastes guesses against the LongComplex set.

As a result of these factors, the largest guess is necessarily unequal among approaches we test, and even among test sets within each approach. To account for this, we only compare approaches directly at equivalent guess numbers. In addition, we argue that these computational limitations are important in practice, so our findings can help researchers understand these approaches and choose among them appropriately.

4.3 Results

We first show, in Section 4.3.1, that for each automated guessing approach we evaluated, different seemingly reasonable configurations produce very different cracking results, and that out-of-the-box configurations commonly used by researchers substantially underestimate password vulnerability.

Next, in Section 4.3.2, we examine the relative performance of the four automated approaches. We find they are similarly effective against Basic passwords. They have far less success against the other password sets, and their relative effectiveness also diverges.

For the three non-Basic sets, we also compare the automated approaches to the professional attack. Pros outperform the automated approaches, but only after a large number of guesses. As Pros crack more passwords, their manual adjustments prove quite effective; automated approaches lack this feedback mechanism. We also find that, at least through 10^{14} guesses, automated approaches can conservatively approximate human password-cracking experts, but only if a password is counted as guessed when *any* of the four automated approaches guesses it. A single approach is not enough.

In Section 4.3.3, we explore the degree to which different cracking approaches overlap in which particular passwords they guess. While multiple approaches successfully guess most Basic passwords, many passwords in the other classes are guessed only by a single approach. We also find that different cracking approaches provide systematically different results based on characteristics like the number of character classes contained in a particular password.

In Section 4.3.4, we revisit how the choice of guessing approach impacts research questions at a high level (e.g., how composition policies impact security) and lower level (e.g., if a particular password is hard to guess). While we find analyses on large, heterogeneous sets of passwords to be fairly robust, security estimates for a given password are very sensitive to the approach used.

4.3.1 The Importance of Configuration

We found that using any guessing approach naively performed far more poorly, sometimes by more than an order of magnitude, than more expert configurations.

Stock versus advanced configurations We experimented with several configurations each for Hashcat and JTR, including the default configurations they ship with, and observed stark differences in performance. We detail a few here; others are described in Section 4.4.2 and Section 4.4.3.

For example, Hashcat configured with the (default) Best64 mangling rules guessed only about 2% of the Complex passwords before running out of guesses. Using the mangling rules described in Section 5.3, it made far more guesses, eventually cracking 30% (Figure 4.1).

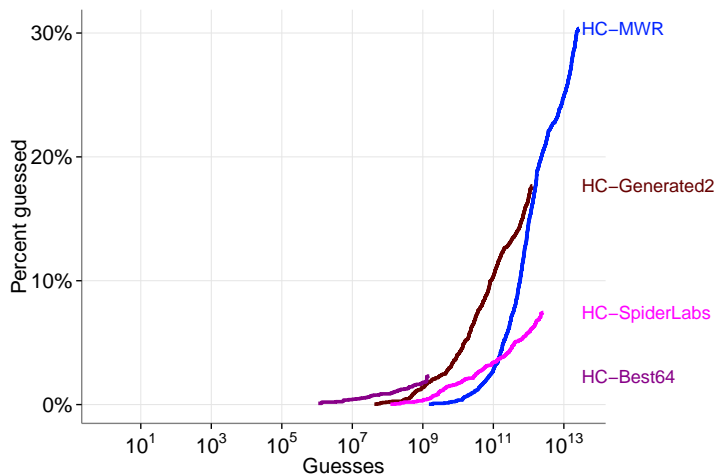


Figure 4.1: Results of Hashcat configured using the same wordlist, but different sets of mangling rules (described in Section 4.4.3), to guess Complex passwords.

Similarly, JTR guessed less than 3% of Complex passwords before exhausting its stock rules. The larger set of rules described in Section 5.3 enabled it to guess 29% (see Section 4.4.2 for details). We found similar configuration effects for LongComplex passwords, and analogous but milder effects for the Basic and LongBasic sets.

We also compared the PCFG implementation we use throughout Section 4.3 [106] with our approximation of the originally published algorithm [197], which differs in how probabilities are assigned (see Section 5.3). As we detail in Section 4.4.1, the newer PCFG consistently outperforms the original algorithm; the details of the same conceptual approach greatly impact guessability analyses.

Choices of training data The performance of PCFG and Markov depends heavily on the quality of training data. Our group previously found that training with closely related passwords improves performance [104]. For our non-basic password sets, however, closely matched data is not available in publicly leaked sets.

In tests reported in Section 4.4.1, we thus incorporated closely matched data via cross-validation, in which we iteratively split the test set into training and testing portions. Using cross-validation improved guessing efficiency for three of the four password sets, most dramatically for LongBasic. This result demonstrates that an algorithm trained with generic training data will miss passwords that are vulnerable to an attacker who has training data that closely matches a target set. To minimize differences across approaches, however, PCFG results in Section 4.3 use generic training data only.

Actionable takeaways Together, these results suggest that a researcher must carefully manage guessing configuration before calculating password guessability. In particular, tools like JTR and Hashcat will “out of the box” systematically underestimate password guessability. Unfortunately, many existing research studies rely on unoptimized configurations [42, 49, 51, 67, 71, 86, 108, 204].

While we report on the configurations we found most effective in extensive testing, we argue that the research community should establish configuration best practices, which may depend on the password sets being targeted.

4.3.2 Comparison of Guessing Approaches

We first show that automated approaches differ in effectiveness based on the nature of the password sets being cracked and the number of guesses at which they are compared. We then compare these automated approaches to cracking by an expert attacker making dynamic updates, finding that the expert lags in initial guessing efficiency, yet becomes stronger over time. We find the minimum guess number across automated approaches can serve as a conservative proxy for guessability by an expert attacker.

Guessing by Automated Approaches

On some password sets and for specific numbers of guesses, the performance of all four approaches was similar (e.g., at 10^{12} guesses all but Markov had guessed 60-70% of Basic passwords). In contrast, on other sets, their performance was inconsistent at many points that would be relevant for real-world cracking (e.g., PCFG cracked 20% of Complex passwords by 10^{10} guesses, while Hashcat and JTR had cracked under 3%).

As shown in Figure 4.2, all four automated approaches were quite successful at guessing Basic passwords, the most widely used of the four classes. Whereas past work has found that, for password sets resembling our Basic passwords, PCFG often guesses more passwords than JTR [54] or that Markov performs significantly better than PCFG [123], good configurations of JTR, Markov, and PCFG performed somewhat similarly in our tests. Hashcat was less efficient at generating successful guesses in the millions and billions of guesses, yet it surpassed JTR by 10^{12} guesses and continued to generate successful guesses beyond 10^{13} guesses.

The four automated approaches had far less success guessing the other password sets. Figure 4.2b shows the guessability of the Complex passwords under each approach. Within the first ten million guesses, very few passwords were cracked by any approach. From that point until its guess cutoff, PCFG performed best, at points having guessed nearly ten times as many passwords as JTR. Although its initial guesses were often successful, the conceptual and implementation-specific performance issues we detailed in Section 4.2.4 prevented Markov from making over 100 million valid Complex guesses, orders of magnitude less than the other approaches we examined. A real attack using this algorithm would be similarly constrained.

Both Hashcat and JTR performed poorly compared to PCFG in early Complex guessing. By 10^9 guesses, each had each guessed under 3% of Complex passwords, compared to 20% for PCFG. Both Hashcat and JTR improve rapidly after 10^{10} guesses, however, eventually guessing around 30% of Complex passwords.

JTR required almost 10^{12} guesses and Hashcat required over 10^{13} guesses to crack 30% of Complex passwords. As we discuss in Section 4.3.3, there was less overlap in which passwords were guessed by multiple automated approaches for Complex passwords than for Basic passwords. As a result, the Min_{auto} curve in Figure 4.2b, representing the smallest guess number per password across the automated approaches, shows that just under 10^{11} guesses are necessary for 30% of Complex passwords to have been guessed by at least one automated approach. Over 40% of Complex passwords were guessed by at least one automated approach in 10^{13} guesses.

LongBasic passwords were also challenging for all approaches to guess, though relative differences across approaches are not as stark as for Complex passwords. Markov was marginally more successful than other approaches at its cutoff just before 10^9 guesses. JTR and PCFG both continued to generate successful guesses through when JTR exhausted its guesses after guessing 10% of the passwords. Hashcat lagged slightly behind JTR at 10^9 guesses (7% cracked vs $\sim 9\%$), but was able to make more guesses than either, eventually guessing over 20% of the passwords, compared to 16% for PCFG and 10% for JTR at those approaches' guess cutoffs.

As with LongBasic passwords, all approaches had difficulty guessing LongComplex passwords. As

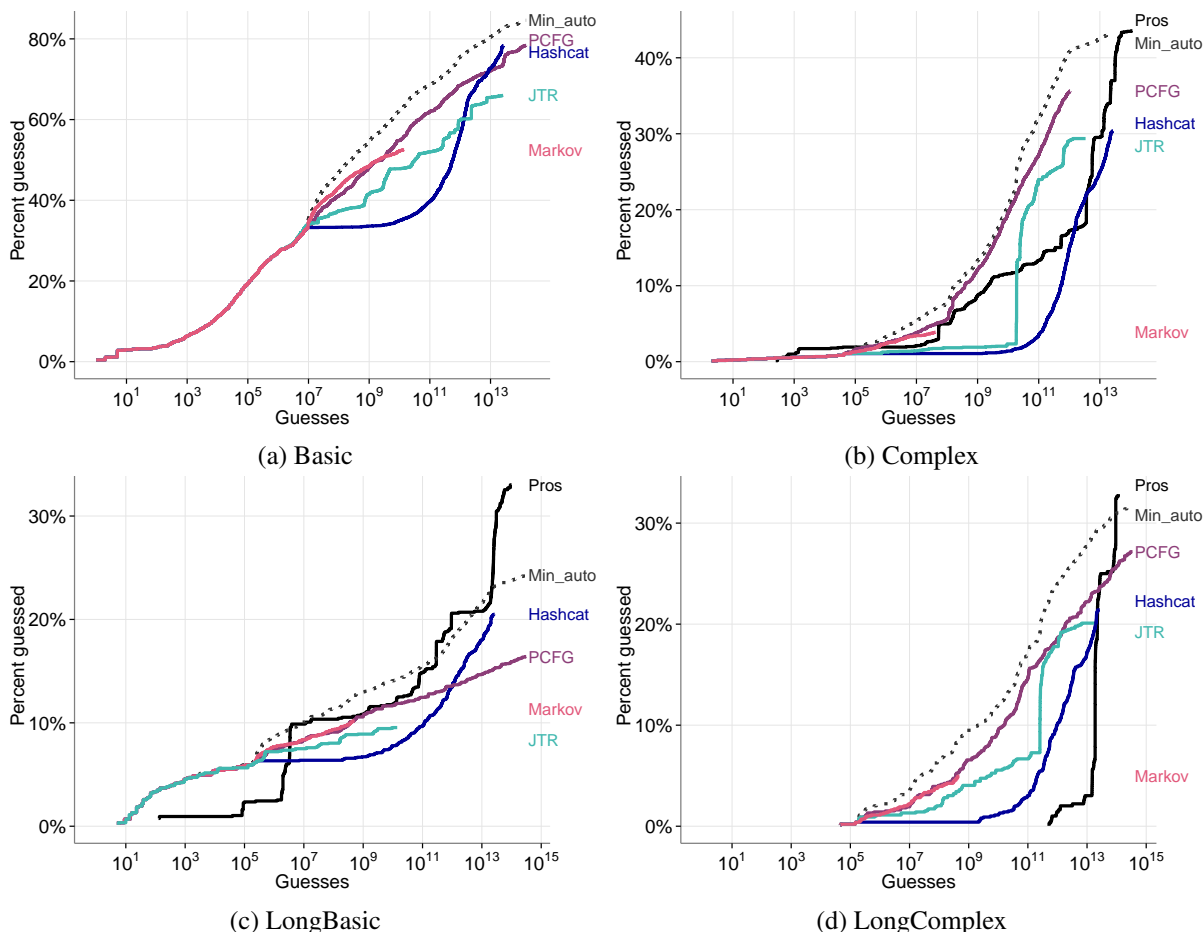


Figure 4.2: Automated approaches’ success guessing the different password sets. Min_{auto} represents the smallest guess number for a password by any automated approach. Pros are experts updating their guessing strategy dynamically.

shown in Figure 4.2d, nearly 70% of LongComplex passwords were not guessed by any of the approaches we examined even after trillions of guesses. The relative performance of the four automated guessing approaches for LongComplex passwords again differed noticeably. Markov and PCFG again outperformed other approaches early. Markov guessed 5% of the passwords after 10^8 guesses, yet reached its guess cutoff soon thereafter. At 10^9 guesses PCFG and JTR had both also guessed at least 5% of the passwords, compared to almost no passwords guessed by Hashcat. PCFG’s and JTR’s performance diverged and then converged at higher guess numbers. Hashcat caught up at around 10^{13} guesses, cracking 20% of LongComplex passwords.

Guessing by Pros

As we expected, Pros guessed more passwords overall than any of the automated approaches. As we discussed in Section 5.3, we chose not to have Pros attack Basic passwords because those passwords could be guessed with automated approaches alone. As shown in Figures 4.2b–4.2d, within 10^{14} guesses Pros cracked 44% of

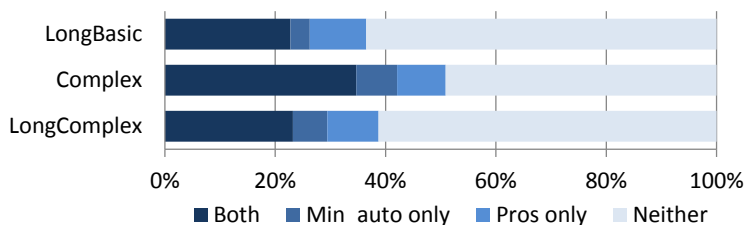


Figure 4.3: The proportion of passwords guessed by Min_{auto} , Pros, both, or neither within 10^{14} guesses.

Complex passwords, 33% of LongBasic passwords, and 33% of LongComplex passwords, improving on the guessing of the best automated approach.

Three aspects of guessing by Pros were particularly notable. First, even though Pros manually examined half of each password set and adjusted their mangling rules and wordlists before making the first guess against each set, automated approaches were often more successful at early guessing. For example, Markov surpassed Pros at guessing Complex passwords in the first 10^2 guesses and again from around 10^6 till Markov’s guess cutoff at 5×10^7 . Similarly, all four automated approaches guessed LongComplex passwords more successfully than Pros from the start of guessing until past 10^{13} guesses. All approaches guessed LongBasic passwords better than Pros for the first 10^6 guesses.

Second, while Pros lagged in early guessing, the freestyle rules an experienced analyst wrote at 10^{13} guesses proved rather effective and caused a large spike in successful guesses for all three password sets. Hashcat, the only automated approach that surpassed 10^{13} guesses for all sets, remained effective past 10^{13} guesses, yet did not experience nearly the same spike.

Third, while Pros were more successful across password sets once a sufficiently high number of guesses had been reached, the automated approaches we tested had guessing success that was, to a very rough approximation, surprisingly similar to Pros. As we discussed in Section 4.3.1 and discuss further in Section 4.4, this success required substantial configuration beyond each approach’s performance out of the box.

We found that our Min_{auto} metric (the minimum guess number for each password across Hashcat, JTR, Markov, and PCFG) served as a conservative approximation of the success of Pros, at least through our automated guess cutoffs around 10^{13} guesses. As seen in Figures 4.2b–4.3, Pros never substantially exceeded Min_{auto} , yet often performed worse than Min_{auto} .

Professional cracking with limitations To unpack why professional crackers have an advantage over novice attackers, we also had KoreLogic attack a different set of Complex passwords in artificially limited configurations. These limitations covered the wordlists they used, the mangling rules they used, and whether they were permitted to write freestyle rules. To avoid biasing subsequent tests, we provided them a comparable set of 4,239 Complex passwords [104] distinct from those examined in Section 4.3. We call this alternate set **Complex_{pilot}**.

As shown in Table 4.2, we limited Pros in Trial 1 to use the same wordlist we used elsewhere and did not allow freestyle rules. In Trial 2, we did not limit the wordlist, but did limit mangling rules to those used in the 2010 Crack Me If You Can contest [110]. In Trial 3 and Trial 4, we did not limit the starting wordlist or mangling rules. In Trial 4, however, KoreLogic analysts dynamically adjusted their attacks through freestyle rules and wordlist tweaks after 10^{14} guesses.

Table 4.2: The four trials of Pros guessing $\text{Complex}_{\text{pilot}}$. We artificially limited the first three trials to uncover why Pros have an advantage over more novice attackers.

Trial	Wordlist	Rules	Freestyle Rules
1	CMU wordlist	Anything	None
2	Anything	2010 CMIYC rules	None
3	Anything	Anything	None
4	Anything	Anything	Unlimited

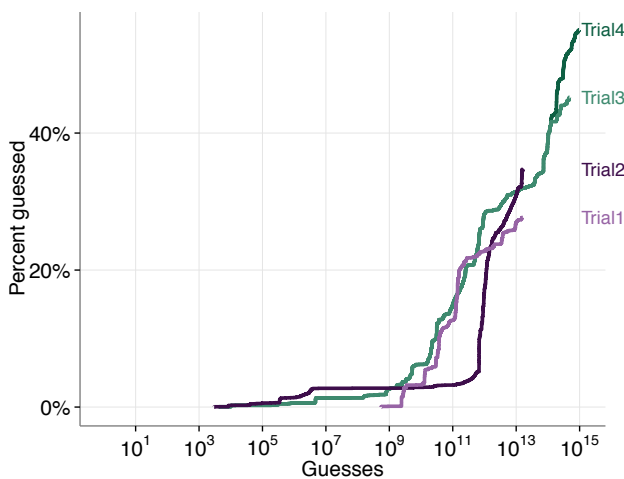


Figure 4.4: $\text{Complex}_{\text{pilot}}$ guessability by trial.

We found that KoreLogic’s set of proprietary mangling rules had a far greater impact on guessing efficiency than their proprietary wordlist (Figure 4.4). Furthermore, as evidenced by the difference between Trial 3 and Trial 4, freestyle rules also had a major impact at the point the analyst wrote them.

Actionable takeaways One conceptual advantage of parameterized metrics is that they model an attack using existing cracking approaches. However, it has long been unclear whether automated cracking approaches used by researchers effectively model the dynamically updated techniques used by expert real-world attackers. Our results demonstrate that only by considering multiple automated approaches in concert can researchers approximate professional password cracking.

One of our primary observations, both from comparing Pros to the automated approaches and from our trials artificially limiting Pros (Section 4.3.2), is that dynamically updated freestyle rules can be highly effective. This result raises the question of to what extent automated approaches can model dynamic updates. Although the adversarial cracking community has discussed techniques for automatically generating mangling rules from previous cracks [125], researchers have yet to leverage such techniques.

Contrary to prior research (e.g., [54, 123]), we found that Hashcat, JTR, Markov, and PCFG all performed relatively effectively when configured and trained according to currently accepted best practices in the cracking and research communities. That said, our tests also highlighted a limitation of the guessability metric in not considering the performance cost of generating a guess. Despite its real-world popularity, Hashcat performed comparatively poorly until making trillions of guesses, yet generated guesses very quickly.

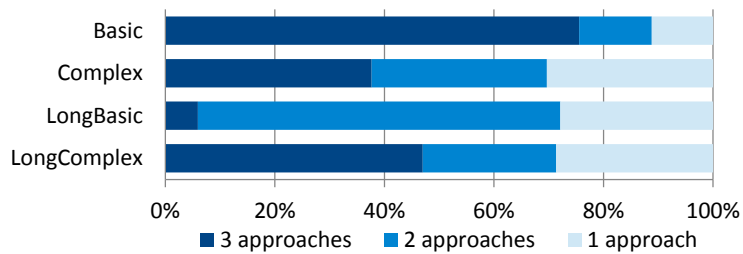


Figure 4.5: Number of automated approaches, excluding Markov, that cracked a particular password. We ignore passwords not guessed by any approach and use the same guess cutoff for all guessing approaches within a set.

If hashing a guess is the dominant time factor, as is the case for intentionally slow hash functions like bcrypt, PBKDF2, and scrypt, probabilistic approaches like Markov and PCFG are advantageous for an attacker. For fast hash functions like MD5 or NTLM, Hashcat’s speed at generating and hashing guesses results in more passwords being guessed in the same wall-clock time. As discussed in Section 4.2.4, Markov proved comparatively very resource-intensive to run to a large guess number, especially for password sets with complex requirements. These practical considerations must play a role in how researchers select the best approaches for their needs.

4.3.3 Differences Across Approaches

Next, we focus on differences between approaches. We first examine if multiple approaches guess the same passwords. We then examine the guessability of passwords with particular characteristics, such as those containing multiple character classes or character substitutions. To examine differences across how approaches model passwords, for analyses in this section we do not prepend the training data to the guesses generated by the approach.

Overlap in Successful Guesses

While one would expect any two cracking approaches to guess slightly different subsets of passwords, we found larger-than-expected differences for three of the four password sets. Figure 4.5 shows the proportion of passwords in each class guessed by all four approaches, or only some subset of them. We exclude passwords guessed by none of the automated approaches. Within a password set, we examine all approaches only up to the minimum guess cutoff among Hashcat, JTR, and PCFG; we exclude Markov due to its low guess cutoffs.

The three approaches guessed many of the same Basic passwords: Three-fourths of Basic passwords guessed by any approach were guessed by all of them. Only 11% of Basic passwords were guessed only by a single approach. In contrast, only 6% of LongBasic passwords were guessed by all approaches, while 28% of Complex, LongBasic, and LongComplex passwords were guessed only by a single approach.

Guessing Success by Password Characteristics

While it is unsurprising that different approaches do better at guessing distinct types of passwords, we found differences that were large and difficult to predict.

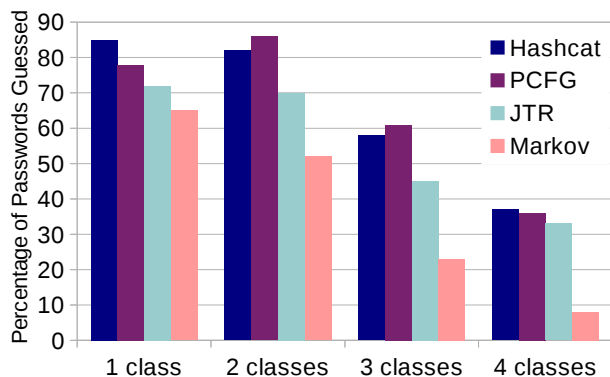


Figure 4.6: Percentage of Basic passwords each approach guessed, by character-class count.

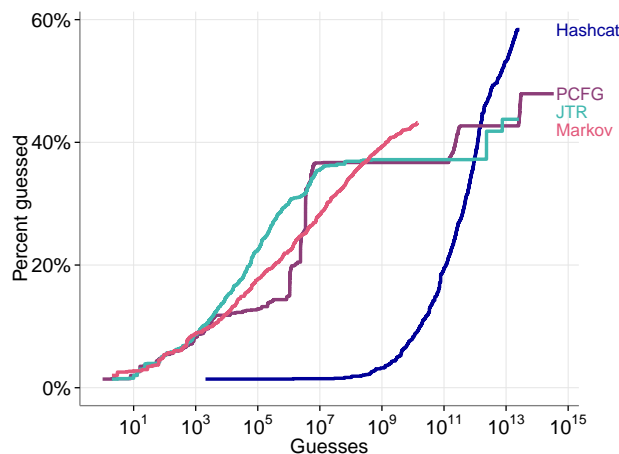


Figure 4.7: Approaches' effectiveness guessing passwords composed entirely of lowercase letters across sets.

Character classes and length We first considered how efficiently automated approaches guessed passwords relative to their length and character-class count. These two characteristics are of particular interest because they are frequently used in password-composition policies.

As shown in Figure 4.6, the impact of adding character classes is not as straightforward as one might expect. While the general trend is for passwords with more character classes to be stronger, the details vary. Markov experiences a large drop in effectiveness with each increase in character classes (63% to 52% to 23% to 8%). JTR, by contrast, finds only a minor difference between one and two classes (72% to 70%). PCFG actually increases in effectiveness between one and two classes (78% to 86%). Since changes in security and usability as a result of different policies are often incremental (e.g., [35]), the magnitude of these disagreements can easily affect research conclusions about the relative strength of passwords.

In contrast, we did not find surprising idiosyncrasies based on the length of the password. For all approaches, cracking efficiency decreased as length increased.

Character-level password characteristics As the research community seeks to understand the characteristics of good passwords, a researcher might investigate how easy it is to guess all-digit passwords, which are

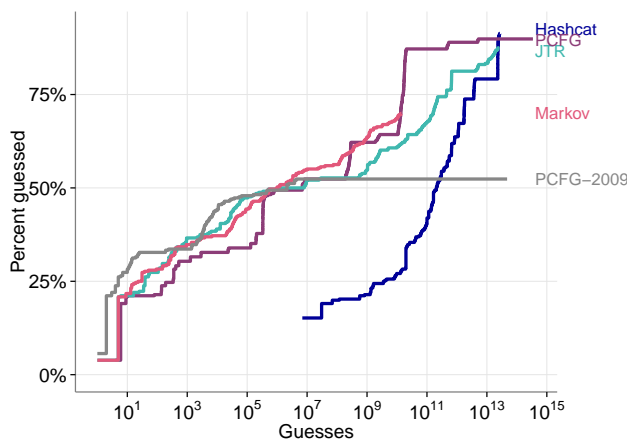


Figure 4.8: Guessing efficiency for the 350 Basic and LongBasic passwords composed entirely of digits.

common [30], or examine the effect of character substitutions (e.g., $\$Hpl0v3cr@ft!\$ \rightarrow \$Hpl0v3cr@ft!\$$) on guessability. Despite their sometimes similar effectiveness overall, approaches often diverged when guessing passwords that had these characteristics. As a result, researchers using different approaches could draw different conclusions about the guessability of these properties.

The guessability of the 1,490 passwords (across sets) composed entirely of lowercase letters varied starkly by guessing approach. This variation is particularly notable because such passwords made up 29% of Basic and LongBasic passwords, and were impermissible under the other two composition policies. As shown in Figure 4.7, Hashcat guessed few such passwords until well into the billions of guesses, whereas Markov successfully guessed passwords composed entirely of lowercase letters throughout its attack. In contrast, PCFG had a large spike in successful guesses between 1 million and 10 million guesses, but then plateaued. JTR had early success, but similarly plateaued from 10 million guesses until into the trillions of guesses.

Figure 4.8 shows the guessability of the 350 passwords comprised only of digits across the Basic and LongBasic sets. Similar to the results for passwords of other common characteristics (Section 4.3.3), approaches differed. Of particular note is PCFG–2009, which is our approximation of the original 2009 Weir et al. algorithm [197] in which alphabetic strings are assigned uniform probability and unseen terminals are a probability of zero. It plateaued at around 50% of such passwords guessed in fewer than 10 million guesses. Idiosyncratically, through 10^{14} guesses, it would never guess another password of this type because of the way it assigns probabilities.

Similarly, approaches differed in their efficiency guessing passwords containing character substitutions, which we identified using crowdsourcing on Amazon’s Mechanical Turk. Passwords identified by crowdworkers as containing character substitutions included *4Everblessed*, *B1cycle_Race*, and *Ca\$hmoneybr0*. PCFG performed poorly relative to JTR and Markov at guessing passwords with character substitutions. A researcher using only PCFG could mistakenly believe these passwords are much stronger than they are. We found similar differences with many other characteristics, potentially skewing research conclusions.

Actionable takeaways Given the many passwords guessed by only a single cracking approach and the systematic differences in when passwords with certain characteristics are guessed, we argue that researchers must consider major cracking approaches in parallel.

Our results also show how comparative analyses uncover relative weaknesses of each approach. Upon close examination, many of these behaviors make sense. For example, PCFG abstracts passwords into structures of non-terminal characters based on character class, ignoring contextual information across these boundaries. As a result, *P@sswOrd* would be split into “P,” “@,” “ssw,” “0,” and “rd,” explaining PCFG’s poor performance guessing passwords with character substitutions.

4.3.4 Robustness of Analyses to Approach

In this section, we examine whether differences among automated cracking approaches are likely to affect conclusions to two main types of research questions. We first consider analyses of password sets, such as passwords created under particular password-composition policies. We find such analyses to be somewhat, but not completely, robust to the approach used.

In contrast, per-password analyses are very sensitive to the guessing approach. Currently, such analyses are mainly used in security audits [171] to detect weak passwords. In the future, however, per-password strength metrics may be used to provide detailed feedback to users during password creation, mirroring the recent trend of data-driven password meters [38, 107]. The ability to calculate a guess number per-password is a major advantage of parameterized metrics over statistical metrics, yet this advantage is lost if guess numbers change dramatically when a different approach is used. Unfortunately, we sometimes found huge differences across approaches.

Per Password Set

As an example of an analysis of large password sets, we consider the relative guessability of passwords created under different composition policies, as has been studied by Shay et al. [157] and Kelley et al. [104].

Figure 4.9 shows the relative guessability of the three password sets examined by the Pros. LongBasic passwords were most vulnerable, and LongComplex passwords least vulnerable, to early guessing (under 10^9 guesses). Between roughly 10^9 and 10^{12} guesses, LongBasic and Complex passwords followed similar curves, though Complex passwords were cracked with higher success past 10^{12} guesses. Very few LongComplex passwords were guessed before 10^{13} guesses, yet Pros quickly guessed about one-third of the LongComplex set between 10^{13} and 10^{14} guesses.

Performing the same analysis using Min_{auto} guess numbers instead (Figure 4.10) would lead to similar conclusions. LongBasic passwords were again more vulnerable than Complex or LongComplex under 10^8 guesses. After 10^{12} guesses, Complex passwords were easier to guess than LongBasic or LongComplex passwords. Basic passwords were easy to guess at all points. The main difference between Min_{auto} and Pros was that LongComplex passwords appear more vulnerable to the first 10^{12} guesses under Min_{auto} than Pros.

Based on this data, a researcher comparing composition policies would likely reach similar conclusions using either professionals or a combination of automated approaches. As shown in Figure 4.11, we repeated this analysis using each of the four automated approaches in isolation. Against every approach, Basic passwords are easily guessable, and LongBasic passwords are comparatively vulnerable during early guessing. After trillions of guesses, Hashcat, PCFG, and JTR find Long Complex passwords more secure than Complex passwords. In each case, a researcher would come to similar conclusions about the relative strength of these password sets.

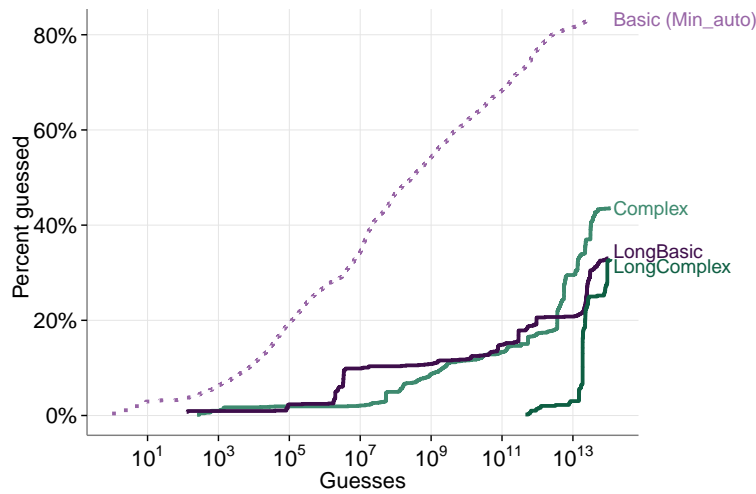


Figure 4.9: Pros' comparative success guessing each password set. For reference, the dotted line represents the Min_{auto} guess across automated approaches for Basic passwords, which the Pros did not try to guess.

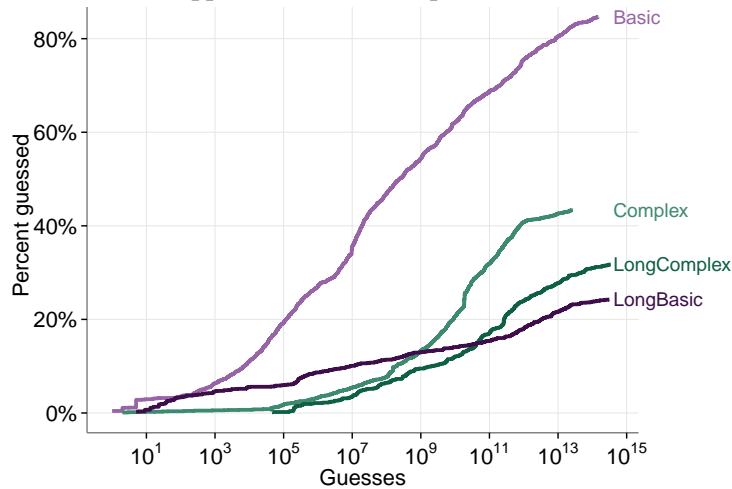


Figure 4.10: The guessability of all four password sets under Min_{auto} , representing the smallest guess number for each password across all four automated approaches.

Per Individual Password

Analyses of the strength of individual passwords, in contrast, proved very sensitive to the guessing approach. Although one would expect different approaches to guess passwords at somewhat different times, many passwords' guess numbers varied by orders of magnitude across approaches. This state of affairs could cause a very weak password to be misclassified as very strong.

We examined per-password differences pairwise among JTR, Markov, and PCFG, using the same guess cutoff for each approach in a pair. Because Hashcat's early guesses were often unsuccessful, we exclude it from this analysis. Passwords not guessed by the guess cutoff were assigned a guess number one past the cutoff, lower-bounding differences between passwords guessed by one approach but not the other. Per password, we calculated \log_{10} of the ratio between the two guess numbers. For example, *iceman1232* was

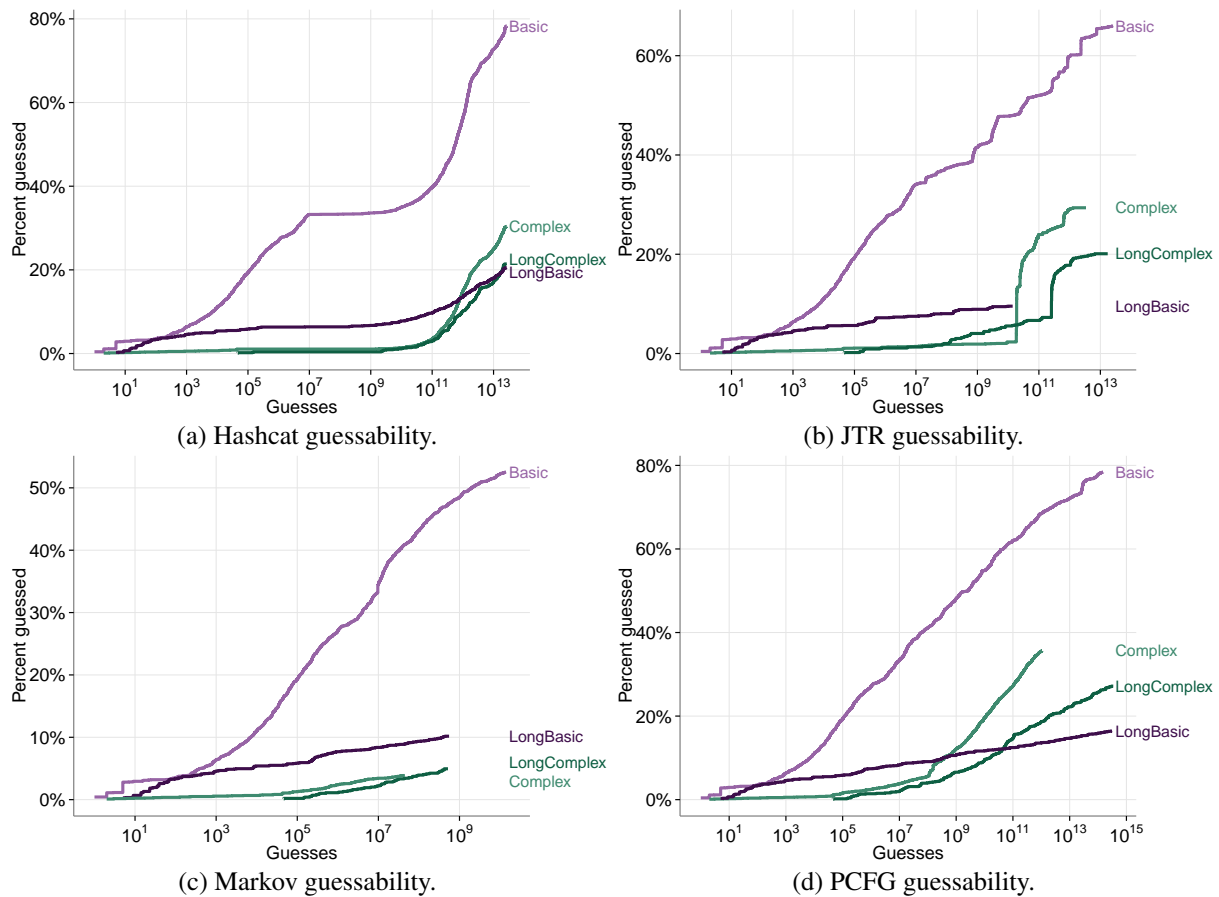


Figure 4.11: The relative guessability of the four different password sets under each of the four automated cracking approaches considered in isolation. The research conclusions would be fairly similar in each case.

guess 595,300,840 for JTR and 61,554,045 for Markov, a 0.985 order of magnitude difference.

Among passwords guessed by JTR, PCFG, or both, 51% of passwords had guess numbers differing by more than an order of magnitude between approaches, indicating large variations in the resulting security conclusions. Alarmingly, some passwords had guess numbers differing by over 12 orders of magnitude (Figure 4.12). For example, *P@ssw0rd!* took JTR only 801 Complex guesses, yet PCFG never guessed it in our tests. Similarly, *1q2w3e4r5t6y7u8i* was the 29th LongBasic JTR guess, yet it was not among the 10^{14} such guesses PCFG made. In contrast, PCFG guessed *Abc@1993* after 48,670 guesses and *12345678password* after 130,555 guesses. JTR never guessed either password.

We found similar results in the two other pairwise comparisons. Among passwords guessed by Markov, PCFG, or both, 41% of guess numbers differed by at least one order of magnitude. In an extreme example, the passwords *1qaz!QAZ* and *1q2w3e4r5t6y7u8i* were among the first few hundred Markov guesses, yet not guessed by PCFG's guess cutoff. Conversely, *unitedstatesofamerica* was among PCFG's first few dozen LongBasic guesses, yet never guessed by Markov. For 37% of passwords, JTR and Markov guess numbers differed by at least one order of magnitude. Markov was particularly strong at guessing long passwords with

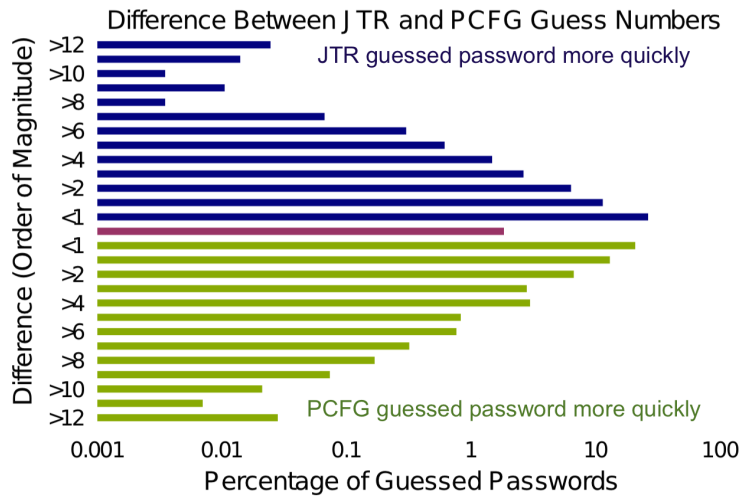


Figure 4.12: The % (log scale) of passwords guessed by JTR or PCFG whose guess numbers differed by a given order of magnitude. e.g., the blue > 6 bar represents passwords guessed by JTR more than 6, but no more than 7, orders of magnitude more quickly than by PCFG.

predictable patterns. For instance, *password123456789*, *1234567890123456*, and *qwertyuiopasdfgh* were among Markov’s first thirty guesses, yet JTR did not guess any of them by its cutoff.

Actionable takeaways As researchers and system administrators ask questions about password strength, they must consider whether their choice of cracking approach biases the results. When evaluating the strength of a large, heterogeneous password set, any of Hashcat, JTR, Markov, or PCFG—if configured effectively—provide fairly similar answers to research questions. Nonetheless, we recommend the more conservative strategy of calculating guessability using Min_{auto} .

In contrast, guessability results per-password can differ by many orders of magnitude between approaches even using the same training data. To mitigate these differences, we again recommend Min_{auto} for the increasingly important task of providing precise feedback on password strength to users.

4.4 Supplementary Experimental Results

We provide additional measurements of how guessing approaches perform in different configurations. To support the ecological validity of our study, we also repeat analyses from Section 4.3 on password sets leaked from RockYou and Yahoo. We provide these details in hopes of encouraging greater accuracy and reproducibility across measurements of password guessability.

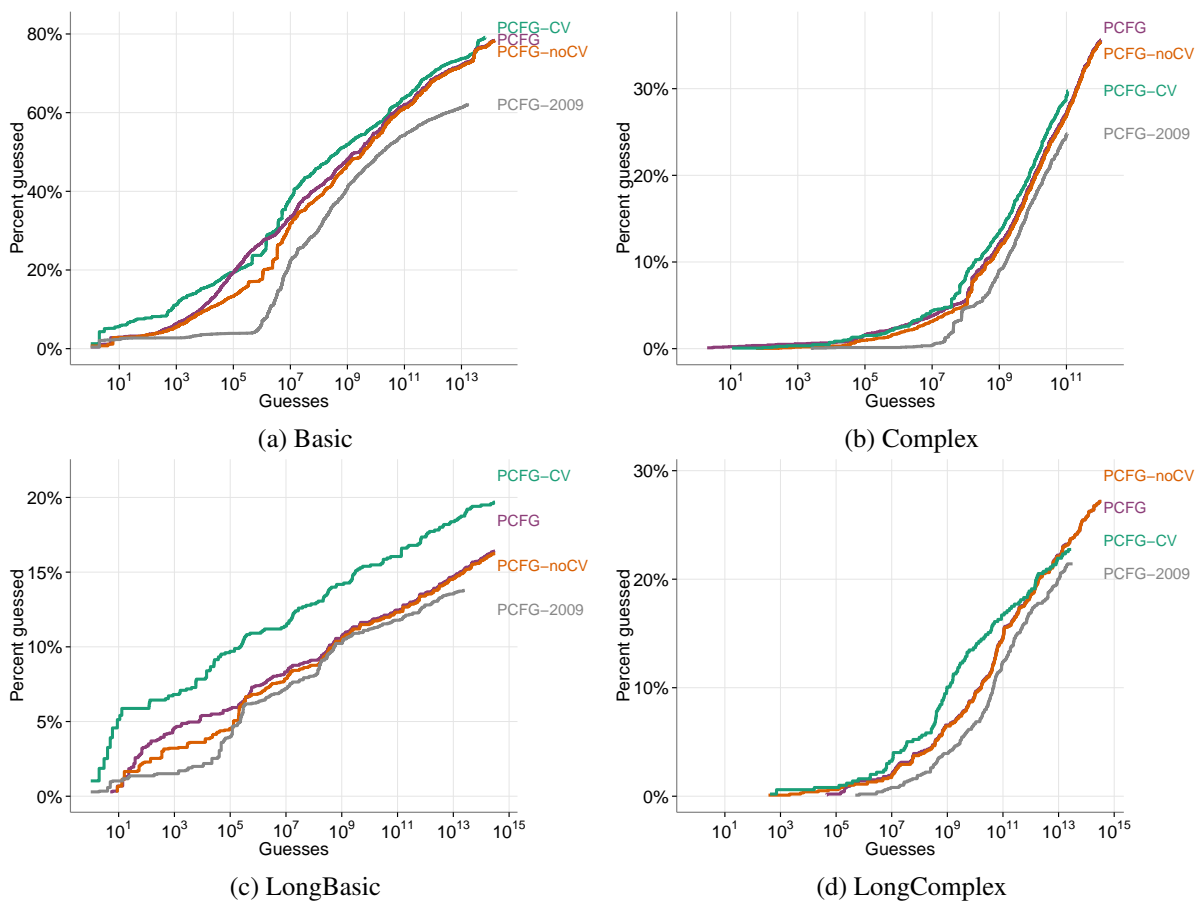


Figure 4.13: The guessing accuracy of the different PCFG configurations we tested.

4.4.1 Alternate PCFG Configurations

We tested four different PCFG configurations. As in Section 4.3, **PCFG** represents Komanduri’s implementation of PCFG [106], which assigns letter strings probabilities based on their frequency in the training data and assigns unseen strings a non-zero probability. For consistency across approaches, we prepend all policy-compliant elements of the training data in lieu of enabling Komanduri’s similar hybrid structures [106].

PCFG-noCV is the same as PCFG, but without the training data prepended. **PCFG-CV** is equivalent to PCFG-noCV except for using two-fold cross-validation. In each fold, we used half of the test passwords as additional training data, with a total weighting equal to the generic training data, as recommended by Kelley et al. [104]. **PCFG-2009** is our approximation of the original 2009 Weir et al. algorithm [197] in which alphabetic strings are assigned uniform probability and unseen terminals are a probability of zero.

As shown in Figure 4.13, prepending the training data and performing cross-validation both usually result in more efficient guessing, particularly for Long and LongBasic passwords. All three other configurations outperform the original PCFG-2009 implementation.

4.4.2 Alternate JTR Configurations

We next separately analyze the sets of JTR mangling rules we combined in Section 4.3. **JTR_stock** represents the 23 default rules that come with JTR. **JTR_SpiderLabs** represents 5,146 rules published by KoreLogic during the 2010 DEF CON “Crack Me If You Can” password-cracking contest [110], later reordered for guessing accuracy by Trustwave Spiderlabs [179].

As detailed in Section 4.2.3, our standard JTR configuration used JTR_stock followed by JTR_SpiderLabs. In isolation (Figure 4.14), JTR_stock rules were far more efficient guess-by-guess than JTR_SpiderLabs. Unfortunately, however, they quickly ran out of guesses. We exhausted JTR_stock in making fewer than 10^9 guesses for Basic passwords. More crucially, we made fewer than 10^5 guesses that were valid Complex passwords before exhausting these rules. Thus, any analysis of passwords that uses only the stock rules will vastly underestimate the guessability of passwords that contain (or are required to have) many different character classes.

The sharp jumps in the proportion of Complex and LongComplex passwords guessed by JTR_SpiderLabs result from one group of 13 rules. These rules capitalize the first letter, append digits, append special characters, and append both digits and special characters.

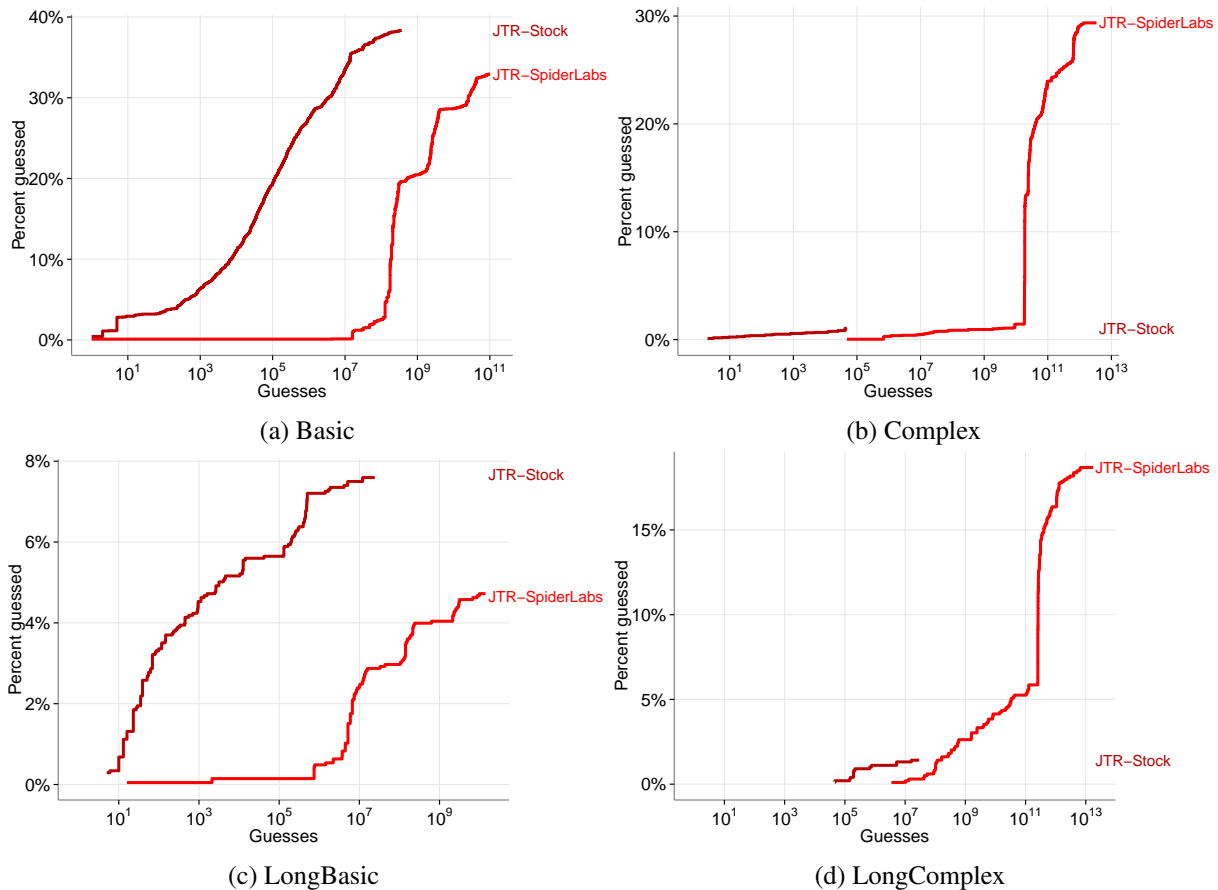


Figure 4.14: The guessing accuracy of JTR rules.

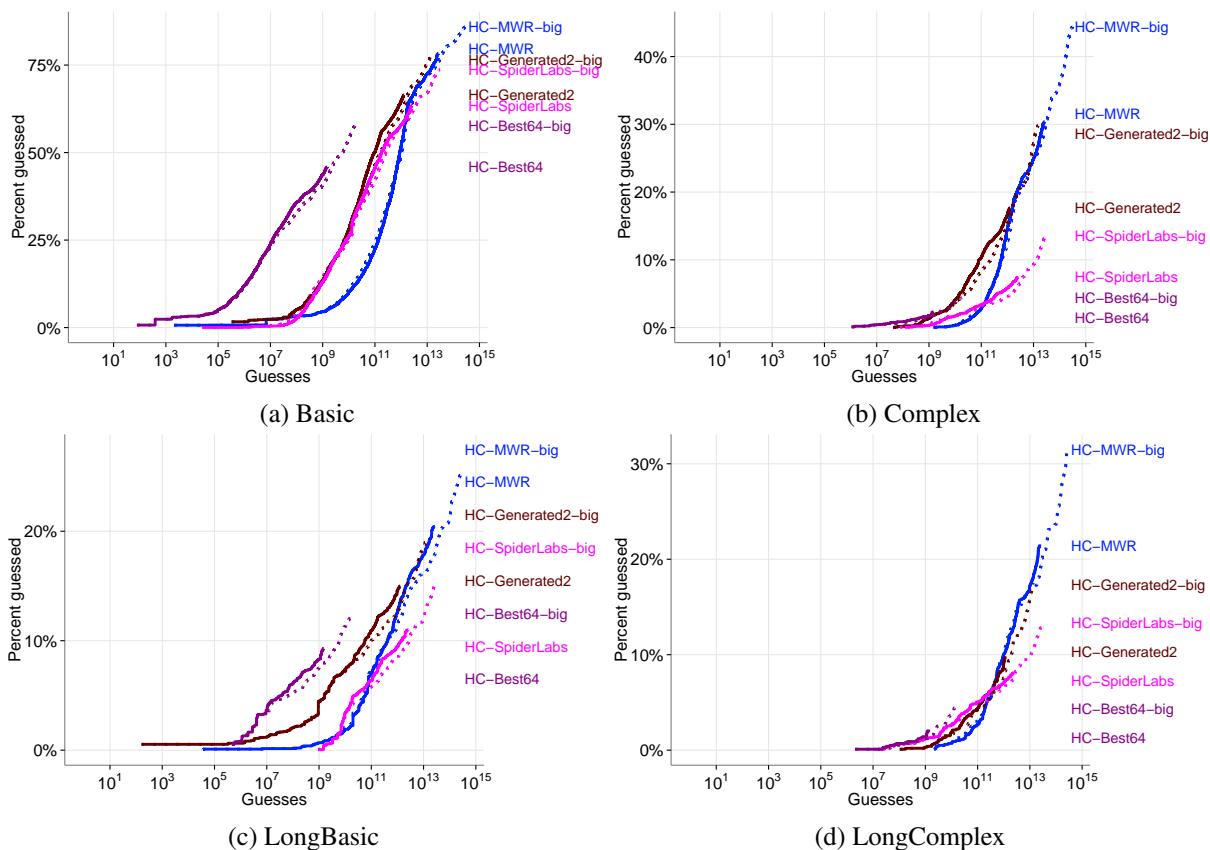


Figure 4.15: The guessing accuracy of Hashcat using four different sets of mangling rules. We tested each set with the wordlist used elsewhere in our analyses, as well as a larger (**-big**) wordlist.

4.4.3 Alternate Hashcat Configurations

We tested eight Hashcat configurations and chose the one that best combined efficient early guessing with successfully continuing to guess passwords into the trillions of guesses. These configurations consist of four different sets of mangling rules, each with two different wordlists. The smaller wordlist was the same one we used in all other tests (Section 4.2.2). The larger wordlist augmented the same wordlist with all InsidePro wordlists¹ in descending frequency order and with duplicates removed.

Our four sets of mangling rules are the following:

Hashcat_best64: Although Hashcat does not have a default set of mangling rules, the Best64 mangling rules are often used analogously to JTR’s stock rules.

Hashcat_generated2: Hashcat comes with a second set of mangling rules, “generated2.” This set comprises 65,536 rules. Dustin Heywood of ATB Financial created them by randomly generating and then testing hundreds of millions of mangling rules over 6 months (2013-2014) on a 42-GPU cluster. The rules were optimized by Hashcat developers by removing semantic equivalents.

Hashcat_SpiderLabs: We performed a manual translation to Hashcat of the SpiderLabs JTR rules (Sec-

¹<http://www.insidepro.com/dictionaries.php>

tion 5.3), which entailed removing clauses mandating minimum criteria; such rules are not permitted in oclHashcat.

Hashcat_MWR: We collaborated with with Matt Marx of MWR InfoSecurity to obtain the set of 1.4 million mangling rules he uses for password auditing [81, 134]. Following his suggestion, we augmented these rules with the aforementioned SpiderLabs rules.

Using the smaller wordlist, we exhausted all four sets of mangling rules. With the larger wordlist, we did not exhaust any set of rules. The curves in Figure 4.15 that use this larger dictionary have **-big** appended to the name and are graphed with dotted, rather than solid, lines.

We present the results of these eight configurations in Figure 4.15. True to their name, the Hashcat.best64 rules were the most efficient at guessing passwords. Unfortunately, they ran out of guesses using the smaller wordlist after only 10^9 guesses. For Complex and LongComplex passwords, Hashcat.best64 therefore guesses only a fraction of the number possible using the other sets of mangling rules, albeit in far fewer guesses. While not the most efficient guess-by-guess, the Hashcat_MWR rules eventually guessed the largest proportion of the different sets, most notably the Complex and LongComplex sets.

4.4.4 Ecological Validity

To better understand how well our password sets, collected for research studies, compare to real plaintext passwords from major password leaks, we compared the accuracy of the four automated cracking approaches in guessing Basic passwords and the following two comparable sets of leaked passwords:

Basic_{rockyou}: 15,000 passwords randomly sampled from those containing 8+ characters in the RockYou gaming website leak of more than 32 million passwords [187]

Basic_{yahoo}: 15,000 passwords randomly sampled from those containing 8+ characters in the Yahoo! Voices leak of more than 450,000 passwords [77]

We found a high degree of similarity in the guessability of the Basic passwords collected for research and the leaked passwords. As shown in Figure 4.16, the four automated cracking approaches followed similar curves across the research passwords and the leaked passwords.

This similar guessability is notable because our analyses depend on using passwords collected by researchers for two reasons. First, no major password leak has contained passwords contained under strict composition requirements. Furthermore, in contracting experienced humans to attack the passwords, it was important to have them attack passwords they had not previously examined or tried to guess. Presumably, these experienced analysts would already have examined all major password leaks.

In Section 4.3, we reported how different approaches were impacted differently by the number of character classes contained in Basic passwords. When we repeated this analysis for Basic_{rockyou} and Basic_{yahoo} passwords, we found similar behavior (Figure 4.17). PCFG was more successful at guessing passwords containing two character classes, as opposed to only a single character class. PCFG only guesses strings that were found verbatim in its training data, which we hypothesize might be the cause of comparatively poor behavior for passwords of a single character class.

4.5 Conclusion

We report on the first broad, scientific investigation of the vulnerability of different types of passwords to guessing by an expert attacker and numerous configurations of off-the-shelf, automated approaches frequently

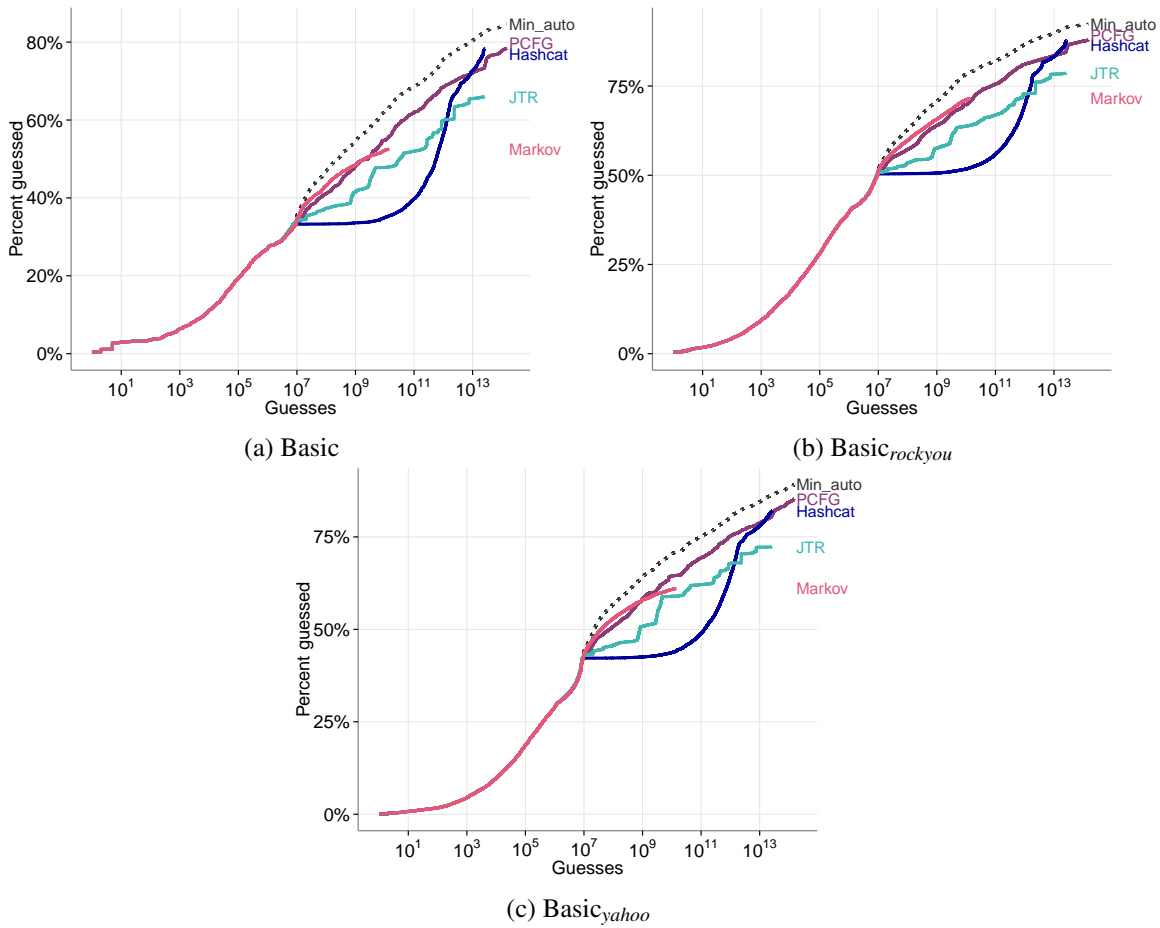


Figure 4.16: The four automated cracking approaches targeting the Basic password set, 15,000 passwords sampled from the RockYou leak, and 15,000 passwords sampled from the Yahoo leak.

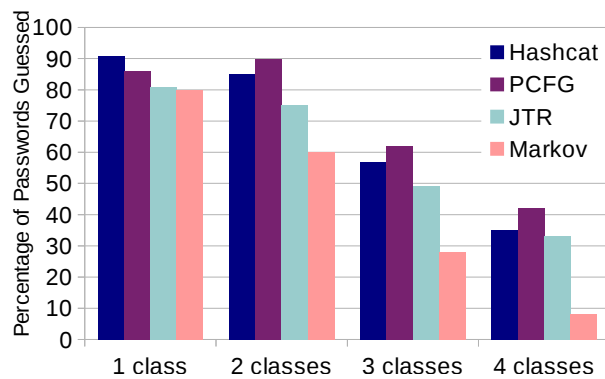


Figure 4.17: Combined percentage of Basic_{rockyou} and Basic_{yahoo} passwords each approach guessed by the number of character classes in the password.

used by researchers. We instrument these approaches, including both adversarial tools and academic research prototypes, to enable precise, guess-by-guess comparisons among automated approaches and between them and the expert.

We find that running a single guessing algorithm, particularly in its out-of-the-box configuration, often yields a very poor estimate of password strength. However, using several such algorithms, well-configured and in parallel, can be a good proxy for passwords' vulnerability to an expert attacker. We also find that while coarse-grained research results targeting heterogeneous sets of passwords are somewhat robust to the choice of (well-configured) guessing algorithm, many other analyses are not. For example, investigations of the effect on password strength of password characteristics, such as the number of character classes and the use of character substitutions, can reach different conclusions depending on the algorithm underlying the strength metric.

We hope our investigation of the effectiveness of many configurations of popular guessing approaches will help facilitate more accurate and easily reproducible research in the passwords research community. To that end, we have created a Password Guessability Service (PGS) [37] that enables researchers to submit plaintext passwords and receive guessability analyses like those we have presented. At the time of writing, other researchers have already used PGS to validate a scale for security behavior intentions [64] and evaluate a password-strength meter using advanced heuristics [199], among others. We particularly encourage researchers investigating password-cracking algorithms to contribute to this service to improve the comparability of experiments.

The more accurate techniques for modeling passwords we developed in this chapter enabled our subsequent experiments, most directly those comparing users' perceptions of password security to the reality of password-guessing attacks [181] (Chapter 6) and comparing password-composition policies [156]. They have also enabled our experiments developing better client-side estimations of password strength using neural networks [130] and our evaluation of our improved password-strength meter with data-driven feedback, which we detail in Chapter 7.

Chapter 5

The Art of Password Creation: Semantics and Strategies

5.1 Introduction

To improve proactive password checking and the advice given to users about password creation, such as in the data-driven password-strength meter I present in Chapter 7, an important precursor is a deep understanding of how precisely users structure passwords. By understanding what patterns, structures, and strategies are common, researchers can guide users away from common, predictable approaches and towards more secure choices. In this chapter, I therefore delve into users' password-creation strategies, as well as the structure and semantics of passwords.

Many characteristics of user-chosen passwords are widely known. For instance, it is common knowledge that passwords often contain dictionary words or names [56, 93], and that digits often appear at the end [108, 189]. Lists of common passwords circulate after every breach. However, far less is known about passwords' deeper structural and semantic properties that can make the difference between a password that is easy to guess and one that is not. These properties include how users combine and transform words while creating passwords, what semantic sources provide inspiration, and how users behave during password creation.

Our investigation is enabled by a novel combination of crowdsourcing and programmatic techniques to reverse engineer more than 45,000 passwords into a representation illuminating their structure and semantics. To reveal otherwise obscured password elements, crowdworkers reverse engineered each password into semantic "chunks" and undid character substitutions. For example, the password `~Cowscomehom3` became *till the cows come home*; the crowdworkers realized the tilde stood for "till the." To understand how patterns we discovered corresponded to attackers' ability to guess passwords, we also modeled each password's vulnerability to two major password-guessing approaches.

Building on this preprocessing phase, we offer three main contributions that collectively provide new insights into users' habits and reveal subtle, yet common, patterns that should be discouraged.

Blase Ur, Saranga Komanduri, Lujo Bauer, Lorrie Faith Cranor, Nicolas Christin, Adam L. Durity, Phillip (Seyoung) Huh, Stephanos Matsumoto, Michelle L. Mazurek, Sean M. Segreti, Richard Shay, Timothy Vidas. The Art of Password Creation: Semantics, Strategies, and Strategies. Unpublished. Excerpts from this work were previously published as Blase Ur, Saranga Komanduri, Richard Shay, Stephanos Matsumoto, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Patrick Gage Kelley, Michelle L. Mazurek, Timothy Vidas. Poster: The Art of Password Creation. In Proc. IEEE Symposium on Security and Privacy Poster Session, 2013.

First, we provide a large-scale quantification of the use of character substitutions in passwords. Between 5% and 18% of passwords, depending on the source, contain substitutions, and the top twenty mappings account for more than 77% of all substitutions. This ground-truth data can enable proactive password checking to account for substitutions.

Second, we report on password semantics, identifying choices that users should avoid. We compare frequencies of words across different password sets and natural language. For example, we find that up to 5% of words used in passwords are contained in a 247-word list of pet names and many others appear in small dictionaries. In addition, roughly half of the passwords we examined that contained any content from a dictionary contained a common multi-word phrase. We use Wikipedia’s categorization system to further analyze semantics, discovering previously unreported patterns.

Finally, we delve into the individual steps of password creation. We analyze sequences of attempts to comply with a password-composition policy to understand how users modify passwords across attempts. Unexpectedly, forcing users to comply with strict policies sometimes reduces security; over 20% of users who generated a completely new password after their original attempt was rejected made a less secure password. In contrast, certain types of small modifications nearly always made a password harder to guess. We also examine the impact of suggestions (e.g., “Add a digit”) during password creation, finding that users do follow these suggestions.

Drawing on these analyses, we discuss directions for improved guidance during password creation and improvements for proactive password-checking mechanisms.

5.2 Datasets

We first introduce the datasets we analyzed. To obtain the benefits of both real-world and (richer) experimental data, we examined in parallel data leaked from real websites and data collected by researchers for experiments.

We used two publicly available sets of leaked passwords: more than 32 million passwords from the RockYou gaming website [187] and more than 450,000 passwords from Yahoo! Voices [77]. These passwords appear to have been created under minimal or no password-composition requirements. To analyze some of these passwords while using the bulk of these passwords as training data, we created two subsets:

- **RockYou8:** 15,000 passwords randomly selected from those containing 8+ characters in the RockYou leak [187]
- **Yahoo8:** 15,000 passwords randomly selected from those containing 8+ characters in the Yahoo! Voices leak [77]

Compared to passwords created for studies, these sets are much larger and protected real accounts. These sets do not, however, include data about the password-creation process.

Using leaked passwords for research raises important ethical concerns. We consider our use justifiable for several reasons. The datasets are publicly available, so that our analysis causes no additional harm. Further, because attackers are likely to use leaked passwords maliciously, it is important to use this data when considering how to strengthen passwords.

We also analyze passwords we collected on Amazon’s Mechanical Turk (*MTurk*) for previous studies of password-composition policies [104], password-strength meters [182], and long passwords [157]. Those previously published papers detail the methodology and participant demographics of each study; we do not

duplicate that information. While these datasets are not as large as leaked sets, their varied composition policies and detailed data on the password-creation process enable many analyses. Although passwords collected for research studies are not perfect proxies for real data, two recent studies [67, 126] have found they are an effective facsimile when real passwords under those policies cannot be obtained.

We used experimental data collected in nine conditions varying either in composition policy or use of a meter. Passwords in all sets were required to be at least eight characters long.

Three conditions specified only a minimum length:

- **basic8:** Passwords contain 8 or more characters
- **basic16:** Passwords contain 16 or more characters
- **basic20:** Passwords contain 20 or more characters

Three additional conditions had special characteristics:

- **blacklist:** Passwords cannot be in a set of 5×10^9 passwords generated using the PCFG algorithm [197]
- **meterStandard:** A visual password meter encouraged longer passwords or additional character classes. Participants were given suggestions like “Consider adding a digit...”
- **meterStringent:** Similar to meterStandard, except the password meter fills only one-half or one-third as quickly in order to encourage longer, more complex passwords

The final three conditions required that passwords contain some number of different character classes. Uppercase letters, lowercase letters, digits, and symbols each form a class.

- **3class12:** Passwords contain 12 or more characters, including characters from 3+ character classes
- **3class16:** Passwords contain 16 or more characters, including characters from 3+ character classes
- **comp8:** Passwords contain 8 or more characters, include all 4 character classes, and are not in the free Openwall dictionary¹ after removing non-alphabetic characters

5.3 Methodology

We first describe our novel method for transforming passwords into a representation that illuminates structures and semantics. We then highlight natural-language data we used to analyze password semantics. Finally, we discuss how we studied password creation and measured password security.

5.3.1 Reverse Engineering Passwords

Humans derive passwords from semantically significant content [56, 93, 124, 188]. These semantics are easy to study when they appear directly and in isolation in a password (e.g., *monkey23*). In other cases, this content is transformed and spaces are removed, burying the semantics inside a password (e.g., *d0llf1n4911* is likely derived from dolphin, while *Charlie&the1* might refer to “Charlie and the Chocolate Factory”).

To investigate password structures and semantics, one must reverse these transformations and separate semantically significant chunks of a password. It is tempting to perform this analysis algorithmically. For example, Jakobsson and Dhiman used heuristics to automatically search for leetspeak, misspellings, and

¹<http://download.openwall.net/pub/wordlists/>

concatenation of dictionary words in passwords [102], while Veras et al. crafted specialized dictionaries to guess passwords with semantic structures [188].

We found that automated approaches run into many subtle, yet significant, barriers. User-created passwords draw from variegated sources encapsulating large amounts of context, and the many possible substitutions introduce ambiguity. For example, @ takes on different meaning in *L@xicondevill* (“e”), *cashCow@3137* (“@”), *Sex@thebeach69* (“at” or “on”), and *Rectlfy\$\$@nctlty* (“a”). Similarly, “4” has very different functions in *4screamingpancakes* (“four”), *4Everblessed* (“for”), and *4qrdg@G9Q* (itself). Any heuristic, dictionary, or mapping of substitutions broad enough to capture human behavior introduces many false positives that human intuition would reject, such as incorrectly discovering “Glest” (the name of a game) in the seemingly random *G134TG\$*.

To enable the analyses we report, as well as to gather ground-truth data for future improvements to automated password analysis, we crowdsourced the reverse engineering of passwords. Maintaining the advantage of human intuition in a scalable way, we had MTurk crowdworkers undo any substitutions and manually separate each password into its semantic parts. Reverse engineering passwords is a tricky task, even for humans, so we adopted best practices from Ipeiritos et al. [100] to bolster quality. This approach uses a “gold standard” subset of answers and an expectation-minimization (EM) algorithm for worker scoring that weights answers based on how well they match either the gold standard or other workers.

Each password in our dataset that contained any letters was first reverse engineered by three MTurk workers, who were given extensive instructions with examples. We asked workers to convert digits and symbols to letters where applicable, correct misspellings, combine word pieces that might be split (e.g., “sch46ool” → “school”), and separate distinct words or patterns with spaces. We asked workers not to modify any string that appeared random (e.g., “lonkiey”). We required workers to pass a qualification test in which they reverse engineered four passwords with known answers. Independently, an “expert” team of a security researcher and a researcher with a linguistics background reverse engineered 5% of the passwords to produce the gold standard. Workers were scored using the EM algorithm and the gold-standard answers [100]. We repeated this process until every answer was agreed on by either the experts or at least two workers with 90%+ scores.

Our crowdworkers successfully reversed many subtle transformations that would have been difficult to detect automatically. As mentioned earlier, they realized the semantic equivalence of tilde and “till the” in recording *~Cowscomehom3* as “till the cows come home.” Similarly, they identified *primo4sho* as “primo for sure” and *2sweet2b4gotten86* “too sweet to be forgotten 86.” Workers frequently noticed semantic meaning in non-letters. For example, they used knowledge of drug slang to turn *bernie44weed4420* into “bernie 44 weed for 420” and knowledge of movies to delimit *jamesbond0071986* as “james bond 007 1986.”

5.3.2 Semantic Analyses

We used three sources of natural-language data to analyze the semantics of the reverse-engineered passwords, illuminating semantic properties of passwords far more comprehensively than previous semantic analyses [56, 124, 145, 146, 188, 207].

Dictionaries and wordlists were our first source of semantic data. Based on prior work [56, 124, 188] and our own observations, we constructed the following wordlists: pet names,² the most populous cities and

²<http://www.babynames.com/Names/Pets/>

countries according to Wikipedia, US surnames,³ and popular US first names (1960-2000).⁴ We also used dictionaries from the Corpus of Contemporary English,⁵ the Unix dictionary, and the Urban Dictionary.⁶

To systematically investigate the degree to which common phrases appear in passwords, we collected and constructed frequency-orderd tables of n -grams as our second source of semantic data. In this context, an n -gram is a phrase containing n words. For example, the 3-gram “rock on Chicago” appeared in one password we analyzed.

We used one large, publicly available n -gram corpus:

Google: 1-grams through 5-grams from over 1 trillion words of English text indexed by Google in 2006 [83].

Given the age and breadth of the Google corpus, we also built 1-gram through 7-gram tables from the following five sources that appeared commonly in the results of automated Google queries for phrases in reverse-engineered passwords:

Books: The text of the over 30,000 English-language books available on Project Gutenberg.

IMDB: All movie and TV quotes on IMDB.com.

Lyrics: Lyrics to all 469,000 songs on SongLyrics.com.

Twitter: 50,000,000 tweets collected in May 2011 [117].

Wikipedia: All English-language text on Wikipedia.

We chose our third source of semantic data, the categorization system on Wikipedia, to gain further insight into the semantic sources that users commonly integrate into passwords. Wikipedia articles can optionally be tagged with categories; for example, “captain planet” is tagged with “Environmental television” and “Superheroes by animated series,” among others. We made automated Wikipedia queries for the longest phrase from each reverse-engineered password for which we could find a Wikipedia entry and recorded the categories returned. Because categories are non-hierarchical and often redundant, we manually examined the 14,000 categories most frequently contained in passwords and iteratively merged similar categories.

5.3.3 The Process of Password Creation

The nine sets of passwords collected for research included detailed data about password creation, illuminating the process of password creation. No leaked password set contains this sort of metadata, forcing us to restrict these analyses to passwords collected for research. The first creation analysis we report examines how users modify a password to comply with a password-composition policy. We considered only participants who at least once submitted a password that did not meet the requirements of the specified policy before eventually submitting a compliant password. We counted consecutive attempts to submit the same password as a single attempt. This analysis focused on the six sets in which an 8+ character password might not comply: blacklist, comp8, 3class12, basic16, 3class16, and basic20.

Our second creation analysis used keystroke data during password creation to gauge the impact of specific suggestions. Such suggestions were provided only in the meterStandard and meterStringent sets. Based on our hypothesis that participants might only be receptive to suggestions when they were unsure how to proceed, we divided keystrokes into those that followed a pause in creation and those that did not. To identify pauses, we calculated the inter-keystroke time for sequential keystrokes and used k -means clustering to split

³<http://names.mongabay.com/>

⁴<http://www.ssa.gov/oact/babynames/>

⁵<http://corpus.byu.edu/coca/>

⁶<http://www.urbandictionary.com>

the times into two groups. If those two groups were separated by at least the median inter-keystroke time, we labeled keystrokes in the latter group as those following pauses.

5.3.4 Security Analysis

To quantify password security, we simulated two popular approaches to password cracking. We calculated each password’s guess number, or the number of guesses that algorithm trained in that specific way would require to guess that password. This approach quantifies the strength of individual passwords under conditions modeling adversarial cracking, rather than theoretical attacks [104]. For each approach, we simulate trillions of guesses. We only count guesses compliant with a set’s composition policy.

The first of the two approaches we simulated, John the Ripper (“*JtR*”),⁷ is representative of tools widely used by adversarial crackers [78]. We simulate JtR using the `john-1.7.9-jumbo-7` “wordlist” mode, which generates guesses based on a user-provided dictionary and mangling rules, or transformations to apply to dictionary entries. We used mangling rules released for the 2010 DEFCON Crack Me If You Can password-cracking contest and reordered by SpiderLabs for efficiency.⁸

The second approach, a probabilistic context-free grammar (*PCFG*) developed by Weir et al. [197], has been commonly used in academic studies [104, 126]. This approach uses existing passwords to model the structure and contents of passwords. We calculate PCFG guessability using Kelley et al.’s modification of Weir et al.’s approach [104]. We conduct two-fold cross validation to match a target password’s policy more closely than using only leaked passwords.

We used identical training data for both approaches. These data included large sets of real passwords leaked from MySpace, RockYou, and Yahoo! (excluding those removed for testing), as well as all words contained in the Google web corpus [83] and a 250,000 word inflection⁹ dictionary.

5.4 Results

We analyzed many aspects of how users choose, structure, transform, and modify passwords. We focus on analyses that provide insight into predictable behaviors that should be discouraged, with the goal of both informing the advice given to users and improving proactive password checking.

After briefly describing general characteristics about the passwords we studied, we detail how users substitute characters in passwords. We then analyze how users structure passwords and the semantic sources from which users draw inspiration in crafting passwords. Finally, we delve into the steps of password creation to understand how users modify passwords to comply with password-composition policies and whether they take suggestions during password creation.

5.4.1 General Password Characteristics

To provide a baseline understanding of our data, Table 5.1 shows traditional analyses on password length and character class usage. Echoing prior work [32, 56, 191, 207], passwords often started with capital letters and ended with digits. For example, in `comp8`, which required all four character classes (uppercase

⁷<http://www.openwall.com/john/>

⁸<https://github.com/SpiderLabs/KoreLogic-Rules>

⁹<http://wordlist.sourceforge.net>

Table 5.1: Password length and character class usage by set. Passwords frequently began with an uppercase letter (“UC First”) and ended with a digit (“Digit Last”).

Set	#	Length			% of Passwords		
		Mean	σ	Med	UC First	Digit Last	All Digit
basic8	3,063	9.6	2.2	9	14	54	8
RockYou8	15,000	9.8	2.2	9	8	58	15
Yahoo8	15,000	9.4	1.5	9	9	59	3
blacklist	1,158	9.9	2.2	9	11	45	6
meterStandard	1,398	11.5	3.5	11	28	56	4
meterStringent	569	14.6	7.2	12	34	51	4
comp8	4,246	10.6	2.5	10	76	36	–
3class12	993	13.8	2.6	13	74	39	–
basic16	2,058	18.1	3.3	17	16	44	4
3class16	983	18.3	4.0	17	72	36	–
basic20	988	22.9	4.7	21	21	41	4

letter, lowercase letter, digit, symbol), 76% of passwords began with a capital letter. The last character in comp8 passwords was most commonly a symbol, whereas the penultimate character was usually a digit. Also matching prior work [108], users often exceeded minimum composition requirements. Between 8% (RockYou8) and 34% (meterStringent) of passwords that did not require multiple character classes nevertheless began with a capital letter, while 18%–32% of characters per set were digits.

5.4.2 Character Substitutions in Passwords

Password crackers have long hypothesized [102] that humans mangle words and phrases in passwords by substituting strings of characters in place of others. Many such substitutions are possible (e.g., a → @, at → @, o → @), which makes automatically identifying them tricky and prone to false positives and false negatives. We rely on the crowdsourced human intuition that underlies our reverse-engineered passwords to provide the first quantification of these substitutions, finding that they are both very predictable and perhaps less common than one might expect given the prevalence of these sorts of mangling in password crackers’ rulesets. Furthermore, this ground-truth data can be leveraged to proactively identify and discourage common substitutions, which we do in our data-driven meter (Chapter 7).

As shown in Table 5.2, the incidence of substitutions was relatively low. More commonly, passwords contained digits or symbols alongside words, rather than as a result of applying substitutions. However, the use of substitutions varied significantly across sets ($\chi^2_{10}=1132$, $p<.001$), from 4.1% to 18.0%. The three sets that required multiple character classes (comp8, 3class12, 3class16) most frequently contained substitutions, suggesting that some users cope with requirements by employing substitutions. Substitutions were correlated with modest decreases in guessability. PCFG cracked 46% of passwords with substitutions and 62% of passwords without them, while JTR cracked 41% of passwords with substitutions and 61% of passwords without them. Substitutions are not a silver bullet for making passwords secure.

Table 5.2: The percentage of passwords with substitutions and, among those with substitutions, the mean number of *substitutions* and *distinct mappings* from one string to another.

Set	% with	Substitutions		Distinct	
		Mean #	σ	Mean #	σ
RockYou8	4.1	1.3	0.6	1.2	0.5
Yahoo8	4.9	1.5	0.9	1.3	0.6
basic8	5.1	1.8	1.3	1.4	0.7
meterStandard	8.5	1.8	1.1	1.5	0.8
meterStringent	9.8	2.0	1.3	1.6	1.0
blacklist	7.7	1.8	1.1	1.5	0.8
comp8	15.3	1.7	1.0	1.5	0.8
basic16	5.3	2.4	1.8	1.8	1.2
basic20	6.2	2.8	2.2	1.8	1.2
3class12	14.8	1.9	1.3	1.6	0.8
3class16	18.0	1.8	1.4	1.4	0.8

Table 5.3: The 20 most frequent substitution mappings. These 20 most frequent mappings account for 77% of all character substitutions we observed.

Mapping	% of total	Mapping	% of total
o \rightarrow 0	16.2	to \rightarrow 2	1.7
e \rightarrow 3	12.4	l \rightarrow 1	1.7
a \rightarrow @	7.8	i \rightarrow !	1.5
i \rightarrow 1	7.6	too \rightarrow 2	1.4
for \rightarrow 4	6.8	one \rightarrow 1	1.2
s \rightarrow \$	3.9	c \rightarrow k	1.2
a \rightarrow 4	3.1	er \rightarrow a	1.0
o \rightarrow u	2.8	and \rightarrow &	0.9
s \rightarrow z	2.5	i \rightarrow y	0.8
s \rightarrow 5	2.0	for you \rightarrow 4u	0.7

We also quantified each mapping of one string of characters to another. For example, we define *pa\$\$word* to contain two substitutions, both using the single mapping “s \rightarrow \$.” While we observed 534 distinct mappings across sets, the top five account for over half of all substitutions used. Furthermore, the 20 most common mappings (Table 5.3) capture 77% of all substitutions. Table 5.2 includes the mean number of substitutions and distinct mappings per password.

Absent this sort of ground-truth data on substitutions in passwords, a major difficulty in proactively identifying substitutions when users create passwords is the very large number of potential transformations, as well as uncertainty about what substitutions real users actually make. Armed with our data, proactive password checkers can try just the twenty most common substitution mappings to identify and discourage this predictable behavior.

5.4.3 Passwords Semantics

Users base passwords on semantically significant content to make them memorable [108, 136, 191], yet this semantic significance makes passwords more predictable [188, 197]. We report on four types of semantic analyses, and discuss directions for discouraging the use of common semantic categories.

Word Usage Across Sets

We first examined the degree to which the lexicon of words that were used in each password sets resembles other password sets, as well as English. The knowledge gained from this analysis is important because many password analyses use information from one source to model passwords from another source for both helping users avoid passwords and for cracking passwords. We found the lexicon to be relatively similar across password sets, yet always far from English.

We used the square root of the Jensen–Shannon Divergence (*JSD*) pairwise across sets for all alphabetic chunks (those containing only letters). *JSD* measurements range from 0 to 1; smaller values indicate greater similarity. Password sets were relatively similar to each other, with pairwise *JSD* values from 0.24 to 0.48. Password sets collected for research were particularly similar to each other and to our baseline, which was the pairwise *JSD* for random samples of 3,000 RockYou8 passwords. Although RockYou8 and Yahoo8, the two leaked datasets, were relatively similar to each other (*JSD* of 0.33), we found the alphabetic chunks from Yahoo8 to more closely resemble the experimental sets than the chunks from RockYou8 did. None of the password conditions had distributions particularly similar to the Corpus of Contemporary American English, with *JSD* ranging from 0.70 to 0.75.

We also examined which words contributed most to the divergence between sets. Words that appeared disproportionately in a set tended to be related to the source of passwords. For example, Yahoo8 passwords, leaked from Yahoo! Voices (originally named Associated Content), disproportionately contained the chunks “associated,” “content,” and “writer.” Experimental passwords made by MTurk users who made a simulated email password disproportionately contained “amazon,” “mechanical,” and “mail.”

Dictionary Words in Passwords

Next, we investigated the degree to which different wordlists, collectively termed *dictionaries*, captured the words contained in our reverse-engineered passwords. Based on our own manual analyses and prior work [29, 32, 114, 127, 188, 191], we collected and constructed specialized dictionaries that we hypothesized would represent common words. For each set, we analyzed alphabetic chunks of three or more characters, excluding keyboard patterns for dictionary membership.

Overall, we found that users’ choice of the words in passwords was quite poor. Across all sets, less than 0.5% of chunks did not appear in any dictionary, and even small dictionaries of pet names and places (countries, cities, and states) contained many of the words used in passwords. As shown in Table 5.4, up to 5% of alphabetic chunks per set were contained in a 247-entry list of common pet names. As in prior work [114, 127, 188], locations and first names were common. More general dictionaries of English words (the Unix dictionary), slang (the Urban dictionary), and all words written on the English Wikipedia contained up to 95% of the chunks.

Table 5.4: The percentage of alphabetic chunks in dictionaries of *pet* names, *places* and locations, first *names*, the *Unix* dictionary, the Urban dictionary (*slang*), and Wikipedia.

	Pets	Places	Names	Unix	Slang	Wikipedia
Size	247	12,691	42,103	72,843	816,186	8,576,716
basic8	3.5	16.6	31.1	51.5	78.3	89.3
RockYou8	1.9	12.1	27.6	37.3	66.4	79.5
Yahoo8	2.0	13.0	26.1	45.6	71.2	81.1
blacklist	3.3	14.3	29.2	41.6	67.4	76.6
meterStandard	3.4	16.6	30.8	53.8	79.4	90.8
meterStringent	2.8	19.3	29.1	55.5	79.4	90.7
comprehensive8	2.6	11.7	23.3	38.4	65.2	78.8
3class12	5.1	23.8	31.8	72.1	88.7	92.9
basic16	3.3	20.8	32.6	60.4	80.8	88.9
3class16	4.6	23.1	32.7	73.1	90.7	95.1
basic20	4.8	25.7	31.5	77.1	90.8	95.0

Conceptual Categorization

To delve further into the semantic concepts used in passwords, we leveraged Wikipedia’s categorization system, which tags articles with categories like “food/beverage,” “history,” and “religion.” We searched for the Wikipedia article containing the longest sequence of alphabetic chunks from each password. As a result, we categorized both multi-word phrases and individual words. As described in the methodology, we counted distinct tags from each article separately, yet manually merged redundant categories.

A handful of concepts predominated. Echoing prior work [56, 114, 127, 188], locations and names were the top categories (Table 5.5). Our list, however, identified many concepts that have not been reported widely, including food/beverage, band names, companies/products, and nature. We hypothesize that the thread that binds many of these categories is that they are things users like or things that are near the user during password creation. These concepts should be discouraged. We performed this analysis both with and without removing duplicate words and phrases, finding similar results.

Multi-Word Phrases

While the inclusion of multiple words in a password can make them harder to guess, the security benefit is mostly lost if the sequence of words forms a common phrase. Using tables of n-grams (n-word phrases) that we built, as described in the methodology, we found that many multi-word sequences in passwords do indeed form common phrases. This practice should be more actively discouraged to improve security.

Of the 45,054 unique reverse-engineered passwords across our 11 sets, 39,211 passwords contained at least one alphabetic chunk, while 19,662 of these contained two or more alphabetic chunks. Of these 19,662 passwords that contain multiple words, 13,115 passwords (66.7%) matched a phrase in our corpora. These common phrases can be modeled far more easily than sequences of unrelated words.

Some particularly long passwords contained many words, yet the relationship between these words was often easily modeled. A total of 36 passwords contained a 7-gram from one of the corpora, including *michaeljacksonisthekingofpop* and *thisisthesongthatneverends*. An additional 64 passwords contained a 6-gram (e.g., *Theriseandfalofziggy1@*), 161 (0.8%) contained a 5-gram, 595 (3.0%) contained a 4-gram,

Table 5.5: The 15 most common Wikipedia categories after removing duplicate words and phrases.

Category	#	Randomly selected example
locations	323	deutschland
names	299	madeline
food/beverage	291	root beer
celebrities/bands	288	wes anderson
movies/TV	244	arm slave
fictional characters	190	beavis and butthead
animals/insects	189	dolphins
music	131	freebird
companies/products	129	twinkies
religion	107	deuteronomy
flowers/plants	95	sunflower
sports	86	all-star
pop culture, idioms	85	yo momma
literature	85	frankenstein
historical events/figures	85	rasputin

Table 5.6: Among the 13,115 n-gram matches, the number *contained* in each corpus and the number contained *only* in that corpus. We also show the number of matches each corpus would guess *most efficiently*, ordering by frequency.

Corpus	Contained (%)	Only Match	Most Efficient
Google	3,111 (24%)	231	738
IMDB	5,234 (40%)	16	1,581
Lyrics	7,012 (53%)	117	2,888
Books	7,463 (57%)	209	1,059
Twitter	10,300 (79%)	747	3,813
Wikipedia	11,125 (85%)	1,242	3,036

1,949 (9.9%) contained a 3-gram, and 10,381 (52.8%) contained a 2-gram. Most phrases were common; three-quarters were among the 10% most frequent in their corpus.

Each of the six corpora we investigated contributed insight into the use of phrases in passwords. As shown in Table 5.6, each corpus reached some n-grams from passwords more quickly than any other corpus and contained some n-grams that were not in any other corpus. That said, the Twitter, Wikipedia, and Lyrics corpora were most efficient at finding n-grams, while the 7-year-old corpus was least efficient.

Using the reverse-engineered passwords, we next analyzed how users integrate semantic content, both individual words and phrases, into the actual passwords. Most users insert the semantic content directly, without modification. As shown in Table 5.7, two-thirds of passwords with semantic content began with the semantic content. Furthermore, 3,934 passwords (13.3%) were simply the word or phrase itself, ignoring capitalization. While neither PCFG nor JTR cracked most of these passwords in our tests, we trained these algorithms on released passwords, rather than on natural-language corpora.

While most passwords contained the semantic content as a contiguous substring, 4963 passwords (14.3%)

Table 5.7: How words and phrases appeared in passwords. An “x” indicates unrelated content.

Pattern	Instances	% Cracked	
		PCFG	JTR
{Word/Phrase}{x}	19,161	23.2%	14.1%
{Word/Phrase}	3,934	16.1%	16.8%
Mangled	3,834	14.7%	6.2%
{x}{Word/Phrase}	3,683	17.8%	7.2%
{x}{Word/Phrase}{x}	2,846	10.5%	3.8%
Phrase with insertions at wordbreaks	1,308	16.1%	1.9%

Table 5.8: The security impact of complying with a password-composition policy, comparing the original candidate password with the final, compliant password. If neither password was guessed by our cutoff, the security impact is unknown. We show the mean Levenshtein distance between the original and final password, as well as the mean of the cumulative distance across all retries.

Set	Retries		PCFG Guessability Impact			JTR Guessability Impact			Edit Distance	
	%	Mean #	Harder to guess	Unknown impact	Easier to guess	Harder to guess	Unknown impact	Easier to guess	Final	Total
blacklist	43.7	1.9	77.5	2.0	20.5	70.0	21.7	8.3	8.3	14.0
comp8	71.1	2.5	64.7	28.9	6.4	27.6	71.8	0.6	6.6	13.5
3class12	39.3	1.3	52.2	39.8	8.0	15.9	80.3	3.8	6.4	7.5
basic16	43.8	1.3	49.9	43.3	6.8	33.1	59.5	7.3	10.8	12.3
3class16	47.2	1.5	43.4	53.6	3.0	17.5	81.7	0.9	9.8	12.9
basic20	53.9	1.4	48.4	46.2	5.4	27.0	68.3	4.7	14.7	17.5

did not. Of those that did not, 1,308 (26%) inserted content at the breaks between words. For instance, “josh72quincy” contained digits inserted between “josh” and “quincy.” We coded a random selection of 251 of the 3,834 passwords that contained the n-grams in mangled form. Of the 251, 54% had used the types of transformation rules discussed earlier. An additional 38% used alternate spellings (e.g., “luv” instead of “love”), and 4.8% inserted characters into the middle of words.

5.4.4 The Process of Password Creation

To gain insight into the individual choices users make when creating a password, our final analyses delve into the steps of password creation. We use the knowledge gained from these analyses to pinpoint interventions that guide users towards more secure decisions. First, we examine how users modify passwords that do not comply with a password-composition policy, sometimes making a password easier to guess in the process. Second, we investigate the impact of providing suggestions about what character class to include, finding that users do take these suggestions during pauses in creation.

Table 5.9: The impact of modifying a non-compliant password to comply with a composition policy.

Strategy	#	% of retries employing strategy						PCFG		JTR	
		blacklist	comp8	3class12	basic16	3class16	basic20	% Harder to guess	% Easier to guess	% Harder to guess	% Easier to guess
Add to end	1,226	18	18	23	21	7	2	47	0	27	1
Change capitalization	1,021	6	5	2	1	12	1	47	9	12	6
Add to middle (inserted)	651	12	10	8	7	5	1	54	5	22	1
One substitution (non-133t)	573	5	2	2	2	7	3	37	15	9	4
Replace 2nd half of password	412	5	7	6	8	2	2	46	15	22	6
Replace 2nd half & change capitalization	318	2	2	2	2	3	2	53	20	10	5
Capitalize & add to end	309	2	2	1	1	4	0	59	1	45	0
Two substitutions (non-133t)	193	0	1	1	0	2	1	42	15	11	1
Add to front	183	1	3	5	3	1	1	55	2	30	1
Capitalize & add to middle	145	2	1	1	0	2	–	59	3	30	1
One substitution & change capitalization	138	1	1	0	–	2	–	52	16	21	6
Replace 1st half of password	126	1	1	0	1	1	0	44	21	21	0
Delete from middle	119	1	1	1	1	1	1	17	39	3	13
One substitution & add to end	84	3	2	1	1	1	–	46	8	26	4
Leet substitutions (any amount)	72	0	–	–	0	1	1	60	3	19	0
Repeated password	70	1	1	3	3	–	–	7	0	49	0
Transposed two adjacent characters	55	–	1	1	0	1	0	22	35	0	9
Capitalize & add to front	48	1	1	0	1	0	0	78	2	25	2
Two substitutions & change capitalization	40	–	0	–	0	1	–	55	18	28	3
<i>Other, dissimilar password</i>	5,538	33	33	40	44	45	73	52	23	20	9
<i>Other, similar password</i>	528	2	3	3	3	4	11	50	22	39	8

Policy-Compliance Strategies

To understand users' approaches for coping with password-composition policies, we next examined how they modified their candidate password after failing to comply with a policy. We focused on ease of compliance, the strategies participants employed, and the security impact of these strategies. We find some strategies lead to passwords that are often easier to guess, whereas others appear prudent, suggesting new guidance for users who initially fail to comply with a policy.

Password-composition policies differed in the difficulty users had complying (Table 5.8). While 3class12 proved easiest to comply with, comp8 proved most difficult in both the rate of non-compliance and number of retries required.

Counter to the goal of password-composition policies, complying with a policy sometimes resulted in passwords that were easier to guess. We compared the guessability for the first (non-compliant) attempt and the final (compliant) password. In all sets, more passwords became harder to guess than easier to guess after complying with the given composition policy, yet 20.5% of blacklist passwords became easier to guess. Some users whose candidate password did not comply switched to easily guessable passwords containing only digits to avoid the blacklist's dictionary check. Some users' initial and final passwords were both past

Table 5.10: The percentage of times each character class was added (rows) when a specific action was suggested (columns) during pauses and non-pauses during password creation.

Character	Total	<i>Suggestion to add:</i>			
		Length	Symbol	Digit	Letter
Pause					
Symbol	11.6	2.9	22.2	10.4	10.0
Digit	35.7	20.9	40.0	54.8	35.4
Letter	52.7	76.2	37.4	34.7	54.6
Non-pause					
Symbol	4.0	2.7	4.3	3.8	4.6
Digit	22.7	12.1	29.7	10.0	30.9
Letter	73.3	85.2	66.0	86.1	64.5

the guessing cutoff. The exact security impact for these users is unknown, although both the initial and final passwords are hard to guess.

We next focused on users' modification strategies from attempt to attempt. We measure the prevalence of several simple strategies we hypothesized would be common: adding characters to the password, deleting characters, changing the capitalization of one or more letters, or making character substitutions. We also measured combinations of two strategies. If none of these strategies was observed, we classified the change as either a small alteration (*similar*) or a completely new password (*dissimilar*) using an edit distance at least two-thirds the length of the former password as the dividing line.

As shown in Table 5.9, participants employed a panoply of strategies, though some strategies were more secure than others. The two most frequent strategies were appending to the end of the prior candidate and changing capitalization. Both strategies usually increased security. In contrast, deleting characters from the middle of a password or transposing adjacent characters more often made passwords easier to guess than harder to guess. These strategies should be discouraged.

Creating a dissimilar password was among the most common strategies, yet frequently led to weaker passwords. The rate at which participants created dissimilar passwords varied sharply between sets. Users in comp8 and blacklist tended toward smaller modifications, while those in basic20 tended toward entirely new passwords. Unfortunately, 23% of dissimilar modifications made the password easier to guess, suggesting that helping users improve a password without starting over may be beneficial for security.

Receptiveness to Specific Suggestions

Rather than waiting until the submission of a non-compliant password to give users guidance, one can make suggestions while a user is creating a password. Our final analysis examines the impact of providing specific suggestions about character-class usage (e.g., "Consider adding a symbol") during password creation. We find that users often follow these suggestions during pauses in password creation.

Users in the meterStandard and meterStringent sets saw such suggestions, but we did not see a correlation between the suggestion and the user's action for most keystrokes. However, when the user paused during password creation, he or she often followed the specific suggestion (Table 5.10). As described in the methodology, we identified pauses based on the interkeystroke timing; 38% of users in these sets paused at

least once. Directly following pauses, participants added a symbol or digit to their password at a far higher rate when given that particular suggestion. This result demonstrates the effectiveness of suggestions during pauses in password creation.

5.5 Design Recommendations and Conclusions

We have used novel methods to delve into the semantics, structures, and strategies from which users craft passwords. Leveraging reverse-engineered passwords and rich metadata, we uncovered how users substitute characters and integrate semantically significant content into passwords, from what semantic sources users commonly draw inspiration, and the strategies users adopt step-by-step in creating passwords.

Users are asked to create passwords that are both memorable and hard to guess without much information or context on how to perform this job. Our findings suggest improvements to both the advice given to users and the algorithmic mechanisms for proactively preventing bad passwords.

Our first design recommendations center on improving the advice given to users. We found that users tend to make passwords about things they like (e.g., food, celebrities, friends and family, animals) or things that are on their mind while creating a password, such as words related to the website or objects near them—echoing an observation made with graphical passwords [88]. These predictable choices should be discouraged. Furthermore, we uncovered many common phrases in passwords. While users should be lauded for including more than a single word, they should be guided to choose multiple, unrelated concepts as password building blocks. In addition, our analysis of user modifications to comply with composition policies suggests that encouraging small, judicious modifications would be prudent. Absent this advice, many users start from scratch and frequently end up making the password easier to guess.

We direct our remaining recommendations toward researchers and engineers who design password-creation infrastructure. Based on our finding that users are receptive to specific suggestions during password creation, password-creation pages should include suggestions that guide users towards choices that are hard for an attacker to predict. Komanduri et al. recently introduced a system that displays its guess of what a user will type next during password creation to discourage the user from continuing along that path [107]. Rather than highlighting insecure behavior and hoping the user knows how to proceed, we propose directly guiding users towards a randomly chosen secure behavior. We encourage future investigation of what suggestions most help users to make secure choices.

Our results also suggest improvements to proactive password-checking mechanisms, which often reject potentially insecure passwords based on heuristics and lists of banned passwords [182]. We suggest that these mechanisms use our ground-truth data on character substitutions to attempt to reverse the most common substitutions to also identify weak passwords in disguise. Further, based on our finding that the individual words used in passwords and their frequencies are similar across password sets yet quite different than English, it would be prudent for researchers to establish a “password lexicon,” or frequency-ordered list of words that commonly appear in passwords. While lists of common passwords are available [77, 124, 146, 187], a lexicon could aid in identifying component parts of insecure passwords. We imagine this lexicon would be used alongside lists of common words and phrases to proactively identify predictable components.

The insights from the analyses described in this chapter form the foundation for the advanced heuristic scoring of the password-strength meter I describe in Chapter 7. In particular, the insights based on how users perform character substitutions, the source of linguistic content (single words and phrases), and the predictability of certain password structures are all deeply embedded in our improved meter’s scoring.

Chapter 6

Do Users' Perceptions of Password Security Match Reality?

6.1 Introduction

While the predictability of user-chosen passwords has been widely documented [22,94,126,187,188,191,196], very little research has investigated users' perceptions of password security. That is, do users realize they are selecting terrible passwords and choose to do so intentionally, or are they unwittingly creating weak passwords when they believe they are making secure ones? Understanding the extent to which users' perceptions of security differ from actual security enables researchers to focus on correcting security misconceptions, steps we take in the data-driven password-strength meter I present in Chapter 7.

We conducted a 165-participant study of users' perceptions of password security. Participants provided their perceptions about the security and memorability of passwords chosen to exhibit particular characteristics, as well as common strategies for password creation and management. We compare participants' perceptions to the passwords' actual resilience to a variety of large-scale password-guessing attacks.

In the first of four tasks, we showed participants 25 pairs of passwords differing in specific characteristics (e.g., appending a digit, as opposed to a letter, to the end of the password). We asked participants to rate which password was more secure, if any, and to justify their rating in free text. In the second and third tasks, we showed participants a selection of passwords from the well-studied breach of the website RockYou [187], as well as descriptions of common password-creation strategies. We asked participants to rate both the security and the memorability of each password or strategy. In the fourth task, we had participants articulate their model of password attackers and their expectations for how attackers try to guess passwords.

We observed some serious misconceptions about password security. Many participants overestimated the benefits of including digits, as opposed to other characters, in a password. Many participants also underestimated the poor security properties of building a password around common keyboard patterns and common phrases. In most other cases, however, participants' perceptions of what characteristics make a password more secure matched the performance of today's password-cracking tools. This result calls into question why users often fail to follow their (correct) understanding when crafting passwords. However,

Previously published as Blase Ur, Jonathan Bees, Sean M. Segreti, Lujjo Bauer, Nicolas Christin, Lorrie Faith Cranor. Do Users' Perceptions of Password Security Match Reality? In Proc. CHI, 2016.

In your opinion, which of the following passwords is more secure?

punk4life **punkforlife**

punk4life is much more secure	punk4life is more secure	punk4life is slightly more secure	Both passwords are equally secure	punkforlife is slightly more secure	punkforlife is more secure	punkforlife is much more secure
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Why? *

Figure 6.1: Example *password pair* comparison testing the hypothesis that substituting a digit for multiple letters will be perceived as more secure.

most participants displayed an unrealistic mental model of attackers, which may prevent them from fully accounting for the actual spectrum of threats to their passwords.

Although much has been written about text passwords in recent years, our study is the first to focus specifically on users' perceptions of security. The main outcome of our work is to inform design directions for helping users both make stronger passwords and better understand the implications of the decisions they make when creating a password.

6.2 Methodology

We conducted an online study to gauge users' perceptions of password strength and memorability, as well as their understanding of attacker models. We then compared these perceptions to passwords' resistance to current large-scale attacks.

6.2.1 Study Structure

We structured the study in five parts designed to take 30 minutes total. The first part of the study asked about demographics, including age and gender. Because participants' perceptions would likely be influenced by their technical understanding of the password ecosystem, we also asked whether they were a "security professional or a student studying computer security," and whether they had a job or degree in a field related to computer science or technology.

In the second part of the study, which we term *password pairs*, we investigated 25 hypotheses about how different password characteristics impact perceptions of security. As shown in Figure 6.1, a participant saw two similar passwords that varied in a way dictated by the hypothesis. The participant rated the passwords on a 7-point, labeled scale denoting which password is more secure. In addition, we required a free-response justification for the rating.

We chose the 25 hypotheses (see Table 6.4 in the results section), to investigate eight broad categories of password characteristics inspired by prior work [29, 102, 184, 188, 189]: capitalization; the location of digits and symbols; the strength of letters vs. digits vs. symbols; the choice of words and phrases; the choice of digits; keyboard patterns; the use of personal information; and character substitutions. As an attention check,

Please rate the security of the following password: rolltide1 *						
1 (very insecure)	2	3	4	5	6	7 (very secure)
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Please rate the memorability of the following password: rolltide1 *						
1 (very hard to remember)	2	3	4	5	6	7 (very easy to remember)
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure 6.2: An example task for rating the security and memorability in our *selected-password analysis*.

a 26th pair compared a password to itself. We randomized the order of the 26 pairs and left-right orientations of each pair per participant.

To reduce potential selection biases, we created three pairs of passwords for each of the 25 hypotheses. Each participant saw one of the three pairs, randomly selected. To create each pair, we first chose a password from the widely studied [118, 126, 195] dataset of 32 million passwords leaked in plaintext from the gaming website RockYou in 2009 [187]. In particular, we randomly permuted this set and selected the first password that could plausibly be tested as part of each hypothesis. Thus, at least one password in each pair is from the RockYou breach. For the second password in each pair, we either created the minimally different password to test the hypothesis (e.g., we created "astley123" to correspond to RockYou's "astleyabc") or selected a second RockYou password, as appropriate.

In the third part of the study, *selected-password analysis*, we investigated broader perceptions by asking participants to rate their opinion of the security and memorability of 20 passwords selected from the RockYou set [187]. As detailed below, we selected new passwords for each participant without replacement. As shown in Figure 6.2, participants used a 7-point scale to rate the security ("very insecure" to "very secure") and memorability ("very hard to remember" to "very easy to remember"); we labeled only the endpoints of the scale. We biased the selection of the passwords shown to each participant to include diverse characteristics. Ten passwords were selected randomly from among RockYou passwords matching each of the ten most common character-class structures. In addition, we selected one password containing at least three character classes, one password containing a symbol, two long passwords (12+ characters), and six additional passwords that do not fit any of the previous categories. We randomized the order in which we showed the passwords.

The fourth part of the study was similar to the third, except we instead asked about 11 strategies for password creation and password management. We chose common strategies from prior work on password creation [29, 115, 184, 188] and password management [49, 74, 169]. For example, one strategy we presented was creating a password "using a phrase taken from the lyrics to a song (e.g., somewhere over the rainbow)." We provide the full list of 11 strategies in the results section.

The fifth and final part of the study focused on participants' impressions and understanding of attackers who might try to guess their password. We intentionally presented this part of the study last to avoid priming participants as they evaluated password security in the rest of the study.

We asked seven questions about attackers. Participants wrote free-text responses to separate questions about "what characteristics make a password {easy, hard} for an attacker to guess." Participants "describe[d] the type of attacker (or multiple types of attackers), if any, whom you worry might try to guess your password,"

and explained to the best of their knowledge *why* an attacker would try to guess their password, as well as *how* attackers try to do so. Finally, participants provided a numerical estimate of “how many guesses (by an attacker) would a password need to be able to withstand for you to consider it secure,” as well as a free-text justification for why they chose that number.

6.2.2 Recruitment

We recruited participants on Amazon’s Mechanical Turk (MTurk) platform for a “research study about password security.” While imperfect, MTurk can provide data of at least the same quality as methods traditionally used in research as long as the experiment is designed carefully [17, 33]. We limited participation in this study to MTurk users age 18 and older who live in the United States. We compensated participants \$5 U.S. for the study, which took approximately 30 minutes.

To ensure quality MTurk data [100], we inserted the attention check described above. We only accepted data from participants who rated that pair as equal in strength and wrote a variant of “the passwords are the same” in their justification.

6.2.3 Measuring Real-World Attacks on Passwords

To understand how users’ perceptions of password security correspond to actual threats, we calculate each password’s guessability [22, 23, 104] by simulating attacks using modern password-cracking techniques. We use the Password Guessability Service (PGS) [37, 185], a product of our group’s prior evaluations of metrics for calculating password strength [104, 185].

In prior work, we showed that considering only one of the numerous approaches to password cracking can vastly underestimate the guessability of passwords with particular characteristics, while using a number of well-configured approaches in parallel can conservatively estimate password guessability against an expert attacker [185]. Thus, PGS simulates password-guessing attacks using Markov models [123], a probabilistic context-free grammar [104, 106, 196], and the software tools oclHashcat [166] and John the Ripper [139]. For each password, PGS conservatively outputs the smallest guess number across these four major password-cracking approaches. Evaluating guessability using several password-cracking approaches in parallel helps account for passwords that are modeled particularly well by some approaches, but not by others.

These approaches order their guesses based on training data, comprising sets of leaked passwords and natural-language dictionaries [37]. Furthermore, we configured the software tools to reflect behaviors common in the password-cracking community [185]. Thus, within the limitations of the training data and theoretical models of how humans craft passwords, the ordering of guesses is grounded in data. If the guess numbers are within an order of magnitude of each other, we judge the passwords to be of similar security. When we judge passwords to be of different security, their guess numbers differ by over an order of magnitude. These differences occur when some words or characteristics are far more common than others in the sets of real passwords used to train the tools.

In the results section, we frequently compare participants’ perceptions of the relative security of passwords to the relative difference in guess numbers. Because the PGS guess numbers reflect the performance of current password-cracking approaches, we either state that participants’ perceptions were *consistent* or *inconsistent* with current approaches.

6.2.4 Quantitative Analysis

We used different statistical tests for our quantitative analyses investigating, respectively, participants' strength ratings, the relationship between security and memorability, and the relationship between independent variables. For all tests, we set $\alpha = .05$. We corrected for multiple testing using the conservative Bonferroni method, which we applied per type of test (e.g., we multiplied p values by 75 for the 75 password pairs).

We treated participants' rating for each password pair $\{PW_1, PW_2\}$ as an ordinal rating from -3 to 3, where -3 indicates the perception that PW_1 is much stronger and 0 indicates that the passwords are equally strong. To test whether participants tended to rate one password in the pair as stronger than the other, we used the one-sample, two-sided Wilcoxon Signed-Rank test. This non-parametric test evaluates the null hypothesis that the true password rating is 0 (equally secure) and the alternative hypothesis that the true rating is non-zero (one password is perceived more secure than the other).

To investigate the relationship between security and memorability for the selected-password analysis and password-creation strategies, we calculated Spearman's rank correlation coefficient (Spearman's ρ), which is a nonparametric evaluation of the correlation between variables. The value for ρ varies between 1 (perfect correlation) and -1 (perfect inverse correlation), where 0 indicates no correlation.

For our selected-password analysis, we also used regression models to evaluate the relationship between numerous independent variables (e.g., password length, number of digits) and participants' ratings of password security and memorability. In particular, because participants' ratings were ordinal on a 7-point scale and because each participant rated 20 different passwords, we use a mixed-model ordinal regression.

6.2.5 Qualitative Analysis

We also used qualitative methods to better understand participants' free-text responses. In particular, we coded responses to the seven questions about attacker models, as well as all password pairs where participants' perceptions differed statistically significantly from the guess numbers we calculated. One member of the research team first read through all responses to a question and proposed codes that would capture common themes. This researcher then coded all responses and updated the codebook when necessary. A second coder used the annotated codebook to independently code the data. Inter-coder agreement ranged from 85.0% to 91.4% per question, while Krippendorff's α ranged from 0.80 to 0.88. The coders met, discussed discrepancies, and agreed on the final codes.

In presenting our results, we report counts of how many participants wrote responses exhibiting particular themes to comprehensively summarize our data, not to suggest statistical significance or generalizability of the proportions we report.

6.2.6 Limitations

The generalizability of our study is limited due to our use of an online convenience sample that is not representative of any larger population. Password practices are impacted by an individual's technical skills [170], and the MTurk population is younger and more technical than the overall U.S. population [147]. This skew may be exacerbated by the self-selection biases of workers who would select a study on password security. However, very few of our participants displayed any sophisticated understanding of potential or likely threats to the security of their passwords.

The security of a password, both in actuality and in perception, depends on far more factors [59, 76] than one could test in a single study. These factors include expectations about potential attackers, how the user values the account [70, 136], the user's demographics [22, 118, 126], and how well the training data used to guess passwords matches the target population and the individual [118, 123]. Some of these factors require a very large set of user-chosen passwords to analyze accurately [23]. Furthermore, the types and number of guesses an attacker might make against a particular password are influenced by the value of the information the password protects and either the hash function used or the rate-limiting employed.

While the Password Guessability Service we use reflects the performance of current password-cracking approaches and has been shown to model a skilled attacker [185], no model is perfect. A new algorithm or unexplored source of training data could vastly improve cracking and impact the ordering of guesses, changing what features make a password secure.

6.3 Results

We first briefly describe our participants. To contextualize their other answers, we then report on participants' impressions of attackers and password threats. Note that we asked these questions about attackers last in the actual study to avoid priming participants. We then present participants' perceptions of the password pairs, followed by perceptions of both security and memorability for selected passwords and strategies.

6.3.1 Participants

A total of 165 individuals participated in our study. Our sample was nearly gender-balanced; 49% of participants identified as female, and 51% as male. Participants hailed from 33 U.S. states. They ranged in age from 18–66, with a mean age of 34.2 years and median of 33. All participants correctly answered the attention-check question.

Few participants had special familiarity with computer security. Only six participants (4%) responded that they were a professional or student in computer security. In addition, only 26 participants (16%) said they had held a job or received a degree in “computer science or any related technology field.”

6.3.2 Attacker Model

Because any analysis of perceived or actual password security depends on the threat, we investigated *whom*, if anyone, participants expected might try to guess their passwords and *why* such people might do so. We also investigated participants' understanding of *how* attackers guess passwords and expectation for how many guesses an attacker might make.

Who Tries to Guess Passwords

Actual threats to passwords include both familiar people, who might attempt to access the account of a friend or family member, and strangers conducting large-scale attacks on passwords. Most participants' expectations for *who* might try to guess their password centered on some combination of these two types (Table 6.1). Overall, 135 participants (82%) mentioned a stranger of any sort as a possible attacker, and 38 participants (23%) mentioned someone they know as a possible attacker.

Type of Attacker	#	%
Stranger	135	82%
Financially motivated	88	53%
Hackers	66	40%
Other strangers	14	8%
Government	3	2%
Familiar person	38	23%
People I know (generic)	23	14%
Family	9	5%
Friend	9	5%
Coworker	3	2%
No one	8	5%

Table 6.1: Themes describing “the type of attacker (or multiple types of attackers), if any, whom you worry might try to guess your password.” # is the number of participants who mentioned the theme. The bolded categories represent participants who mentioned at least one sub-theme.

Participants generally expected strangers to be both unfamiliar and geographically far away. For instance, P62 feared “someone on the other side of world who compromises all my accounts.” Hackers were specifically mentioned by 66 participants (40%). Most of these participants discussed hackers abstractly; only one (P30, who has a technical background) expressed detailed knowledge of attacks. He wrote, “I mainly worry about large scale attacks....If my password used personal information like my telephone number...it might not be that detrimental because the attackers aren’t going to do a search for personal information on each individual.”

In contrast, many other participants anticipated that attackers who were strangers would have access to their personal information. For instance, P126 worried about “a stranger that has gotten hold of the names and birthdays of my family and pets,” while P164 worried about people who have “hacked into businesses and gotten personal information, like my name, account numbers, my birth date.”

Other expected attackers were familiar; 38 participants (23%) mentioned worrying about attacks from someone they know, such as “an angry ex or friend” (P98). Only 23 participants (14%), however, listed both strangers and familiar people. P111 was one of these participants, listing both “cyber-thieves and nosy friends or family members.”

Eight participants (5%) did not expect anyone would try to guess their password, most frequently because they did not think they had anything an attacker would want. For example, P20 did not “worry too much about people trying to guess my password as I am an insignificant person.”

Why Attackers Guess Passwords

There are a litany of reasons attackers might try to guess passwords, ranging from the hope of selling credentials on the black market to pride within the hacker community. When we explicitly asked *why* someone might try to guess their password, participants most frequently mentioned financial motivations and the theft of personal information (Table 6.2).

Of the 165 participants, 109 (66%) specifically mentioned financial motivations why attackers would try to guess a user’s password. For instance, P3 and P30 mentioned “credit card” and “banking information” as objectives. Thirty-three participants (20%) specifically mentioned identity theft.

Motivation	#	%
Financial payoff	109	66%
Gather personal information	67	41%
Identity theft	33	20%
Fun / prove they can	10	6%
Spamming	6	4%
Spying	4	2%

Table 6.2: Themes' frequency of occurrence in participants' responses to "why would an attacker try to guess your password, if at all?"

Next most commonly, participants listed the theft of personal information (67 participants, 41%). For instance, P146 worried attackers "might try to hack into my email account so they can find more personal information about me." P19 articulated both financial and personal reasons, expecting attackers would try "to find something embarrassing" or use personal information to "impersonate me, or get my money."

Ten participants mentioned motivations related to attackers having fun or proving their skills. P105 articulated this motivation as, "To cause chaos, to say they did, because they can," while P128 described such attacks as "for their sick chuckles." Six participants (4%) mentioned either email spam or social media spam, while another four participants (2%) mentioned spying, including for "state intelligence purposes" (P11).

How Attackers Try to Guess Passwords

We also asked, "As far as you know, how do attackers try to guess your password?" As detailed in Table 6.3, participants most commonly mentioned large-scale, automated guessing attacks (121 participants, 73%) or attacks targeted to the particular user (72 participants, 44%). In reality, both types of attacks occur. While 146 participants (88%) mentioned at least one of these types, only 47 participants (28%) mentioned both.

Most, but not all, participants (121 participants, 73%) anticipated that passwords might be subjected to large-scale guessing attacks. Nearly half of participants (79, 48%) specifically mentioned that they expected attackers to use software or other algorithmic techniques for large-scale guessing. P3 explained, "They use software designed to hack passwords. I've read about it." Similarly, P48 anticipated "some kind of script that runs down through password combinations automatically."

Attackers often use lists of leaked passwords and dictionaries of words and phrases as a starting point [79, 185]. Participants expected that large-scale guessing would first prioritize, in P31's words, "common things. People are fairly uncreative." For instance, P120 thought attackers would try "common names and numbers first and then work from there like maybe what people like." P157 expected attackers would "first [try] a dictionary of common words" before proceeding to try all "combinations of letters & numbers." While 42 participants (26%) used the phrase "brute force," some meant trying every possible combination, while others meant trying many possibilities (e.g., P10's "brute forcing the dictionary").

In contrast to large-scale guessing, 72 participants (44%) expected that password-guessing attacks could be targeted specifically to them by "using information that they already know about me" (P29), or if an attacker were to "scrape [my] personal details from social media" (P32). Participants expected that attackers might use information including "my likes, hobbies, music" (P58), "important dates" (P87), "favorite places" (P119) and "family members' names or birthdates" (P61).

Some of the 47 participants who mentioned both large-scale guessing and targeted attacks spoke of them

Guessing Method	#	%
Automated, large-scale	121	73%
Software / algorithms	79	48%
“Brute force”	42	26%
Dictionaries / words	27	16%
Common passwords	26	16%
Common names	8	5%
Try guessing dates	7	4%
Targeted to user	72	44%
Use personalized information	62	38%
Social engineering	7	4%
Manual guessing	4	2%
Other means	22	13%
Hacking into system / database	12	7%
Keyloggers	10	6%
Phishing	5	3%

Table 6.3: Themes’ frequency of occurrence in participants’ responses to “*how do attackers try to guess your password?*”

as separate attacks, while others expected the techniques to be used in tandem. P162 exemplified those who discussed them separately, describing that attackers would guess passwords “if they know personal information about you or use hacker software to decipher passwords.” In contrast, P80 wrote, “I think they look for weak passwords that are commonly used and narrow their guesses with any personal information that they have.”

Estimating Numbers of Adversarial Guesses

To understand participants’ security calculus, we asked, “How many guesses (by an attacker) would a password need to be able to withstand for you to consider it secure?” We required a numerical estimate (neither words nor exponential notation were permitted) and free-text justification. If stored following best practices, a password that can withstand 10^6 and 10^{14} guesses would likely be safe from online and offline attacks, respectively [69]. For passwords stored unsalted using a fast hash function (e.g., MD5), 10^{20} guesses is plausible [78, 171].

Participants’ responses ranged very widely, from considering a password secure if it can withstand 2 guesses to estimating a secure password should be able to withstand 10^{59} guesses. We observed three main categories of estimates; 34% of participants wrote a number of guesses that was 50 or smaller, 67% of participants wrote a number of guesses that was 50,000 or smaller, and only seven participants (4%) wrote at least 10^{14} guesses, which required that they type 14 or more zeros.

These three categories map to three streams of reasoning. The first stream focused on online attacks. In total, 27 participants (16%) specifically noted lock-out policies, in which a server blocks guessing after a few incorrect attempts. P12 explained, “Most secure sites cut you off after 3 or 4 guesses,” while P67 chose 20 guesses because “if the authentication mechanism hasn’t shut down attempts by this point, I’m more worried about the platform than my password.”

The second stream of reasoning centered on an attacker “giving up.” In total, 42 participants (25%)

explicitly mentioned that an attacker would give up, yet the number of guesses they estimated it would take varied widely. Some participants expected an attacker to get frustrated after dozens of guesses. For example, P150 wrote, "I feel like by the 10th [guess] they'd give up." Other participants chose far larger numbers. For instance, P104 chose 150,000 guesses because "hackers have short attention spans...Hopefully if by that many guesses they haven't gotten it they are on to something else." Ten other participants (6%) wrote that attackers would move on to other users with even weaker passwords than them, implicitly giving up. P4 chose 1,000 guesses, explaining, "I feel as though it wouldn't be efficient to continue attempting beyond that point, even if the process were automated. There are so many more potential victims whose passwords might be more obvious."

The third stream of reasoning involved participants estimating a strong attacker's computational resources. The magnitude that constituted a very large number varied widely. For example, P3 chose 1 million guesses, explaining, "I've read that hackers use sophisticated software that can bombard a computer or website with thousands of 'guesses' a minute." P78 also chose 1 million guesses because passwords "should be able to withstand a pretty extensive 'brute force' attack." Other participants chose far larger numbers. P103 chose 10^{14} because it "seemed like a high enough number to make it impossible or take more than 50 years."

As evidenced by the wide variance in estimates, many participants were uncertain of the scale of guessing attacks. P38 wrote, "I really wanted to write 'infinite.' I didn't know how to quantify this because I don't know how many guesses hackers typically take." She settled on 1,000 guesses as a proxy for "infinite." Many others made very low estimates. For example, P88 wrote, "A dozen guesses would mean they tried every obvious password and hopefully move on after that point." In contrast, P127 chose 10^{12} because "that's basically the highest number I can think of short of infinity," yet represents under three seconds of guessing in an offline attack against MD5-hashed passwords [166, 171]. Expertise did not reduce this variance. Of the seven participants who wrote 10^{14} guesses or higher, corresponding to an offline attack [69], only one held a degree or job in computer science. Similarly, guess estimates from the six participants who reported computer security expertise ranged from 3 guesses to 500 million.

6.3.3 Password Pairs

In the password pairs portion of the study, participants rated the relative security of careful juxtapositions of two passwords. As shown in Table 6.4, participants' perceptions matched many, but not all, of our 25 hypotheses of their perceptions.

Beyond matching our hypothesized perceptions, participants' perceptions were frequently consistent with the passwords' relative guessability. Of the 75 pairs of passwords (25 hypotheses \times 3 pairs each), participants' perceptions of the relative security of 59 pairs (79%) were consistent with the performance of current password-cracking approaches. In short, participants realized the following behaviors are beneficial to security:

- capitalizing the middle of words, rather than the beginning
- putting digits and symbols in the middle of the password, as opposed to the end
- using random-seeming digit sequences, rather than years or obvious sequences
- using symbols in place of digits
- preferring dictionary words over common first names
- avoiding personal content (e.g., a relative's name)
- avoiding terms related to the account (e.g., "survey" for an MTurk password)

Table 6.4: The 25 hypotheses we investigated among password pairs. The # column indicates for how many of the three pairs per hypothesis participants' perceptions matched the hypothesized perception.

Hypothesized user perception	#
<i>Capitalization</i>	
Non-standard capitalization more secure than capitalizing first letters	3
<i>The use of letters vs. digits vs. symbols</i>	
Appending a lowercase letter more secure than appending a digit	2
Appending lowercase letters more secure than appending digits	0
Symbol more secure than corresponding digit (e.g., “!” vs. “1”)	3
All-symbol password more secure than all-digit password	3
Adding an exclamation point makes a password more secure	3
Appending 1 makes a password more secure	3
Appending 1! makes a password more secure	3
<i>Location of digits and symbols</i>	
Digit in middle of password more secure than at beginning or end	3
Symbol in middle of password more secure than at beginning or end	2
<i>Choice of digits and symbols</i>	
Appending random digits more secure than appending a recent year	3
Random digits more secure than common sequence (e.g., “123”)	3
<i>Choice of words and phrases</i>	
Dictionary word more secure than a person's name	3
Word that is hard to spell more secure than easy-to-spell word/phrase	1
Uncommon phrase more secure than common phrase	0
Phrase more secure than single word	2
<i>Targeted and personal information</i>	
Unrelated word more secure than word related to the account	3
Unrelated name more secure than name of friend/family	3
Unrelated date more secure than birthdate of friend/family	3
<i>Keyboard patterns</i>	
Common keyboard pattern more secure than common phrase	1
Password without obvious pattern more secure than keyboard pattern	2
<i>Character substitutions</i>	
Lowercase letter less secure than number/symbol (e.g., “3” for “e”)	3
Uppercase letter less secure than number/symbol (e.g., “3” for “E”)	1
Relevant digit (e.g., “punk4life”) less secure than unrelated digit	3
Relevant digit (e.g., “4”) less secure than full word (e.g., “for”)	0

Their free-text responses supported their numerical ratings. Participants preferred “random capitization of letters rather than capitalizing each word” (P8). They knew “the use of people's names is more common” (P12) and that “everyone puts the numbers at the end, moving them to a different spot helps” (P66). They knew they should not use words associated with their account; P165 correctly noted, “Surveys are popular on mturk and one password is associated with that.” In addition, they knew “people use years in passwords (birthdays, anniversaries, etc.) often, so they are easier to guess” (P163).

PW ₁	PW ₂	Actually Stronger	Perceived Stronger	p	Perceptions
p@ssw0rd	pAsswOrd	PW ₂ (4×10^3)	PW ₁	<.001	
punk4life	punkforlife	PW ₂ (1×10^3)	PW ₁	<.001	
1qaz2wsx3edc	thefirstkiss	PW ₂ (3×10^2)	PW ₁	<.001	
iloveyou88	ieatkale88	PW ₂ (4×10^9)	Neither	—	
astley123	astleyabc	PW ₂ (9×10^5)	Neither	—	
jonny1421	jonnyrtxe	PW ₂ (7×10^5)	Neither	—	
brooklyn16	brooklynqy	PW ₂ (3×10^5)	Neither	—	
abc123def789	293070844005	PW ₂ (8×10^2)	Neither	—	
puppydog3	puppydogv	PW ₂ (7×10^2)	Neither	—	
qwertyuiop	bradybunch	PW ₂ (4×10^2)	Neither	—	
bluewater	nightgown	PW ₂ (3×10^1)	Neither	—	
iloveliverpool	questionnaires	PW ₂ (2×10^1)	Neither	—	
L0vemetal	Lovemetal	Neither	PW ₁	<.001	
sk8erboy	skaterboy	Neither	PW ₁	<.001	
badboys234	badboys833	Neither	PW ₂	.001	
jackie1234	soccer1234	Neither	PW ₂	.034	

■ PW₁ much more secure
■ PW₁ more secure
■ PW₁ slightly more secure
■ Equally secure
■ PW₂ slightly more secure
■ PW₂ more secure
■ PW₂ much more secure

Table 6.5: Pairs of passwords for which participants' perceptions of the relative security of the passwords differed from actual security. The number in parentheses indicates how many times stronger PW₂ was than PW₁ (ratio of guess numbers).

Participants also correctly recognized that users rarely include symbols in their passwords, rating passwords with symbols higher than those with digits even though we always replaced a digit with the symbol that shares its key on the keyboard. As P16 explained, “The ^ symbol is slightly more obscure than the 6, although it’s the same key on the keyboard.” They also realized that “obscure words” (P4) would be less likely to be guessed, particularly when the words were uncommon enough for P106 to incorrectly assert, “Moldovan is more secure because it’s a made up word.”

In contrast, the 16 pairs for which users' perceptions were inconsistent with current password cracking reveal four main misconceptions. In Table 6.5, we list these pairs and the actual ratio between the passwords' guess numbers. We consider two passwords to be equivalent in strength if their guess numbers are within an order of magnitude of each other (i.e., the ratio is between 0.1 and 10). We also graph the distribution of users' perceptions and give the (Bonferroni-corrected) p-value from the one-sample Wilcoxon Signed-Rank Test. Significant p-values indicate that participants tended to rate one password as more secure than the other.

The first common misconception was that adding digits inherently makes a password more secure than

Characteristic	Coefficient	Pr(> z)
Length	0.144	<.001
Contains uppercase letter	0.584	.020
Contains digit	0.971	<.001
Contains symbol	1.220	.006
Interaction: Upper*Digit	0.441	.010
Interaction: Digit*Symbol	-0.613	.008

Table 6.6: Significant terms in our mixed-model, ordinal regression of how password characteristics correlate with participants’ *security* ratings.

using only letters. Participants expected passwords like *brooklyn16* and *astley123* to be more secure than *brooklynqy* and *astleyabc*, respectively. Participants felt that “a mix of numbers and letters is always more secure and harder to guess” (P23) and that using “both numbers and letters...makes it more secure (unless the numbers were a birthday, address, etc.)” (P101). Because users frequently append numbers, however, the opposite is true in current password cracking. While, as P126 wrote, “Adding numbers makes the password more complex (more potential combinations when 26 letters and 10 numbers are possible),” arguments based on combinatorics fail because attackers exploit users’ tendency to append digits to passwords [79, 139].

Participants’ misconceptions about the security of digits also influenced how they perceived passwords that substitute digits or symbols for letters. Inconsistent with password-cracking tools, which exploit users’ tendency to make predictable substitutions, participants expected passwords like *punk4life* to be more secure than *punkforlife* and *p@ssw0rd* to be more secure than *pAsswOrd*. Participants incorrectly expected that “adding a number helps a lot” (P81). Similarly, P8 underestimated the rarity of unexpected capitalization, writing that “symbols and numbers are used instead of just capitalization.”

Third, participants overestimated the security of keyboard patterns. Inconsistent with current password cracking [91], participants believed that *1qaz2wsx3edc* would be more secure than *thefirstkiss*, and that *qwertyuiop* would be more secure than *bradybunch*. The fact that *1qaz2wsx3edc* “contains [both] numbers and letters” (P54) outweighed its status as a keyboard pattern. The significance of the Brady Bunch in popular culture led participants to think it was more obvious than a keyboard pattern. P14 wrote, “Bradybunch is a dictionary type of guess which makes it more vulnerable.” These participants, and many others, failed to realize that attackers’ “dictionaries” include common strings like keyboard patterns, not just words.

Finally, participants misjudged the popularity of particular words and phrases. In our security analysis, *ieatkale88* required over a billion times as many guesses as *iloveyou88* because the string “iloveyou” is one of the most common in passwords [94]. While some participants realized that “eating kale is a lot more rare than love” (P122), most did not; participants on the whole did not perceive one as more secure than the other. For instance, P50 wrote, “I think both are the same. Both are a combination of dictionary words and are appended by numbers.” Even beyond “iloveyou,” passwords often contain professions of love [188]. Participants did not realize that the dictionary word *questionnaires* would thus be a more secure password than *iloveliverpool*. Many participants thought the latter would be “more secure because it is a phrase, whereas the other password is just one word” (P146).

Characteristic	Coefficient	Pr(> z)
Length	-0.125	<.001
Contains digit	-0.753	<.001

Table 6.7: The only two significant terms in our mixed-model, ordinal regression of how characteristics correlate with *memorability* ratings.

6.3.4 Selected-Password Analysis

For the passwords in our selected-password analysis, we ran two mixed-model, ordinal regressions where the security and memorability ratings (1–7) were each dependent variables, and the password's length and inclusion of {0,1,2+} uppercase letters, digits, and symbols were the independent variables. We included terms for interactions among character classes.

In our model of *security* ratings (Table 6.6), participants tended to rate a password as more secure if it was longer and if it included uppercase letters, digits, or symbols. More precisely, security ratings for passwords selected from the RockYou set were significantly correlated with all four main independent variables. We also observed a significant positive interaction between the inclusion of uppercase letters and digits, and a significant negative interaction between digits and symbols.

Participants' *memorability* ratings were less clear-cut (Table 6.7). Participants perceived a password as significantly less memorable if it was longer or contained digits. Note, however, that many RockYou passwords contain long, random-seeming strings of digits that contain subtle patterns or are semantically significant for speakers of other languages [118], which we hypothesize caused participants to perceive digits as particularly hard to remember. No other regression terms were significant, suggesting that factors other than length and character-class usage primarily impact perceived memorability. Unsurprisingly, participants' memorability ratings were inversely correlated with strength ratings (Spearman's $\rho = -0.678$, $p < .001$).

6.3.5 Password-Creation Strategies

Participants' perceptions of the 11 common strategies for password creation and management that we showed were generally consistent with current attacks on passwords. As shown in Table 6.8, participants realized that password reuse is wholly insecure, yet memorable. While participants believed passwords based on song lyrics or relevant dates would be memorable, they also mostly realized such passwords are insecure. In contrast, participants had divergent perceptions of the security of writing a password down. Writing passwords down was traditionally discouraged, yet has more recently been argued as a sensible coping mechanism [70, 151].

As one might expect, participants perceived a tradeoff between security and memorability; the more secure a participant rated a strategy, the less memorable he or she tended to rate it. As shown in Table 6.8, for each strategy we calculated Spearman's ρ to find the correlation between security and memorability ratings. For all ten strategies other than writing a password down, we found a negative correlation between security and memorability (ρ ranging from -0.22 to -0.47).

Some strategies balanced security and memorability more successfully. To ease comparison, we plot the mean ratings for each strategy in Figure 6.3. Creating a made-up phrase (S3) and combining languages (S5) had both security and memorability ratings with means above 4.4 on the 7-point scale. In contrast,

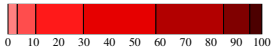
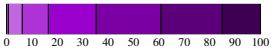
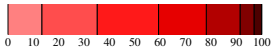
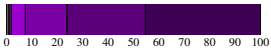




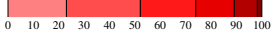
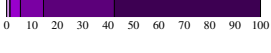
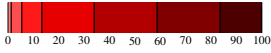
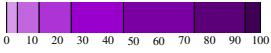


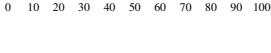
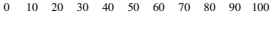
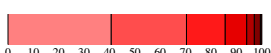
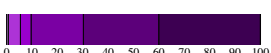




Strategy	ρ	Perceived Security	Perceived Memorability
S1: Starting with a word that comes to mind, and then adding digits or symbols to the end (e.g., bubblegum1!).	-0.22		
S2: Using a phrase taken from the lyrics to a song (e.g., somewhere over the rainbow).	-0.26		
S3: Using a phrase that you make up exclusively for this account and that has nothing to do with the account (e.g., skyscraper cornstalks).	-0.24		
S4: Using the name of one of your family members and their birth year (e.g., Zachary1976), assuming that information is not on Facebook or other social media.	-0.30		
S5: Combining words from two different languages (e.g., desaparecido rainbow).	-0.23		
S6: For your password, using a date that is meaningful to you (e.g., 12151976 because your sibling was born on 12/15/1976), assuming that information is not on Facebook or other social media.	-0.27		
S7: Basing a password on a phrase that describes your relationship to the account (e.g., iloveshoppingonamazon for your Amazon.com password).	-0.47		
S8: Building the password by following a pattern on the keyboard (e.g., 1qaz2wsx3edc).	-0.47		
S9: Using the same password that you use for other accounts.	-0.40		
S10: Using a tool that can randomly generate a complex password for you.	-0.45		
S11: Picking a complex password and writing that password down on a piece of paper that only you know about.	n.s.		

Table 6.8: Perceptions of the security and memorability of strategies. Participants rated both on a 1–7 scale, where 7 (darker colors on the graphs) indicates “very secure” and “very easy to remember,” respectively. Spearman’s ρ indicates the correlation between security and memorability ratings.

automatically generated passwords (S10) were perceived as secure, but not memorable, whereas basing passwords on the account was perceived as very memorable, yet very insecure.

6.4 Discussion

We have presented the first study comparing users’ perceptions of the security of text passwords with those passwords’ ability to withstand state-of-the-art password cracking. Because predictable passwords are ubiquitous [22, 79, 94, 187] even for important accounts [126], we were surprised to find that participants’ perceptions of what characteristics make a password more secure are, more often than not, consistent with the performance of current password-cracking approaches.

Participants did have some critical misunderstandings, however. They severely overestimated the benefit of adding digits to passwords and underestimated the predictability of keyboard patterns and common

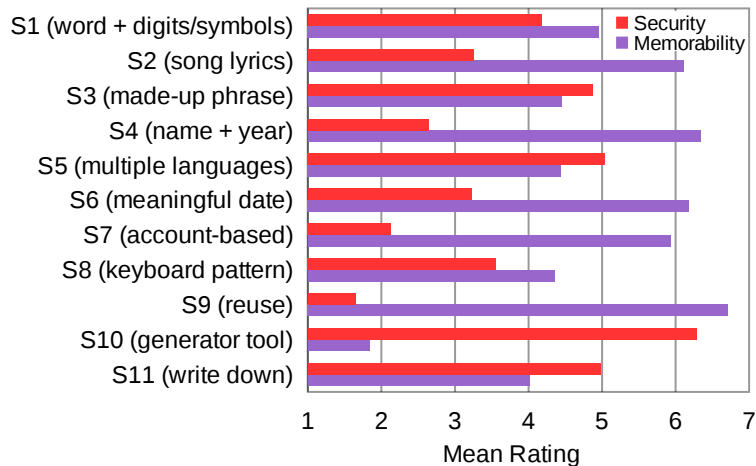


Figure 6.3: Mean ratings for the security and memorability of the 11 password-creation strategies.

phrases (e.g., “iloveyou”). In essence, participants did not realize how common these behaviors are, which is not surprising since users never see other users’ passwords. A promising direction to help users better evaluate their passwords relative to common practices is through targeted, data-driven feedback during password creation. Current password-strength meters only tell users if a password is weak or strong, not why [51, 182, 198]. Future work in this area could build on a recent study that showed users likely “autocompletions” of the partial password they had typed [107]. In large part, our results suggest that users are already aware of ways to make their passwords stronger, but they do not do so. Thus, such future work could build on research using motivational statements [62, 186] or peer pressure [65, 163] to “nudge” users [175] to create stronger passwords.

Our finding that participants mostly knew whether particular characteristics would make passwords easier or harder for attackers to guess may seem at odds with the pervasiveness of poor passwords. This gap, however, may be the result of neglecting to help users understand the spectrum of attacks against passwords. As in other studies [98, 184, 194], our participants knew passwords were important, yet their models of attackers were often incomplete. Whereas one-third of our participants considered a password secure if it can withstand as little as several dozen guesses, others believed a password must withstand quadrillions of guesses or more.

Users’ incomplete understanding of the scale of potential attacks thus seems to be a root cause of bad passwords. As we surveyed in the background section, the spectrum of threats to passwords is complex and nuanced. For instance, a password’s resistance to large-scale guessing matters mostly if the user reuses that password for other accounts [69] or if the service provider fails to follow security best practices [36, 82, 122]. Following the principle of defense in depth, users should protect themselves against all likely attackers, which is why security experts often recommend using password managers to store unique passwords for each account [98]. Unfortunately, users receive scant holistic advice on the overall password ecosystem [70, 98, 144]. No system administrators are incentivized to encourage users to make weak passwords for unimportant accounts or to write their passwords down [151]. Thus, users derive oversimplified folk models [194] and misconceptions [184]. We showed that users understand quite a bit about the characteristics of strong and weak passwords, which should be leveraged to help users create stronger passwords.

Chapter 7

Design and Evaluation of a Data-Driven Password Meter

7.1 Introduction

In spite of their ubiquity [182], password-strength meters often provide ratings of password strength that are, at best, only weakly correlated to actual password strength [51]. Furthermore, in contrast to greater trends in the usable security community to give actionable, intelligible feedback, current meters provide minimal feedback to users. They may tell a user that his or her password is “weak” or “fair” [51, 182, 199], but they do not explain what the user is doing wrong in constructing a password, nor do they guide the user towards mechanisms for doing better.

In this chapter, I describe our development and validation of an open-source password-strength meter¹ that is both more accurate at rating the strength of a password and provides more useful, actionable feedback to users. Whereas previous meters scored passwords using very basic heuristics (e.g., examining only the password’s length and whether it includes digits, symbols, and uppercase letters), we use the complementary techniques of simulating a very large number of adversarial guesses using artificial neural networks [130] and employing nearly two dozen advanced heuristics to rate password strength. As we detail in Section 7.2, these two techniques are complementary. Neural networks, particularly when used with Monte Carlo methods [53], can provide accurate password-strength estimates in real time entirely on the client side [130]. However, neural networks are effectively a black box and provide minimal human-understandable transparency for their scoring. Although advanced heuristics also provide relatively accurate password-strength estimates, at least for resistance to online attacks [199], we use them primarily to identify characteristics of the password that are associated with guessability.

Beyond scoring password strength more accurately than previous meters, our meter also gives users actionable, data-driven feedback about how to improve their specific candidate password. As we describe in Section 7.3, we provide users with up to three ways in which they could improve their password based on the characteristics the advanced heuristics identified in their particular password. We also provide modal windows

Blase Ur, Felicia Alfieri, Maung Aung, Lujo Bauer, Nicolas Christin, Jessica Colnago, Lorrie Faith Cranor, Henry Dixon, Pardis Emami Naeini, Hana Habib, Noah Johnson, William Melicher. Design and Evaluation of a Data-Driven Password Meter. Unpublished.

¹Demo available at <https://cups.cs.cmu.edu/~meter/>

that detail why these particular suggestions would improve the password and discuss general strategies for making strong passwords. Furthermore, we generate proposed modifications to the user’s password through judicious insertions, substitutions, rearrangements, and case changes. If our strength estimates indicate substantial improvements in password strength, we show the user this suggested improvement.

We developed our meter through an iterative design process involving regular meetings with the members of our research team and additional meetings with industry partners. As we detail in Section 7.4, we also conducted a ten-participant laboratory study of the meter, iteratively updating the meter’s design after each session. To measure how different aspects of our open-source meter impacted password security and usability, we subsequently conducted a 4,509-participant online study, which we report in Section 7.5. Under the more common password-composition policy we tested, we found that our data-driven meter with detailed feedback led users to create significantly more secure passwords than a meter with only a bar as a strength indicator, without a significant impact on any of our memorability metrics. In addition, most of our participants reported that the text feedback was informative and helped them create stronger passwords.

7.2 Measuring Password Strength in our Data-Driven Meter

Measuring password strength in the browser involves complicated tradeoffs and considerations. On the one hand, it is crucial for security that this rating of password strength accurately reflect the password’s resistance to plausible attacks. On the other hand, accurate estimates are not useful during password creation unless they are delivered quickly. For protecting against side-channel attacks, reducing the attack surface of a server, and reducing latency, it is desirable that these accurate and fast estimates are delivered entirely on the client side, which requires a compact model.

Below, we first detail why accurately estimating password strength on the client side is difficult. Subsequently, we describe our two-pronged approach to scoring password strength in our meter. These two approaches are highly complementary and enable us to provide accurate and actionable feedback on password strength quickly and entirely on the client side. Our use of neural networks [130] as the first prong in this approach provides principled, statistical models of password guessing, yet provides no intelligibility to humans for its ratings. The second prong in our approach, employing advanced heuristics, is both fairly accurate [199] and also provides human-intelligible explanations for its scoring.

7.2.1 The Difficulty of Accurate Client-Side Password-Strength Estimation

To evaluate a password’s resistance to a guessing attack, one can calculate a password’s *guessability*, which we define as the number of guesses that a particular password-guessing model (in a particular configuration and using particular training data) would take to guess that password. In prior work (Chapter 4), we showed that well-configured models of numerous leading password-guessing approaches considered in parallel can serve as a conservative proxy for password guessing directed by a human expert in password forensics, at least through the cutoff of 10^{14} guesses we used in many of our tests [185].

Unfortunately, these models of password guessing are often inappropriate for real-time password-strength evaluation. First of all, the models used in probabilistic password-guessing approaches, such as probabilistic context-free grammars and Markov models, are often on the order of gigabytes [60, 104, 123, 185]. Even modeling password-guessing software tools like John the Ripper and Hashcat (e.g., using HashcatJS [10], a Javascript implementation of the Hashcat rule engine) requires that a full wordlist (often hundreds of

megabytes deduplicated and compressed) be transferred to a client [185]. Overall, measuring password strength using models of password guessing requires either more time or more disk space than would be practical for performing this measurement on the client side, even when using variants of models optimized for looking up password strength [104, 185].

As a result, previous attempts to use principled models of password-guessing attacks have necessitated having a server-side component. For instance, researchers have built a password-strength meter based on adaptive Markov models [38]. While Markov models are fairly accurate at guessing passwords [60, 123], each password lookup requires a round-trip call to the server, and having a server-side component increases a system’s attack surface, configuration requirements, and the possibility of timing-based side-channel attacks. Other researchers have built Telepathwords, a meter that relies on extensive models of passwords and natural language to predict the next character a user would type, subsequently rating a password’s strength based on the number of unpredictable choices made [107]. For the same reasons as Markov models, Telepathwords’ models must be stored on a server and therefore suffer from the same disadvantages.

To this point, password-strength meters on the client side have relied on basic heuristics that can be easily encoded in Javascript, run quickly, and require minimal data to be transferred to a client. The most common heuristics include counting the number of characters in the password, identifying which character classes (digits, symbols, lowercase letters, uppercase letters) have been used, and sometimes checking the candidate password against a small blacklist of very common passwords [51, 182]. For a given password, these characteristics can be calculated quickly entirely on the client side. Unfortunately, they are only weakly correlated to actual password strength [51].

Recent innovations in password-strength estimation from both our research group and others have enabled faster and more compact, yet still accurate, estimations of password strength entirely on the client side. In our data-driven meter, we use two complementary approaches to measure password strength. The first approach, using artificial neural networks (henceforth “neural networks”) to measure the probability of a password represents recent work from our group [130]. The second approach, using advanced heuristics to estimate password strength by examining common password patterns statistically, is our own implementation, yet is conceptually inspired by the zxcvbn meter [198, 199]. Our approach models correlations between patterns commonly found in passwords and password guessability. We ran a linear regression to identify how to weight each characteristic to estimate password guessability. While neural networks model a very effective password-guessing attack [130], they are effectively a black box for explaining why they assign a particular probability to a particular password. As a result, the advanced heuristics provide complementary human-intelligible explanations for password guessability in a way we have designed to correlate well with the results of a password-guessing attack.

7.2.2 Neural Networks for Password-Strength Estimation

The first of our two methods for estimating password strength relies directly on the insights and codebase from our recent research on training artificial neural networks to guess passwords [130]. For this portion of the project, we build on the codebase developed for that project.²

Our implementation of neural networks for estimating password strength works by estimating the probability of each character of the password based on the characters that came before it. The overall probability of a password is the product of the probabilities of its characters. While this approach may

²https://github.com/cupslab/neural_network_cracking

superficially sound reminiscent of Markov models [60, 123], which also calculate the probability of each character in a password based on the characters that came before it, our neural networks approach differs in key ways. First, we train the neural network using up to the previous ten characters of context (previous characters in the password), and the neural network effectively learns how much context to consider in its training process. Thus, one aspect of the neural network is that its estimates of the probability of a character is akin to considering a Markov model of variable length. Furthermore, neural networks learn additional, higher-order features during their training.

We use recurrent neural networks, which allow connections between the layers of the network. Rather than considering only individual characters, we calculate the probabilities of characters in a password by considering 2,000 unique password tokens, which can be individual characters, syllables, or other multi-character tokens. The potential design space for configuring other aspects of neural networks to guess passwords is vast. In prior work [130], we tested a number of features of neural networks that intuition suggested would likely improve password guessing. As a result, we provide context characters in reverse order, which we found to work more effectively than providing them in the order in which they appear in the password. We account for the probabilities that a letter is capitalized, or that a rare symbol is used, outside the neural network in post-processing. We chose not to augment passwords with natural-language corpora in our training data. While the inclusion of natural-language data generally improves the accuracy of other password-guessing approaches [185, 197], we found the opposite effect for neural networks. Based on our tests, our neural network uses transference learning, which enables the network to be trained in part on passwords that do not fit the specified password-composition policy.

We tested a number of additional design dimensions that were chosen primarily to reduce the space requirements for storing the model. Techniques for reducing the disk space required for the model while minimally impacting guessing success enable us to model passwords entirely on the client side. For instance, we tested a larger neural network containing 15,700,675 parameters and a smaller network containing 682,851 parameters. While the larger network was more accurate at guessing, the smaller network was not much worse, yet far smaller. Therefore, our open-source password meter uses the smaller network. Similarly, we quantize probabilities and store numbers using fixed-point encoding and ZigZag encoding. These techniques again reduce the disk space needed for the model, yet only minimally impact guessing accuracy.

After calculating a password's probability using the neural network model, the final step is to convert that probability to an approximate number of guesses in a password-guessing attack. Making the likely assumption that an attacker would guess passwords in order of descending probability, this step reduces to estimating how many passwords have a higher probability than the candidate password. We approximate this number using Monte Carlo methods [53] and pre-compute a mapping between probabilities and estimated guess numbers.

To avoid causing perceptible lags in the meter interface, we perform all neural network calculations asynchronously in a separate thread using the WebWorker framework [133].

7.2.3 Advanced Heuristics for Password-Strength Estimation

While basic heuristics (e.g., length and character-class usage) that have traditionally been used to estimate password strength lead to poor estimates of password strength [51], the notable exception is the relatively accurate [51, 199] zxcvbn meter [198]. In contrast to traditional meters, zxcvbn rates a password by using a number of more advanced heuristics to identify common password patterns contained within the password. In

particular, it searches for the following patterns: tokens (common passwords, dictionary entries, etc.), reversed tokens (e.g., “drowssap”), sequences (e.g., “2468”), repeated sections (e.g., “princeprince”), keyboard patterns (e.g., “zxcvbn”), dates (e.g., “8.7.47”), and sections that would need to be bruteforced (no obvious pattern). By considering the space of possibilities within each of those patterns, as well as evaluating the frequency of some of the most common password patterns based on data from large password breaches, zxcvbn provides an estimate of the number of guesses required to arrive at that password.

While our meter searches for some of the same common password characteristics as zxcvbn when rating a password, we take a different approach in using these characteristics to estimate password guessability. Beyond verifying that a password has met the requirements specified by the password-composition policy, we rate the password on 21 different characteristics, each implemented in its own function. Some of these characteristics overlap conceptually with characteristics examined by zxcvbn (e.g., the use of keyboard patterns, dates, and alphabetic sequences), while other characteristics (e.g., the predictability of the character-class structure, the location of digits within the password) are unique to our approach.

Below, we first describe the different characteristics on which we rate passwords using the function names we employ in our code. Afterwards, we describe how we used a linear regression to determine how to combine the individual scores into a final score for the password.

- **verifyMinimumRequirements()** is a customizable function that evaluates whether a particular password meets a service’s password-composition policy. We currently support the eight dimensions of requirements listed below. System administrators can choose which of these dimensions to include in their minimum requirements, in addition to specifying the particulars of each included dimension.
 1. Length (minimum and optionally also a maximum)
 2. Minimum number of character classes
 3. Make the inclusion of certain character classes mandatory
 4. Forbid certain character classes
 5. Forbid particular passwords (case-insensitive)
 6. Specify the range of acceptable character codes, as well as identify any characters in that range that are forbidden
 7. Forbid a password from containing N or more consecutive identical characters
 8. Require that a password differ from the chosen username by a minimum number of characters
- **pwLength()** returns the total number of characters in the password.
- **countUC()** returns the number of uppercase letters contained in the password.
- **countLC()** returns the number of lowercase letters contained in the password.
- **countDIGS()** returns the number of digits contained in the password.
- **countSYMS()** returns the number of symbols contained in the password.
- **characterClasses()** returns the number of character classes (1–4) in the password.

- **keyboardPatterns()** returns the total number of characters of a password contained in one or more keyboard patterns. We define a keyboard pattern to be 4+ characters in a row for which the inter-key x-y coordinate change on a physical QWERTY keyboard layout is the same. For instance, “getu” would be a keyboard pattern because each inter-key coordinate change is 2 keys to the right horizontally, and no change vertically. Note that we only consider a string to be a keyboard pattern if the inter-key vector on a QWERTY keyboard is identical. While some keyboard patterns in practice could include snake-like bends, they would lead to many false positives (e.g., “reds,” “polk”) and common keyboard patterns of that type would be identified as a common password substring, so we do not look for them.
- **structurePredictable()** identifies how common the character-class structure (e.g., “6 lowercase letters, followed by 2 digits”) of the password is. It returns a number between 1 (Nth most common structure) and N (most common structure). We are currently using $N = 2,124$ structures based on our work on adaptive password-composition policies.
- **uppercasePredictable()** returns 1 (true) or 0 (false) whether the usage of uppercase characters in this password is predictable. To determine predictability, we examined capitalization patterns in the 10 million Xato passwords [34]. The two most common capitalization patterns, which are thus the ones we label as predictable, are capitalizing only the first character and using all uppercase characters.
- **digitsPredictable()** returns 1 (true) or 0 (false) whether the location of digits in this password is predictable. To determine predictability, we examined patterns in the location of digits in the 10 million Xato passwords [34]. The patterns we identified as predictable are constructing the password exclusively of digits, putting the digits in one group at the beginning of the password, and putting the digits in one group at the end of the password.
- **symbolsPredictable()** returns 1 (true) or 0 (false) whether the location of symbols in this password is predictable. To determine predictability, we examined patterns in the location of symbols in the 10 million Xato passwords [34]. The patterns we identified as predictable are putting the symbols in one group at the end of the password or constructing the password as letters-symbols-digits.
- **alphabeticSequenceCheck()** return the number of characters that are part of alphabetic sequences (e.g., “abc” or “aceg”) or numerical sequences (e.g., “123” or “1357”) that are at least 3 characters long and are defined using the difference in ASCII codes between adjacent characters. If the inter-character difference in ASCII codes is the same, the elements in that string or substring are an alphabetic sequence. If there are multiple such sequences, it returns the sum of the number of characters in each.
- **commonsubstringCheck()** returns the number of characters in the password that are common substrings in passwords. We require that these substrings contain 4–8 characters and occur at least 2,000 times each among the 10 million Xato passwords [34]. We build the list of common substrings in order of decreasing length, ignoring potential substrings that are themselves substrings already on our list. For instance, if we identify “monkey” as a common substring, will not add “monke” to the list of common substrings. In total, we identified 2,385 substrings that met these criteria.
- **combinedDictCheck()** returns three values. First, it returns the number of characters in the password contained from any of the sources listed below:

1. A list of the 234 most popular pet names
2. The 2,500 most popular male and 2,500 most popular female names according to the U.S. census
3. The top 50,000 three-word phrases used on Wikipedia [200]
4. Frequently used English words taken from the intersection of the BYU Corpus of Contemporary American English (COCA) 100,000 most frequent 1-grams [50] and the Unix dictionary
5. The 100,000 top single words (1-grams) used on Wikipedia [200]

For each list, we removed those that were internal duplicates (e.g., some common male and female names are identical, and some distinct three-word phrases appear the same after removing spaces and punctuation), and we also removed any that appeared on a list above it (following the order listed above) or was a keyboard pattern, string of a single character repeated, or alphabetic/numeric sequence.

In addition to checking for these words in a case-insensitive manner, we also evaluate whether a transformation of these words is present by reversing all instances of the 10 most common character substitutions in passwords [183]. For instance, if the user's password contains a "4," we will evaluate whether replacing that character by an "a" or "for" leads to the password containing a dictionary word. The commonness of the substitution (what percentage of all substitutions follow that particular rule, as determined in Chapter 5) is the second value returned by this function. It also returns the number of distinct dictionary tokens (e.g., a password that contains two separate dictionary words contains two tokens) as the third value.

- **commonpwCheck()** returns the length of the longest substring of the password that is itself one of the 90,116 passwords that appears at least 10 times in the Xato.net corpus of 10 million passwords [34].
- **repeats()** returns the number of characters in the longest string of at least 3+ consecutive, repeated characters (e.g., "monkeey" returns 3, while "monkey" returns 0) in the password.
- **repeatedSections()** returns the number of characters in the password that repeat, either forwards or backwards, a string of 3+ characters that appeared earlier in the password (case insensitive) . For instance, "monkey" returns 0, "monkeymonkey" and "monkeyyeknom" would each return 6, while "monkeymonkey123yeknom" would return 12.
- **identifyDates()** returns the number of characters in the password contained in a date. We use the common ways of writing dates observed by Veras et al. [189] in their investigation of the use of dates in passwords from the RockYou breach [187]. We search for dates in MM-DD-YYYY, DD-MM-YYYY, MM-DD-YY, and DD-MM-YY format using the following delimiters: space; period; hyphen; forward slash. We subsequently search for dates in MMDDYYYY and DDMMYYYY format without any delimiters. We also search for written-out months (e.g., "april") and recent years (4 digits or 2 digits). We also search for MM-DD and DD-MM dates using the following delimiters: space; period; hyphen; forward slash. Finally, we search for recent years (1900 through 2049).
- **duplicatedCharacters()** returns the total number of characters that are duplicates of characters used previously in the password. The repetition of characters does not need to be consecutive. For instance, "zcbm" contains 0 duplicated characters, "zcbmcb" contains 2 duplicated characters, and "zcbmbb" also contains 2 duplicated characters.

Table 7.1: Linear regression results. The order of magnitude of the Password Guessability Service min_auto guess number [185] (that is, $\log_{10}(\text{guess}\#)$) is the dependent variable, and the ratings of the passwords on each characteristic are the independent variables. The function countSyms() was collinear with other terms, so we dropped it from the regression. Adjusted $R^2 = 0.580$.

Characteristic	Coefficient	Std. Error	t-value	p
<i>Intercept</i>	1.530	0.182	8.406	<.001
pwLength().score	0.313	0.050	6.231	<.001
countUC().score	0.911	0.051	17.894	<.001
countLC().score	0.737	0.050	14.802	<.001
countDIGS().score	0.758	0.050	15.258	<.001
countSYMS().score	–	–	–	–
characterClasses().score	0.991	0.048	20.61	<.001
keyboardPatterns().score	-0.117	0.020	-5.920	<.001
structurePredictable().score	-0.001	<.001	-17.074	<.001
uppercasePredictable().score	-0.301	0.088	-3.407	<.001
digitsPredictable().score	-0.557	0.058	-9.661	<.001
symbolsPredictable().score	0.135	0.115	1.177	0.239
alphabeticSequenceCheck().score	-0.240	0.012	-20.07	<.001
commonsubstringCheck().score	-0.136	0.006	-22.533	<.001
combinedDictCheck().score	-0.553	0.013	-44.276	<.001
combinedDictCheck().dictionaryTokens	1.927	0.053	36.369	<.001
combinedDictCheck().substitutionCommonness	0.001	<.001	2.294	0.022
commonpwCheck().score	-0.395	0.008	-51.182	<.001
repeats().score	-0.004	0.015	-2/551	0.011
repeatedSections().score	-0.298	0.021	-14.013	<.001
identifyDates().score	-0.121	0.012	-10.528	<.001
duplicatedCharacters().score	0.046	0.015	3.005	0.003

- **contextual()** returns the password after removing the longest string of five or more contiguous characters of the password that overlap (case-insensitive) with the user’s chosen username. If there is no such overlap, the function returns the original password.
- **blacklist()** returns the password after removing all occurrences of a service-specific substring blacklist of terms very closely related to the service. The site-specific blacklist for Carnegie Mellon, for instance, might contain terms like “carnegie,” “mellon,” “cmu,” “education,” “tartans,” “andrew,” and other terms closely associated with the institution. If there is no such overlap, the function returns the original password.

After scoring a particular password on each of these different characteristics, we have 23 distinct scores, three from combinedDictCheck() and one from each of the 20 other functions. To combine these individual scores into a final score for the password in a principled way, we performed a linear regression. In particular, we scored a sample of 15,000 passwords taken from the breach of the gaming website RockYou [187] and another 15,000 passwords taken from the breach of Yahoo’s Associated Content service [77] by all of these characteristics except for contextual() and blacklist(). We did not score the passwords on these two functions because they are respectively user-specific and account-specific. The 21 scores from the 19 remaining functions were the independent variables in our regression.

The dependent variable in our regression was the guessability of a password. We calculated each password's guessability using the CMU Password Guessability Service (PGS) [37, 185], which is a product of our prior work modeling password-guessing algorithms and tools (Chapter 4). We used the PGS recommended settings, modeling a probabilistic context-free grammar [197], order-5 Markov model [123], Hashcat [166], and John the Ripper [139], with the latter two in wordlist mode. For each password under each guessing approach, PGS returns a *guess number*, which is the number of guesses that approach (in that configuration and with that training data) took to arrive at that password, or “unguessed” if it did not guess that password by its guessing cutoff (approach-dependent, but generally around 10^{14}). We similarly followed the PGS recommendation of conservatively considering a password to be guessed as soon as any of those four approaches guessed it.

Because we were most concerned with the order of magnitude of guessing, rather than the specific guess number, we took the logarithm (base 10) of the guess number and set that value as the dependent variable. We arbitrarily set the dependent variable for unguessed passwords to 15, which is beyond the order of magnitude for any password's guess number in our PGS tests. We then tried modeling the scores of our 30,000 test passwords' scores to a generalized linear regression model, as well as an ordinal model. The linear regression model was a better fit, and therefore we used the linear regression results, shown in Table 7.1. We also initially tested using *countSYMS()* as part of the regression, but it was collinear (completely correlated) with the number of character classes in the password (*characterClasses()*), so we dropped it from the regression.

The results of this regression enable us to weight the characteristics in a principled way to estimate password strength. In scoring a candidate password, we use the coefficients determined in this regression to weight the score from each characteristic. Before scoring the password, however, we remove from the candidate password the overlap between a user's chosen username and his or her chosen password (identified by the *contextual()* function) and substrings of the password contained in a service-specific blacklist of terms very closely related to the service (identified by the *blacklist()* function). In scoring a password, we thus completely discount (give no scoring credit to) all characters in the password identified by either.

7.3 Visual Design and User Experience

In this section, we describe the visual design of our meter. At a high level, the meter comprises three different screens. The *main screen* uses the visual metaphor of a bar to display the strength of the password, and it also provides detailed, data-driven feedback about how the user can improve his or her password. The main screen also contains links to the two other screens. Users who click “(Why?)” links next to the feedback about their specific password are taken to the *specific-feedback modal*, which gives more detailed feedback. Users who click the “How to make strong passwords” link or “(Why?)” links adjacent to feedback about password reuse are taken to the *generic-advice modal*, which is a static list of abstract strategies for creating strong passwords.

Below, we first explain how we synthesize the two password-strength ratings (described in Section 7.2) and map these ratings to a visual representation, specifically a bar metaphor. We then describe each of the three screens in our visual design, as well as their sub-components, in greater detail.

7.3.1 Translating Scores to a Visual Bar

Password-strength measurements are normally displayed to users not as a numerical score, but using visual metaphors like a colored bar [51, 182]. In creating our meter, we needed to decide how to map a password's scores from both the neural network and advanced heuristic methods to the amount of the bar that should be filled. This step is particularly crucial because we found in our prior work (Chapter 3) that the stringency of the rating displayed by the meter has a significant impact on the strength of passwords users create [182].

We conservatively took the lower of the neural network and advanced heuristic estimates for the number of guesses the password might withstand. So that the bar would scale with the order of magnitude of guessing resistance and subsequently show progress even for the first few characters a user typed, we then calculated the \log_{10} of this lower estimate of the number of guesses.

Our prior work has shown that most users do not endeavor to fill the bar, but rather consider a password sufficiently strong if only part of the bar is full [182]. As a result, we chose to map scores to the bar such that one-third of the bar being filled means that the candidate password would likely resist an online attack, while two-thirds of the bar being filled means that the candidate password would likely resist an offline attack, assuming that a hash function designed for password storage (see Section 2.1) is used. In our default configuration, we chose to map an estimated 10^{12} guesses to two-thirds of the bar being filled, and thus map 10^6 guesses to one-third of the bar being filled. We based these estimates on those of Florêncio et al. [69]. As we describe in Section 7.5, however, we tested three different mappings (scoring stringencies) as part of our online study of the meter.

7.3.2 Main Screen

The main screen includes fields for the username, password, and password confirmation. Similar to many prior password-strength meters [51, 65, 182], a bar below the password field fills up and changes color to indicate increasing password strength. Different from previous meters, though, our meter also displays data-driven text feedback about what aspects of the user's specific password could be improved.

As shown in Figure 7.1, the meter initially displays text noting which requirements have, and which have not, been met. Once the user begins to enter a password, the meter indicates that a particular requirement has been met by coloring that requirement's text green and displaying a check mark. It denotes unmet requirements by coloring those requirements' text red and displaying empty checkboxes. Until the user's password meets the requirements, the bar displaying the password's strength rating is always gray, as opposed to changing colors to reflect password strength.

Colored Bar Once the password meets the account's composition requirements, we display the bar in color (termed the *colored bar* in the remainder of this chapter). With increasing password-strength ratings, the bar progresses from red to orange to yellow to green. As described above, a bar that is at least one-third full represents a password that is estimated to resist an online attack, whereas a bar that is at least two-thirds full represents a password that is estimated to resist an offline attack. When it is one-third full, the bar is a dark shade of orange. At two-thirds full, it is yellow, soon to turn to green. Rather than employing a small set of discrete colors, the meter reflects small changes in password strength with small changes in color.

Text Feedback Whereas the colored bar is typical of password-strength meters [182], our meter is among the first to provide detailed, data-driven text feedback on how the user can improve his or her specific

Figure 7.1: The meter’s main screen prior to the password meeting the stated password-composition policy. The bar is gray because the password does not comply with the composition policy.

candidate password. We designed the text feedback to be directly actionable by the user, in addition to being specific to his or her password. Examples of this feedback include, as appropriate, exhortations to avoid dictionary words and keyboard patterns, encouragement to move uppercase letters away from the front of the password and digits away from the end of the password, and suggestions for including digits and symbols.

Most of the feedback comments on specific aspects of the password. Because users are likely not expecting their password would be shown on screen, particularly if they are creating their password in a public place, we designed two variants for each piece of feedback: a *public* and a *sensitive* version. The public versions of feedback mention only the general class of characteristic the password contains (e.g., “avoid using keyboard patterns”), whereas the sensitive versions also display the problematic portion of the password (e.g., “avoid using keyboard patterns like adgjl”). As shown in the top part of Figure 7.2, we display the public versions of feedback when the user is not showing his or her password on screen, which is the default behavior. We provide checkboxes with which a user can “show [their] password & detailed feedback,” at which point we use the sensitive version (as shown in the bottom part of Figure 7.2).

To avoid overwhelming the user, we show at most three pieces of feedback at a time. Each of our 21 advanced heuristic functions returns either one sentence of feedback or the empty string. If the candidate password is predictable in the way measured by the particular function, it returns the sentence of feedback. If the password is not predictable in that way, the function returns the empty string. To choose which of the potentially 21 pieces of feedback to display, we manually ordered the 21 functions to prioritize those that would display the most critical problems and would provide the most novel information to users. We list our final prioritization in Section D.1 in the appendix. Based on our iterative testing and formative laboratory study (Section 7.4), we reordered these functions numerous times.

Suggested Improvement Humans are poor sources of randomness. Because many potential improvements to a password involve making hard-to-predict modifications, we decided to augment our text feedback with

The figure displays two screenshots of a password creation interface, both titled "Create Your Password".

Top Screenshot (Hidden Password):

- Username:** blase
- Password:** Hidden with dots. A progress bar below the field is approximately 25% full (orange).
- Show Password & Detailed Feedback:**
- Feedback Panel:**
 - Header: "Your password could be better."
 - Item 1: "Don't use dictionary words" (Why?)
 - Item 2: "Capitalize a letter in the middle" (Why?)
 - Item 3: "Move symbols and digits elsewhere in your password" (Why?)
 - Buttons: "See Your Password With Our Improvements" and "How to make strong passwords"
- Continue:**

Bottom Screenshot (Shown Password):

- Username:** blase
- Password:** Examplepassword%| (with red dashed lines under "Example" and "password"). A progress bar below the field is approximately 75% full (orange).
- Show Password & Detailed Feedback:**
- Feedback Panel:**
 - Header: "Your password could be better."
 - Item 1: "Don't use dictionary words (password and Example)" (Why?)
 - Item 2: "Capitalize a letter in the middle, rather than the first character" (Why?)
 - Item 3: "Move your symbols earlier, rather than just at the end" (Why?)
 - Text: "A better choice: E?amplepasswor%d"
 - Buttons: "See Your Password With Our Improvements" and "How to make strong passwords"
- Continue:**

Figure 7.2: The meter's main screen giving feedback specific to the password the user typed. The two variants for the same password reflect when the password is hidden (top) and shown (bottom).

The screenshot shows a 'Create Your Password' form with three input fields: Username, Password, and Confirm Password. The Password field contains 'Mypassword123' and has a red progress bar below it. A 'Show Password & Detailed Feedback' checkbox is checked. A feedback box on the right lists three issues: 'Don't use dictionary words (password)', 'Capitalize a letter in the middle, rather than the first character', and 'Consider inserting digits into the middle, not just at the end'. A suggested improvement 'My123passwoRzd' is shown with '123' and 'Rzd' highlighted in magenta. A 'Continue' button is at the bottom right.

Figure 7.3: An example of suggested improvements for the password “Mypassword123.” The suggested improvement is “My123passwoRzd,” with the changes highlighted in magenta.

a *suggested improvement*. Although the text feedback is specific to a particular candidate password, its actionable requests (e.g., “capitalize a letter in the middle”) still require that the user make an unpredictable choice. Therefore we also randomly generated a suggested improvement, or concrete modification of the user’s candidate password. As shown in Figure 7.3, a suggested improvement for the password “Mypassword123” might be “My123passwoRzd.”

We generate this suggested improvement as follows. First, we take the user’s current candidate password and make one of the following modifications: toggle the case of a random letter (lowercase to uppercase, or vice versa), insert a random character in a random position, or randomly choose a character for which we substitute a randomly chosen different character. In addition, if all of the password’s digits or symbols are in a common location (e.g., at the end), we move them as a group to a random location within the password. We choose from among this large set of modifications, rather than just making modifications corresponding to the specific text feedback the meter displays, to greatly increase the space of possible modifications. We then verify that this modification still complies with the password-composition requirements. If so, we rate its strength by both the neural network and advanced heuristics.

To encourage passwords that we estimate would resist an offline attack, we require that the suggested improvement would both fill at least two-thirds of the bar and would be at least 1.5 orders of magnitude harder to guess than their current password. When we have generated a suggested improvement that meets our strength threshold and the user is currently showing his or her password, we display the suggested modification on screen with changes highlighted in magenta, as in Figure 7.3. If the user is not showing his or her password, we do not want to display the suggested improvement on screen because it would reveal the user’s password. Therefore, we instead show a blue button stating “see your password with our improvements.” If our first try at making such a modification is not enough of an improvement and the user

has not already modified his or her candidate password, we recursively try to make additional modifications. We stop recursing at a depth of 8 to avoid performance issues.

7.3.3 Specific-Feedback Modal

For the main screen, we designed the feedback specific to a user’s password to be both succinct and action-oriented. Although the rationale for these specific suggestions might be obvious to many users, we expected it would not be obvious to all users based on our prior work on password perceptions [181] and misconceptions [184].

To have the screen real estate to give more detailed explanations of why our specific suggestions would improve a password, we created a modal window. We term this window our *specific-feedback modal*, and we show an example of it for the password “Potat0es5678” in Figure 7.4. When a user clicks the blue, underlined “(Why?)” link next to each piece of password-specific feedback on the main screen, the modal appears.

The specific-feedback modal’s main bullet points mirror those from the main screen. As on the main screen, we show more detailed feedback (e.g., the specific dictionary words observed) when the user is showing his or her password. Below each of these main points, however, the specific-feedback modal also explains why this action would improve the password. Our explanations take one of two primary forms. In the first form, we explain how attackers could exploit particular characteristics we observe in the candidate password. For instance, as in Figure 7.4, we explain that attackers try simple transformations of dictionary words when we observe a password that contains such simple transformations. In the second form, we provide statistics about how common different characteristics are to discourage users from employing those characteristics. For instance, in Figure 7.4, we note (in more succinct phrasing) that 30% of passwords that contain a capital letter have only the first character of the password capitalized. We base all of our statistics on our analyses of the Xato corpus of 10 million passwords [34].

Because the detailed explanations might spur a user to modify his or her candidate password, we include a password field at the top of the specific-feedback modal. When the user has modified this password, the “ok” button at the bottom of the screen is replaced by buttons to cancel or keep the changes made to the password in the modal window.

7.3.4 Generic-Advice Modal

Whereas we designed the main screen and the specific-advice modal to give feedback particular to the candidate password a user had typed in, it is not always possible to generate such data-driven feedback. For instance, until a user has typed more than a few characters into the password field, the strength of the candidate password, or even what the candidate password might be, cannot yet be definitively determined. In addition, extremely predictable candidate passwords (e.g., “password” or “monkey1”) essentially require that the user completely rethink his or her strategy for creating a password.

We therefore also created a *generic-advice modal* (Figure 7.5) that recommends abstract strategies for creating passwords. Users access the generic-advice modal by clicking “how to make strong passwords” or “(Why?)” next to suggestions to avoid password reuse. The first of four points we make on the generic-advice modal advises against reusing a password across accounts. We chose to make this the first point because password reuse is very common [49, 68, 70, 169], yet is a major threat to security, as discussed in Section 2.1.

Abstract advice for creating a password can either focus on steps the user should take (i.e., “do this”) or common patterns they should avoid (i.e., “do not do this”). Because the former more directly guides a

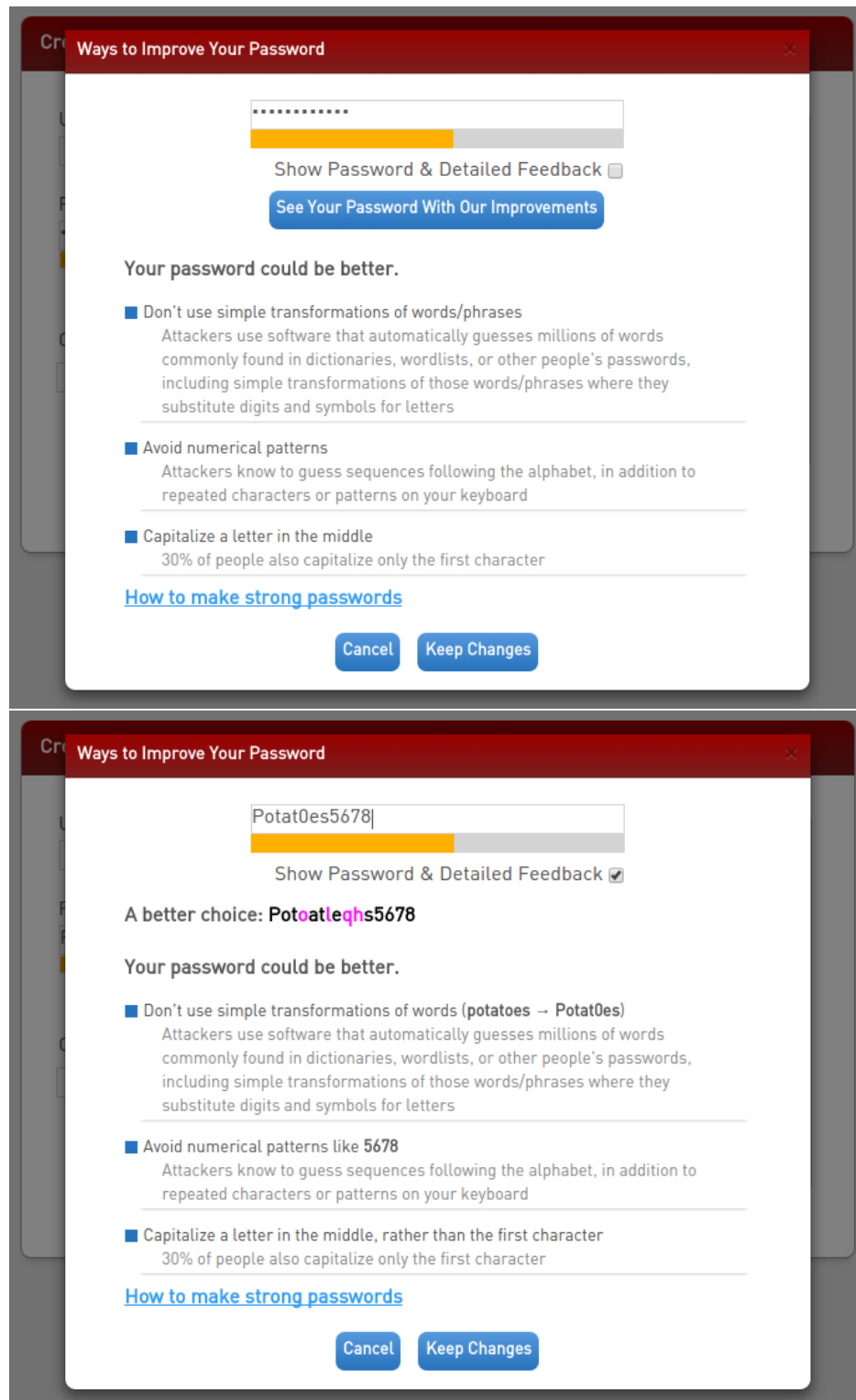


Figure 7.4: The meter's data-driven modal window giving feedback specific to the password the user typed. The two variants reflect when the password is hidden (top) and shown (bottom).



Figure 7.5: The meter’s generic-advice modal window, which warns against password reuse and provides guidance about strategies for creating strong passwords, as well as highlighting predictable decisions to avoid.

user towards taking some action, we chose to recommend constructive, action-oriented strategies as our second and third points. In particular, reflecting research that showed that passwords balancing length and character-class complexity were often strong [155] and because attackers can often brute-force passwords up to at least 9 characters in an offline attack [80, 171], our second point recommends using at least 12 characters in the password. To help inspire users who are unfamiliar with how to make a 12+ character password in unfamiliar ways, we use the third point to propose a way of doing so based on Schneier’s recommended method [152]. In particular, we encourage the user to make up a unique sentence and use mnemonics or fragments from that sentence as the basis for their password. Finally, the fourth point recommends against common password characteristics (Section 2.4) because some users mistakenly believe those characteristics to be secure [181, 184].

7.4 Formative Laboratory Study

While we had followed an iterative design process involving members of our research team and collaborators from industry, we wanted to solicit additional rich feedback from users. We therefore conducted a formative study in which we brought ten participants to our lab to use our password-strength meter and provide detailed feedback. We also interviewed them briefly about their password-creation habits, opinions, and strategies. Based on this formative laboratory study, we made numerous tweaks to the wording and visual design of the meter. Below, we discuss the study methodology, results, and our subsequent changes to the meter.

7.4.1 Methodology

We recruited participants from our local Craigslist site for a study about passwords. For the study, which lasted approximately 45 minutes, we compensated participants with a \$20 Amazon gift card.

The study comprised three parts. In the first part, we conducted a semi-structured interview that solicited the participant's goals when making a password, opinions about what makes a password secure, experiences with different sources of password feedback (e.g., password-strength meters) and advice (e.g., newspaper articles), and password management strategies. We designed these questions to gain a baseline understanding of the participant's opinions about password creation and management. In the second part, we showed participants a static slideshow of five password-strength meters we observed when we surveyed the use of password meters on the 100 most popular websites according to Alexa [5], updating for 2016 a survey we first performed in 2012 (Section 3.2). To prompt participants to start providing feedback about the user experience of password feedback, we asked participants to point out aspects they both liked and disliked of each meter they saw.

The third part of the study, which was of primary interest to us, gave the participant the chance to try two prototype password-strength meters. One meter, which we called the "red meter," represented our prototype meter at the moment the session was conducted. To give participants a point of comparison, we created a second meter, which we called the "purple meter," that used a re-colored version of the red meter's main screen. Rather than providing text feedback (other than the same feedback about meeting password-composition requirements) or using our scoring, the purple meter instead scored the password using `zxcvbn` [198]. The purple meter thus represents a best-in-class [51, 199] example of currently deployed password-strength meters. We did not indicate the provenance of these meters until the debrief at the end of the session. Instead, we told participants that these were two prototype concepts that we were testing, and we wanted both positive and negative feedback on both.

We provided the participant with a laptop that had the red and purple prototypes open in separate browser tabs. We instructed the participant to try a number of passwords on each of the prototypes "to get a feel for how they work." We also asked that they try some of the same passwords on both. If a participant ran out of ideas for passwords to try, we gave them a sheet of suggested passwords that would exercise key features of each prototype. We provided participants a piece of paper with columns for each meter's pros and cons, and we instructed them that they could either tell us their feedback as they went along or they could jot down their feedback on that sheet and summarize it for us at the end of the session. We emphasized that we were curious about both positive and negative feedback. We used the feedback from this portion of the session to identify potential usability issues.

The primary purpose of our formative laboratory study was to gather feedback on key usability characteristics of the meter with the intent of improving its wording, design, and user experience. We therefore chose to update the meter after each session to address usability problems or unclear aspects of the meter identified in that session. This approach limits the generalizability of our scientific findings, yet maximizes the benefits for improving the design of the meter, allowing us to quickly test potential remediations for usability shortfalls.

To facilitate analysis, we transcribed all interviews. We developed a preliminary codebook based on our initial hypotheses and the topics covered in the interview script. While coding, we updated the codebook to reflect themes of potential interest not captured by the initial codebook.

7.4.2 Results

We begin by describing the demographics of the ten participants in our in-lab interviews. We then present the key findings from our interviews, subsequently describing the design take-aways and resultant modifications we made to our meter prior to beginning the online study.

Participant Demographics

Participants' ages ranged from 22 to 61 (mean 39.2, median 41). Four participants identified as male, while the other six identified as female. One participant was unemployed, three were full-time students, and the other six were employed in a variety of fields (retail work, human services administration, social work, medical billing, audio engineering, and construction).

Overall, participants were not especially tech-savvy. Only one of the ten participants reported having a degree or job in a field related to computer science or technology, and all ten participants responded "no" when asked if they have "particular expertise in computer security topics." Despite this lack of technical expertise, all used passwords regularly. Nine of the ten participants reported creating or changing a password in the prior month, while the remaining participant reported doing so in the prior six months.

Key Findings

On the whole, most participants found the prototype representing our proposed meter to be helpful and informative. As P17 explained, "It gives me lots of feedback right away, and...it instructed me to put my capital in a different place, because I put it at the beginning where evidently most people do." In particular, she had not previously realized that her tendency to put capital letters at the beginning of passwords was very typical until clicking a "(why?)" link next to feedback our meter provided. Other participants noted that the optional nature of the feedback was an advantage, with P15 stating, "I liked that it had the little suggestion button, and it kind of like taught you things and it was available if you wanted it." Not every feature would suit every participant, though. P19 made up sentences as the basis for her passwords, and she found that the suggested improvement was useless because it was "the same letters, but jumbled around a bit with some new letters added, so it doesn't make sense with my original sentence."

Some participants found the meter's feedback slightly creepy. While P10 was explaining why he found the meter to be fun, he explained, "Like if you are repeating a phrase, it tells you you should not repeat a phrase. If you have just characters, it encourages you to put numbers and to mix uppercases. And it tells you to not use words that you could find in a dictionary. Yeah. So that's the fun part about it. But...it would make some users think, 'Oh someone's reading my password right now.' It's this sense of paranoia."

Another major theme in our laboratory study was participants' reluctance to click on aspects of the interface to learn more. Some information we deemed important was only available on these secondary screens, including full explanations for why to avoid certain patterns in passwords and our suggestions for ways to approach making a password, which is why we were concerned that many participants did not click on these elements. Some participants explained that they did not do so because they are "not interested in password education" or thought the feedback they received was obvious or otherwise self-explanatory. As P17 explained, "I'm very naive and trusting I guess...I look at my password and say, 'Yes, any fool can sort of figure out what I'm trying to put down there.'" In other cases, though, the reluctance to click was a misunderstanding. For instance, P15 did not click on the button to see the suggested improvement because

she thought it would be simply a random string. She did not understand why we forced users to take the step of showing their password, saying, “Like why couldn’t it just suggest a better one, unless it was somehow linked to what I had already [as my password] just changing something slightly.” Indeed, the password we would have shown was a slightly changed version of her password, and we subsequently changed the wording on the button to make this point clear.

A number of prior studies [49, 68, 169, 170, 184] have identified password reuse as very common, yet very problematic for security because of the nature of attacks against passwords (see Section 2.1). Unfortunately, our initial prototype meter did not emphasize to the user not to reuse passwords, and many of our participants said they regularly reused passwords. Two participants in particular reused the same password across all of their accounts. P19 explained, “There was a point when I had slightly different passwords for every site, but then I would forget which password was associated with each site, so I would get locked out of the site because I would try like 5 different passwords...I just like streamlined it and made [my passwords] all the same.” Similarly, P18 mentioned that “whether it’s a bank account, or email, or whether it’s a gambling account online...They’re all generally the same, if not close to the same, password.” When asked if he had any concerns about this approach, he said, “No, because I don’t give [my password] to nobody.” Previously, he carried around a rubber-banded stack of index cards with passwords, but tired of doing so.

7.4.3 Takeaways and Changes to the Meter

A major set of changes during and following the laboratory study related to users’ workflow through the three screens described in Section 7.3. Few of the initial participants in the laboratory study visited either modal window, and few chose to show their password on screen, which means that they did not see the suggested improvement at all. As a result, we tested a number of variants to the workflow for subsequent participants. We tried varying the layout, colors, format, and text descriptions for all buttons and links that would bring the user to either modal window or show their password.

Aspects of the final prototype that we changed as a result of this experimentation include adding “& detailed feedback” to “show password” options, placing “(Why?)” links next to each piece of text feedback rather than presenting a single “learn more” button at the bottom of the window, and adding a “see your password with our improvements” button where the suggested improvement would appear if the password were shown. The text of this final button also underwent several iterations as a result of our participants’ feedback. While we initially used the text “Show a better choice” and “Show a stronger password,” participants expected that we would be presenting a randomly generated password, rather than their password with randomly selected modifications. As a result, participants tended not to click “show a stronger password,” and the one that did was surprised that a small modification of her password appeared on screen. Based on the surprised participant’s feedback and discussions within our group, we settled on the text “see your password with our improvements.”

Password reuse is among the major threats to password security, as discussed in Section 2.1. Even after using our meter, though, two of our ten participants stated that reusing the same password across sites is not problematic because their password was strong, echoing findings from prior work on security misconceptions [184]. As a result, we revised the meter’s user experience to strongly advise against password reuse. As shown in Figure 7.1 of the previous section, our first text feedback, even before listing password-composition requirements, is now an admonition against password reuse with a “(Why?)” link to our generic-advice modal. Similarly, when the meter estimates that the password would resist an offline attack,

the meter notes, “Your password is pretty good. Use it only for this account. (Why?)”

Another series of changes involved how we classified and discussed strings. Some strings conceptually fall into numerous categories. For instance, “aaaa” is a repetition of the same character (the `repeats()` function), yet also is a keyboard pattern by our definition. It also appears in many dictionaries as an initialism. This overlap confused participants. They were surprised, for instance, that “qwerty” was considered a dictionary word and that repeated characters were considered a keyboard pattern. As a result, we rebuilt our dictionaries, removing all strings that were keyboard patterns, repetition of the same character, or other patterns (e.g., “xyz” and “1357”). We also eliminated overlap among these functions, such as preventing “aaaa” from being classified as a keyboard pattern.

As a result of participant confusion, we also changed terminology and separated what we now call “alphabetic patterns” (e.g., “xyz” and “aceg”) from what we now call “numerical patterns” (e.g., “1357”). Similarly, while we had initially classified words that appeared commonly on Wikipedia as “dictionary words,” participants disliked that surnames, uncommon words, and other proper names were termed dictionary words. As P1 ruminated during his interview while expressing surprise at parts of his password that were identified as dictionary words, “It’s like playing words with friends, except not really. It is a real word, right?” We therefore now identify any non-dictionary word that falls into this category as a “word used on Wikipedia.”

Similarly, our function to identify dates had initially been very inclusive of what strings it judged to be dates because of the vast array of ways in which dates appear in passwords [189]. Participants found it confusing that the function identified what they intended to be fairly arbitrary strings of digits as dates, so we subsequently winnowed down the list of patterns recognized by the dates function to include only the most obvious dates.

We also made a number of minor wording tweaks, including to the wording we used when a password was estimated to resist an offline attack (now the hints are “to make it even better”) and in how we explained the technique of creating a novel sentence as the basis for a password. In response to a participant questioning the meter’s feedback not to use the character transformation they had employed, we expanded our explanation about how attackers guess simple transformations of common words. Finally, one participant who had a disability brought up accessibility concerns, encouraging us to plan on adding additional support for screen readers beyond those we had already considered.

7.5 Summative Online Study

After making all of the changes suggested by our formative laboratory study and further stress-testing the meter among our team and industry collaborators, we conducted a summative online study. The dual purposes of this study were both to measure how the meter impacted both the security and usability of the passwords participants created, as well as to quantify on a larger scale how users interacted with different elements of the meter.

7.5.1 Methodology

We recruited participants from Amazon’s Mechanical Turk³ crowdsourcing service for a study on passwords. We required that participants be age 18+ and be located in the United States. In addition, because we had

³Amazon’s Mechanical Turk. <https://www.mturk.com>

only verified that the meter worked correctly on Firefox, Chrome/Chromium, Safari, and Opera, we required they use one of those browsers.

In order to measure both password creation and password recall, the study comprised two parts. The first part of the study, which we term *Part 1*, included a password creation task, a survey, and a password-recall task. We assigned participants round-robin to a condition specifying the meter variant they would see when creating a password. After studying 18 conditions in our first experiment, we were left with lingering questions. We therefore ran a second experiment that added 8 new conditions and repeated 4 existing conditions.

The second part of the study, which we term *Part 2*, took place at least 48 hours after the first part of the study and included a password-recall task and a survey. We compensated participants \$0.55 for completing Part 1 and \$0.70 for Part 2.

Part 1

Following the consent process, we instructed participants that they would be creating a password. We asked that they role play and imagine that they were creating this password for “an account they care a lot about, such as their primary email account.” We informed participants they would be invited back in a few days to recall the password and asked them to “take the steps you would normally take to create and remember your important passwords, and protect this password as you normally would protect your important passwords.”

The participant then created a username and a password. While doing so, he or she saw the password-strength meter (or lack thereof) dictated by his or her assigned condition, described below. Participants then answered a survey about how they created that password. We first asked participants to respond on a 5-point scale (“strongly disagree,” “disagree,” “neutral,” “agree,” “strongly agree”) to statements about whether creating a password was “annoying,” “fun,” or “difficult.” We also asked whether they reused a previous password, modified a previous password, or created an entirely new password.

The next three parts of the survey asked about the meter’s colored bar, text feedback, and suggested improvements. At the top of each page, we showed a text explanation and visual example of the feature in question. Participants in conditions that lacked one or more of these features were not asked questions about that feature. For the first two features, we asked participants to rate on a five-point scale whether that feature “helped me create a strong password,” “was not informative,” and caused them to create “a different password than [they] would have otherwise.” We also asked about the importance participants place on the meter giving their password a high score, their perception of the accuracy of the strength rating, and whether they learned anything new from the feedback.

After the participant completed the survey, we brought him or her to a login page and auto-filled his or her username. The participant then attempted to re-enter his or her password. We refer to this final step as *Part 1 recall*. After five incorrect attempts, we let the participant proceed.

Part 2

After 48 hours, we automatically emailed participants to return and re-enter their password. We term this step *Part 2 recall*, and it was identical to Part 1 recall. We then directed participants to a survey about how they tried to remember their password. In particular, we first asked how they entered their password on the previous screen. We gave options encompassing automatic entry by a password manager or browser, typing the password in entirely from memory, and looking a password up on paper or electronically.

Conditions

In Experiment 1, we assigned participants round-robin to one of 18 different conditions that differed across three dimensions in a full-factorial design. We refer to our conditions using three-part names reflecting the dimensions: 1) password-composition policy; 2) type of feedback; and 3) stringency. Table 7.2 lists the 18 conditions we tested and their short-form names, which we use throughout the results section.

Dimension 1: Password-Composition Policy We expected a meter to have a different impact of password security and usability if used in association with a minimal or a more complex password-composition policy. We tested the following two policies, which respectively represent a widespread, lax policy and a more complex policy.

- **1class8 (1c8)** requires that passwords contain 8 or more characters, and also that they not be (case-sensitive) one of the 96,480 such passwords that appeared four or more times in the Xato corpus of 10 million passwords [34].
- **3class12 (3c12)** requires that passwords contain 12 or more characters from at least 3 different character classes (lowercase letters, uppercase letters, digits, and symbols). It also requires that the password not be (case-sensitive) one of the 96,926 such passwords that appeared in the Xato corpus [34].

Dimension 2: Type of Feedback Our second dimension varies the type of feedback we provide to participants about their password. While the first setting represents our standard meter, we removed features for each of the other settings to test the impact of those features.

- **Standard (Std)** includes all previously described features.
- **No Suggested Improvement (StdNS)** is the same as Standard, except it never displays a suggested improvement.
- **Public (Pub)** is the same as standard, except we never show sensitive text feedback (i.e., we never show a suggested improvement and always show the less informative “public” feedback normally shown when the password is hidden).
- **Bar Only (Bar)** shows a colored bar displaying password strength, but we do not provide any type of text feedback other than which composition requirements have been met.
- **No Feedback (None)** gives no feedback whatsoever.

Dimension 3: Scoring Stringency Ur et al. found the stringency of a meter’s scoring has a significant impact on password strength [182]. We thus tested two scoring stringencies. These stringencies changed the mapping between the estimated number of guesses the password would resist and how much of the colored bar was filled.

- **Medium (M)** One-third of the bar full represents 10^6 estimated guesses and two-thirds full represents 10^{12} guesses.
- **High (H)** One-third of the bar full represents 10^8 estimated guesses and two-thirds full represents 10^{16} guesses.

Table 7.2: The conditions we tested in Experiment 1 of our online study. The bold text represents the short-form names we use throughout our results section, while the corresponding non-bold text details the settings of each dimension.

1c8-Std-M	(1class8-Standard-Med.)	3c12-Std-M	(3class12-Standard-Med.)
1c8-Std-H	(1class8-Standard-High)	3c12-Std-H	(3class12-Standard-High)
1c8-StdNS-M	(1class8-NoSuggestion-Med.)	3c12-StdNS-M	(3class12-NoSuggestion-med.)
1c8-StdNS-H	(1class8-NoSuggestion-High)	3c12-StdNS-H	(3class12-NoSuggestion-High)
1c8-Pub-M	(1class8-Public-Med.)	3c12-Pub-M	(3class12-Public-Med.)
1c8-Pub-H	(1class8-Public-High)	3c12-Pub-H	(3class12-Public-High)
1c8-Bar-M	(1class8-BarOnly-Med.)	3c12-Bar-M	(3class12-BarOnly-Med.)
1c8-Bar-H	(1class8-BarOnly-High)	3c12-Bar-H	(3class12-BarOnly-High)
1c8-None	(1class8-NoFeedback)	3c12-None	(3class12-NoFeedback)

Table 7.3: The conditions we tested in Experiment 2 of our online study. The bold text represents the short-form names we use throughout our results section, while the corresponding non-bold text details the settings of each dimension.

1c8-Std-L	(1class8-Standard-Low)	3c12-Std-L	(3class12-Standard-Low)
1c8-Std-M	(1class8-Standard-Med.)	3c12-Std-M	(3class12-Standard-Med.)
1c8-Std-H	(1class8-Standard-High)	3c12-Std-H	(3class12-Standard-High)
1c8-NoBar-L	(1class8-No Bar-Low)	3c12-NoBar-L	(3class12-No Bar-Low)
1c8-NoBar-M	(1class8-No Bar-Med.)	3c12-NoBar-M	(3class12-No Bar-Med.)
1c8-NoBar-H	(1class8-No Bar-High)	3c12-NoBar-H	(3class12-No Bar-High)

Additional Conditions for Experiment 2 Experiment 2 added the following two settings for our feedback and stringency dimensions, respectively:

- Feedback: **Standard, No Bar (NoBar)** The Standard meter without any colored bar. The text feedback still depends on the password’s score, so stringency still matters.
- Stringency: **Low (L)** One-third of the bar full represents 10^4 estimated guesses and two-thirds full represents 10^8

To investigate these two settings we introduced eight new conditions and re-ran the four existing standard feedback medium and high conditions in a full-factorial design, as detailed in Table 7.3.

Analysis

We collected numerous types of data for each participant. Our main security metric was the guessability of each participant’s password, as calculated by Carnegie Mellon University’s Password Guessability Service [37], which models four types of guessing attacks and which they found to be a conservative proxy for an expert attacker [185]. Our usability measurements encompassed both quantitative and qualitative data. We recorded participants’ keystrokes as they created their password, enabling us to analyze metrics like password

creation time. To understand participants' use of different features, we also instrumented all elements of the user interface to record when they were clicked.

For both Part 1 and Part 2, we measured whether participants successfully recalled their password. For participants who did successfully recall their password, we also measured how long it took them, as well as how many attempts were required. Because not all participants returned for Part 2, we also measured what proportion of participants did, hypothesizing that participants who did not remember their password might be less likely to return. A crucial consideration is how participants remembered their password. To only study attempts at recalling a password from memory, we analyzed password recall only among participants who said they typed their password in entirely from memory, said they did not reuse their study password, and whose keystrokes did not show evidence of copy-pasting.

We augmented our objective measurements with analyses of responses to multiple-choice questions and qualitative analysis of free-text responses. These optional free-text responses solicited participants' thoughts about the interface elements, as well as why they did (or did not) find them useful.

Our primary goal was understanding how varying the three meter-design dimensions impacted our quantitative metrics. Because we had multiple independent variables, each reflecting one design dimension, we performed regressions. We ran a linear regression for continuous data (e.g., the time to create a password), a logistic regression for binary data (e.g., whether or not they clicked on a given UI element), an ordinal regression for ordinal data (e.g., Likert-scale responses), and a multinomial logistic regression for categorical data with no clear ordering (e.g., how they entered their password).

For our security analyses, we performed a Cox Proportional-Hazards Regression, which is borrowed from the literature on survival analysis and was used by Mazurek et al. to compare password guessability [126]. Because we know the starting point of guessing but not the endpoint, we use a right-censored model [72]. In a traditional model using survival analysis in a clinical setting, each data point can be marked as "alive" or "deceased," along with the time of the observation. Our analogues for password guessing are "not guessed" and "guessed," along with the number of guesses at which the password was guessed, or the guessing cutoff.

We always first fit a model with the three design dimensions (composition policy, feedback, and stringency) each treated as ordinal variables fit linearly, as well as interaction terms for each pair of dimensions. To build a parsimonious model, we then removed any interaction terms that were not statistically significant, yet always kept all three main effects, and re-ran the model.

We corrected for multiple testing using the Benjamini-Hochberg (BH) procedure [14]. We chose this approach, which is more powerful and less conservative than methods like Holm Correction, because we performed an exploratory study with a large number of variables. We corrected all p-values for each experiment as a group. We use $\alpha = 0.05$.

Note that we analyzed Experiments 1 and 2 separately. Thus, we do not compare conditions between experiments. However, in the graphs and tables that follow we have combined our reporting of these two experiments for brevity. We only report "NoBar" and low-stringency data from Experiment 2 in these tables and graphs. For conditions that were part of both experiments, we only report the results from Experiment 1.

7.5.2 Limitations

We based our study's design on one researchers have used previously to investigate various aspects of passwords [92, 108, 155, 182]. However, the password participants created did not protect anything of value. Beyond our request that they do so, participants did not need to exhibit their normal behavior. Mazurek et

Table 7.4: Results of our initial Cox regression model for Experiment 1, including all three dimensions and their interactions. Bold p values are significant.

Dimension	coef	exp(coef)	se(coef)	z	Pr(> z)
Composition policy	-0.687	0.503	0.053	-13.032	<.001
Type of feedback	-0.542	0.582	0.373	-1.451	0.147
Scoring stringency	0.006	1.006	0.173	0.034	0.973
Policy * Feedback	0.246	1.279	0.124	1.994	0.046
Policy * Stringency	-0.041	0.960	0.075	-0.552	0.581
Feedback * Stringency	-0.184	0.832	0.526	-0.349	0.727

al. [126] and Fahl et al. [67] examined the ecological validity of this protocol, finding it to be a reasonable, albeit imperfect, proxy for high-value passwords for real accounts.

That said, no controlled experiment can capture every aspect of password creation. We tested recall at two points in time. Similarly, we did not test habituation effects, either to the use of a particular password or to the novel password-meter features we tested. We did not control the device [192, 203] on which participants created a password, nor could we control how participants chose to remember their password.

7.5.3 Participants

We had 4,509 participants (2,717 in Experiment 1 and 1,792 in Experiment 2), and 84.1% of them returned for Part 2. Among our participants, 52% identified as female, 47% identified as male, and the remaining 1% identified as another gender or preferred not to answer. Participants' ages ranged from 18 to 80 years old, with a median of 32 (mean 34.7). We asked whether participants are "majoring in or...have a degree or job in computer science, computer engineering, information technology, or a related field," and 82% responded "no." Demographics did not vary significantly by condition.

7.5.4 Security Impact

Increasing levels of data-driven feedback, even beyond just a colored bar, led users to create stronger 1class8 passwords. That is, detailed text feedback led to even more secure passwords than just the colored bar alone. The 3class12 policy also led to stronger passwords, but varying the stringency had only a small impact.

Impact of Composition Policy

We first ran a Cox regression with all three dimensions and their pairwise interactions as independent variables, and the password's survival term as the dependent variable. For Experiment 1, we found significant main effects for both the policy and type of feedback, but we also observed a significant interaction effect between the policy and type of feedback (Table 7.7). For increased intelligibility, we subsequently ran separate regressions for 1class8 and 3class12 passwords. As the 3class12 policy requires longer passwords than 1class8, participants unsurprisingly created 3class12 passwords that were significantly longer ($p < .001$) and significantly more secure ($p < .001$) than 1class8 passwords. Similarly, 3class12 passwords included more digits, symbols, and uppercase letters than 1class8 passwords (all $p < .001$). Table 7.10 details these password characteristics per condition.

Table 7.5: Results of our Cox regression model for 1class8 passwords in Experiment 1.

Dimension	coef	exp(coef)	se(coef)	z	Pr(> z)
Type of feedback	-0.575	0.563	0.099	-5.787	<.001
Scoring stringency	-0.021	0.980	0.061	-0.338	0.825

Table 7.6: Results of our Cox regression model for 3class12 passwords in Experiment 1.

Dimension	coef	exp(coef)	se(coef)	z	Pr(> z)
Type of feedback	-0.239	0.787	0.144	-1.667	0.171
Scoring stringency	-0.078	0.925	0.086	-0.910	0.463

Impact of the Amount of Feedback

For 1class8 passwords, we found that increasing levels of data-driven feedback led participants to create significantly stronger passwords ($p < .001$). Relative to having no feedback, the full suite of data-driven feedback led to 44% stronger passwords. As shown in Figure 7.6, the colored bar on its own led participants to create stronger passwords than having no feedback, echoing prior work [182]. The detailed text feedback we introduce in this work led to even stronger passwords than just the bar. Increasing the amount of feedback also led participants to create longer passwords ($p < .001$). It also led participants to include more digits, symbols, and uppercase letters ($p = .013$, $p = .001$, $p = .039$). For example, the median length of 1c8-None passwords was 10 characters, whereas the median for 1c8-Std-M was 12 characters.

Notably, the security of 1class8 passwords created with our standard meter (including all text feedback) was more similar to the security of 3class12 passwords created without feedback than to 1class8 passwords created without feedback, as shown in Figure 7.6.

To test how showing more detailed text and the suggested improvement impacts the passwords participants created, we compared our standard meter (“Std”) to a meter that is otherwise identical to our standard meter, yet never offers a suggested improvement (“StdNS”), the variant that never shows sensitive feedback and never shows a suggested improvement (“Pub”), and control variants where we show only the colored bar without text feedback (“Bar”) and no feedback (“None”). We also examined the variant of the standard meter that includes all text feedback, yet leaves out the colored bar (“NoBar”). Figure 7.7 details the comparative security impact of all six feedback levels. Whereas suggested improvements had minimal impact, having the option to show potentially sensitive feedback provided some security benefits over the public (“Pub”) variant. When we investigated removing the colored bar from the standard meter, but leaving the text feedback, we found that removing the colored bar did not significantly impact password strength.

For 3class12 passwords, however, the level of feedback did not significantly impact password strength, as shown in Figure 7.8. We hypothesize that either we are observing a ceiling effect, in which the 3class12 policy by itself led participants to make sufficiently strong passwords, or that the text feedback does not provide sufficiently useful recommendations for a 3class12 policy.

Table 7.7: Results of our initial Cox regression model for Experiment 2, including all three dimensions and their interactions.

Dimension	coef	exp(coef)	se(coef)	z	Pr(> z)
Composition policy	-0.391	0.676	0.101	-3.868	<.001
Type of feedback (Bar / No Bar)	-0.059	0.943	0.072	-0.809	0.418
Scoring stringency	-0.273	0.761	0.140	-1.958	0.050
Policy * Feedback	-0.285	0.752	0.143	-2.002	0.045
Policy * Stringency	0.547	1.728	0.277	1.975	0.048
Feedback * Stringency	-0.319	0.727	0.197	-1.616	0.106

Table 7.8: Results of our Cox regression model for 1class8 passwords in Experiment 2.

Dimension	coef	exp(coef)	se(coef)	z	Pr(> z)
Type of feedback (Bar / No Bar)	-0.043	0.958	0.090	-0.475	0.726
Scoring stringency	-0.124	0.883	0.114	-1.091	0.426

Table 7.9: Results of our Cox regression model for 3class12 passwords in Experiment 2.

Dimension	coef	exp(coef)	se(coef)	z	Pr(> z)
Type of feedback (Bar / No Bar)	-0.082	0.921	0.097	-0.849	0.543
Scoring stringency	-0.318	0.728	0.126	-2.522	0.043

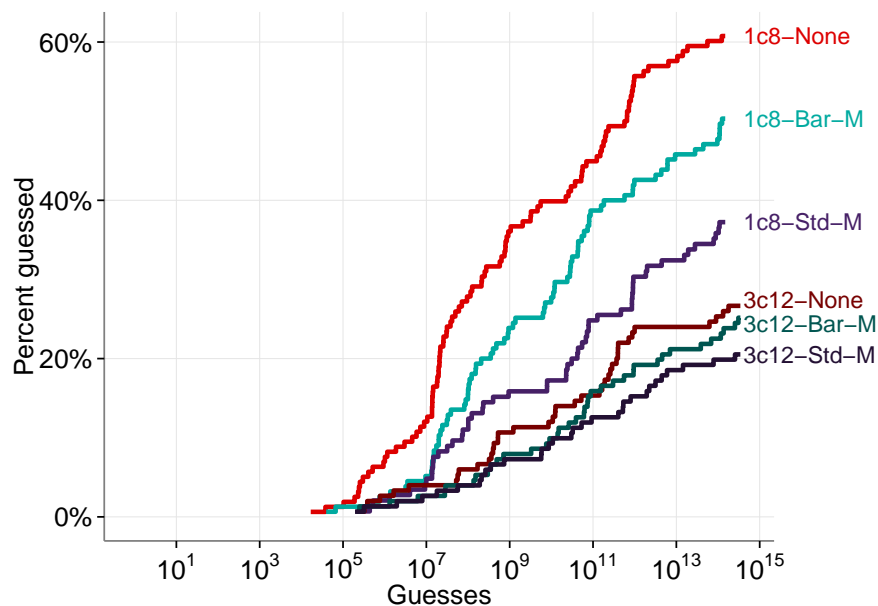
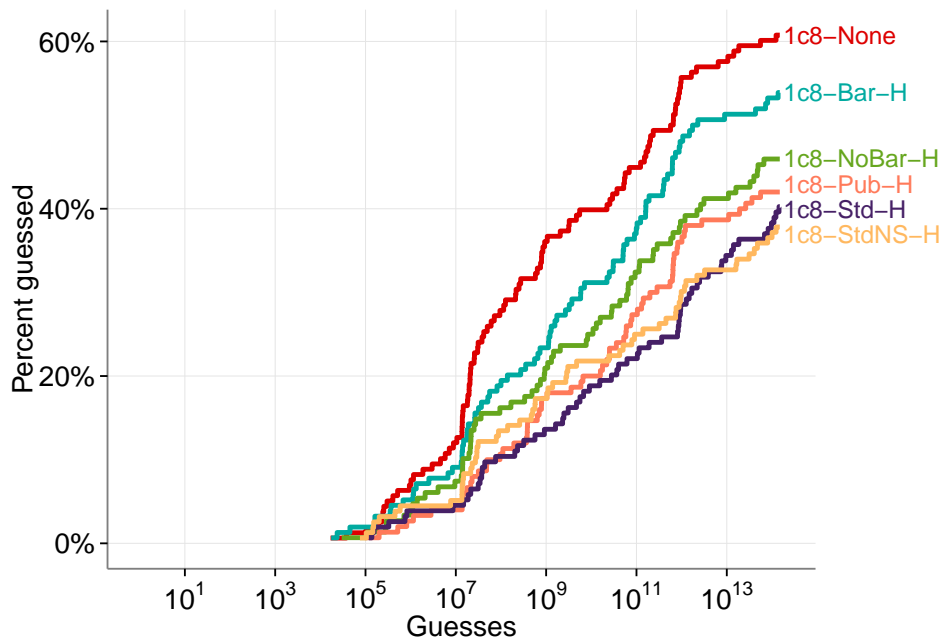
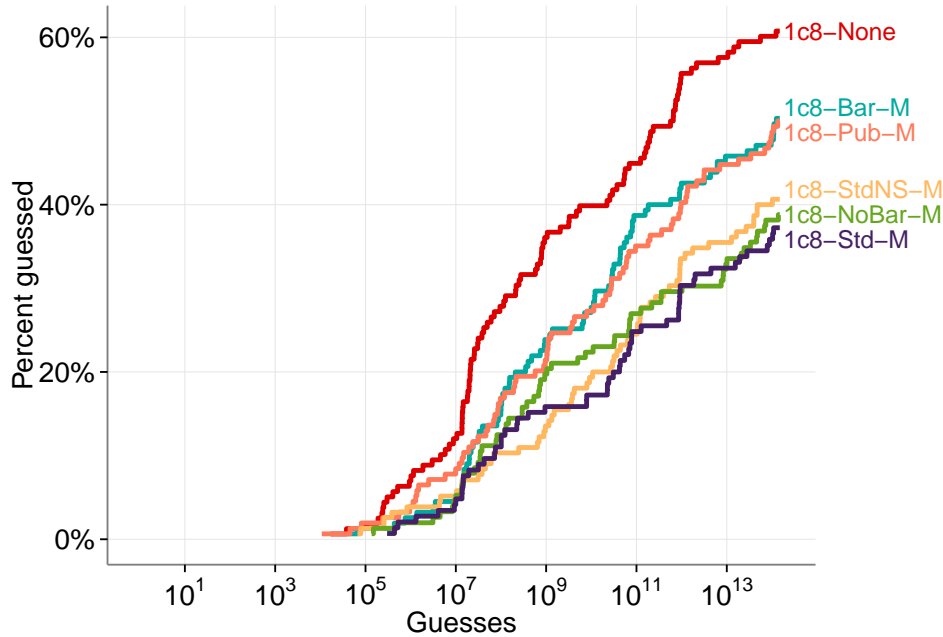


Figure 7.6: The guessability of passwords created without any feedback (“None”), with only a colored bar (“Bar”), and with both a colored bar and text feedback (“Std”) among medium-stringency conditions.



(a) 1class8, High-stringency



(b) 1class8, Medium-stringency

Figure 7.7: Guessability of 1class8 passwords created with different levels of feedback, including none. These graphs show data from Experiment 1. However, 1c8-NoBar-M and 1c8-NoBar-H curves from Experiment 2 have been added for illustrative purposes.

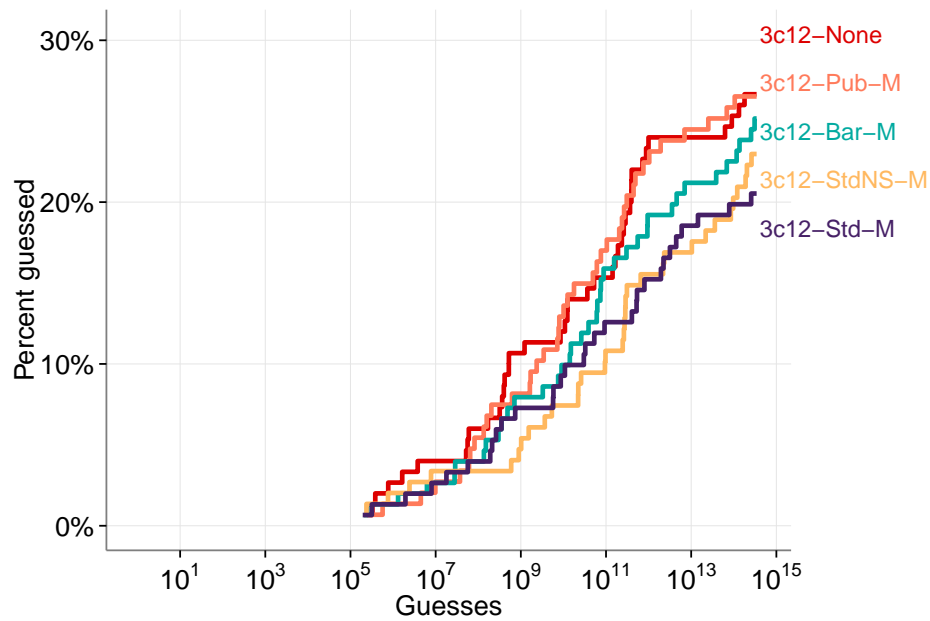


Figure 7.8: Guessability of medium-stringency 3class12 passwords created with different levels of feedback. For 3class12 passwords, the amount of feedback did not significantly impact guessability, which means that no differences in this graph are statistically significant.

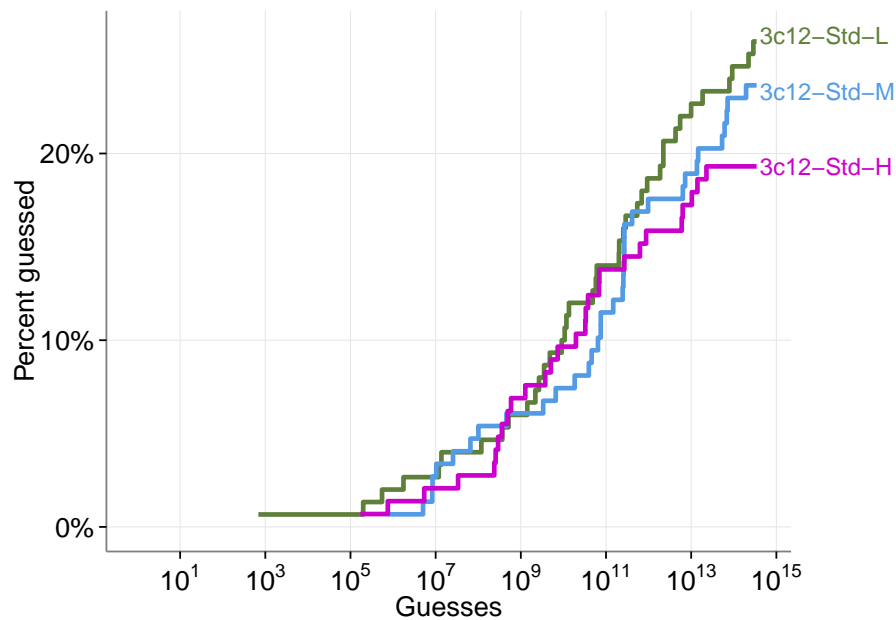


Figure 7.9: The guessability of 3class12 passwords created with different stringency levels. Stringency was a significant factor in the guessability, albeit with a small effect size.

Table 7.10: The characteristics of the passwords participants created in each condition.

Set	Length			# Uppercase			# Digits			# Symbols		
	Mean	σ	Med.	Mean	σ	Med.	Mean	σ	Med.	Mean	σ	Med.
1c8-Std-L	12.0	4.7	11	1.2	1.5	1	2.5	2.3	2	0.7	1.1	0
1c8-Std-M	13.0	5.4	12	1.5	2.1	1	2.9	2.5	2	0.9	1.4	0
1c8-Std-H	12.9	4.4	11	1.9	2.1	1	3.1	2.3	3	0.9	1.2	1
1c8-StdNS-M	12.1	3.7	12	1.4	1.8	1	3.0	2.1	3	0.6	1.1	0
1c8-StdNS-H	13.4	5.3	12	1.4	1.8	1	3.2	2.3	3	0.9	1.3	0
1c8-NoBar-L	11.5	3.5	11	1.3	2.0	1	2.6	1.8	2	0.6	1.1	0
1c8-NoBar-M	12.1	3.9	11	1.5	1.9	1	2.8	2.0	3	0.9	1.6	0
1c8-NoBar-H	11.7	3.9	11	1.6	1.8	1	3.0	2.1	3	0.8	1.3	0
1c8-Pub-M	11.6	3.1	11	1.1	1.6	1	3.0	2.2	3	0.6	0.9	0
1c8-Pub-H	13.0	4.4	12	1.6	2.0	1	3.1	2.2	3	0.9	1.3	0
1c8-Bar-M	11.4	3.1	11	1.1	1.2	1	2.7	1.9	2	0.6	1.0	0
1c8-Bar-H	11.6	3.2	11	1.3	1.8	1	2.8	2.1	2	0.6	1.1	0
1c8-None	10.9	3.0	10	1.0	1.3	1	2.6	2.0	2	0.3	0.7	0
3c12-Std-L	14.0	2.5	13	1.9	1.7	1	3.1	1.8	3	1.1	1.1	1
3c12-Std-M	14.8	3.5	14	2.0	1.3	2	3.6	2.6	3	1.2	1.4	1
3c12-Std-H	15.3	5.2	14	2.4	2.1	2	3.9	2.2	4	1.2	1.5	1
1c8-StdNS-M	14.4	3.1	14	2.1	1.9	2	3.7	2.3	4	1.1	1.5	1
1c8-StdNS-H	15.1	3.7	14	1.9	1.3	2	3.8	2.3	4	1.4	1.4	1
3c12-NoBar-L	14.3	2.9	13	1.9	1.7	1	3.4	1.9	3.5	1.0	1.4	1
3c12-NoBar-M	14.7	7.5	13	2.3	3.5	1	2.8	1.9	3	1.2	1.7	1
3c12-NoBar-H	14.7	4.7	14	2.1	1.5	2	3.6	2.2	3	1.3	1.5	1
3c12-Pub-M	14.1	2.8	13	2.1	1.8	2	3.4	2.0	3	1.2	1.2	1
3c12-Pub-H	14.9	3.4	14	2.2	1.7	2	3.6	2.2	3	1.1	1.2	1
3c12-Bar-M	14.5	3.1	14	2.0	2.2	1	3.3	2.1	3	1.0	1.0	1
3c12-Bar-H	14.6	3.4	14	2.0	2.0	1	3.7	2.1	4	1.1	1.2	1
3c12-None	14.2	2.9	13	2.2	2.3	1	3.4	2.3	3	1.2	1.3	1

Impact of Stringency

Although prior work on password meters found that increased scoring stringency led to stronger passwords [182], we found that varying between medium and high stringency did not significantly impact security. Because both our medium and high stringency levels were more stringent than most real-world password meters, we investigated an additional low stringency setting in Experiment 2. With these three levels, we found that increasing levels of stringency did lead to stronger passwords, but only for 3class12 passwords ($p = .043$). Figure 7.9 shows the impact of varying the 3class12 stringency. In all cases, however, increasing the stringency led participants to create longer passwords ($p < .001$). It also led participants to include more digits, symbols, and uppercase letters ($p = .036$, $p = .018$, $p = .019$). For example, the median high-stringency 1class8 password was one character longer than the median low-stringency 1class8 password. For 3class12 passwords, high-stringency passwords were two characters longer. Note that the prior work [182] tested a meter that used only basic heuristics (length and character classes) to score passwords, with a particular emphasis on length. As a result, participants could fill more of the stringent meters simply by making their password longer. In contrast, our meter scores passwords far more rigorously, which we hypothesize might account for this difference.

Table 7.11: Characteristics of password creation and password recall by condition.

Set	Creation Time (s)			Deletions			Novel Recall	Part 2 Recall				
	Mean	σ	Med.	Mean	σ	Med.	From Memory %	Success %	Attempts			
									Mean	σ	Med.	
1c8-Std-L	42.2	47.3	26.6	9.5	14.3	4	69.2	85.3	1.4	1.0	1	
1c8-Std-M	59.0	130.5	28.1	11.4	16.0	6	51.0	75.4	1.3	0.9	1	
1c8-Std-H	67.2	85.2	40.2	16.4	24.8	9	51.3	82.4	1.4	0.7	1	
1c8-StdNS-M	44.4	61.6	26.8	13.0	23.5	7	59.4	80.0	1.3	0.7	1	
1c8-StdNS-H	73.5	94.4	43.4	18.6	24.4	11	47.4	80.0	1.3	0.8	1	
1c8-NoBar-L	41.0	50.8	18.6	7.7	11.6	2	59.3	84.4	1.1	0.5	1	
1c8-NoBar-M	72.9	143.9	30.6	13.2	19.9	7	48.0	83.0	1.5	0.8	1	
1c8-NoBar-H	65.5	87.1	32.8	13.8	19.8	4	54.1	77.6	1.4	0.7	1	
1c8-Pub-M	46.0	54.7	28.7	11.6	16.3	6.5	53.2	78.8	1.3	0.8	1	
1c8-Pub-H	67.7	75.5	42.7	22.0	33.5	10.5	52.0	83.1	1.4	0.9	1	
1c8-Bar-M	29.7	47.5	15.5	6.5	12.5	0	54.2	82.3	1.4	0.7	1	
1c8-Bar-H	28.0	31.3	15.8	6.0	10.3	0	49.4	84.4	1.5	0.9	1	
1c8-None	21.5	26.7	10.9	3.3	5.3	0	54.4	84.9	1.4	1.0	1	
3c12-Std-L	56.6	72.9	35.4	9.9	12.2	7	58.0	81.0	1.4	0.8	1	
3c12-Std-M	62.8	80.8	42.2	15.1	23.0	8	53.6	79.1	1.3	0.8	1	
3c12-Std-H	76.7	101.3	46.3	15.9	21.5	10	44.8	72.2	1.5	0.9	1	
1c8-StdNS-M	65.9	67.4	41.5	15.6	19.4	11	54.7	71.0	1.7	1.2	1	
1c8-StdNS-H	65.7	96.6	42.2	14.8	21.0	10	45.0	71.2	1.5	0.8	1	
3c12-NoBar-L	57.5	92.6	35.1	10.9	15.6	5	56.1	69.6	1.4	0.8	1	
3c12-NoBar-M	61.2	57.0	43.2	13.5	16.6	9	46.9	72.5	1.4	0.6	1	
3c12-NoBar-H	72.3	87.9	42.5	14.8	17.9	9	49.7	83.6	1.6	1.1	1	
3c12-Pub-M	53.9	53.4	37.7	12.0	15.9	7	46.3	76.5	1.3	0.8	1	
3c12-Pub-H	113.8	54.3	40.1	16.4	21.0	10.5	42.4	80.8	1.6	1.1	1	
3c12-Bar-M	49.3	59.4	31.8	10.2	14.0	6	49.7	71.0	1.4	0.8	1	
3c12-Bar-H	41.0	34.5	29.0	8.8	12.0	3	50.3	79.7	1.5	0.9	1	
3c12-None	51.8	90.1	31.6	10.0	13.6	6	50.7	73.5	1.4	0.7	1	

7.5.5 Usability Impact

All of the meter's design dimensions we tested impacted timing and participant-sentiment metrics, but they mostly did not impact password memorability. Although increasing levels of data-driven feedback led to stronger passwords, we did not observe any significant impact on memorability. We tested many metrics; Table 7.12 summarizes our key findings. If we do not explicitly note a metric as being impacted by one or more dimensions, we did not observe a significant difference.

Overall, 56.5% of participants said they typed their password in entirely from memory. Other participants looked it up on paper (14.6%) or on an electronic source (12.8%), such as their computer or phone. An additional 11.2% of participants said their password was entered automatically for them by a password manager or browser, while 4.9% entered their password in another way (e.g., looking up hints).

Considering password reuse and copy-pasting, 50.6% of participants tried to recall a novel study password from memory, and these are the participants for whom we examine password recall. Overall, 98.4% of these participants successfully recalled their password during Part 1, and the majority did so on their first attempt. In total, 89.3% of participants returned for Part 2, and 78.2% of returnees who tried to recall a novel study password from memory successfully recalled their password, again primarily on their first attempt.

Table 7.12: A summary of how the three dimensions impacted key metrics. We consider the impact when moving from a 1class8 to 3class12 policy, increasing the amount of feedback, or increasing stringency.

Metric	Composition Policy	Amount of Feedback	Scoring Stringency
<i>Security</i>			
Passwords harder to guess	✓	1class8 only	3class12 only
<i>Password creation</i>			
Longer passwords	✓	✓	✓
More time to create	✓	✓	✓
More deletions	–	✓	✓
More likely to show password on screen	✓	–	✓
Less likely to show password on screen	–	✓	–
Care more about receiving a high score from meter	–	✓	–
Care less about receiving a high score from meter	–	–	✓
More likely to show suggested improvement	–	–	✓
<i>Sentiment about creation</i>			
More annoying	✓	✓	✓
More difficult	✓	✓	✓
Less fun	–	✓	✓
<i>Password recall</i>			
Lower recall success (less memorable) in Part 1	–	–	–
Part 1 recall took longer	–	–	✓
Lower recall success (less memorable) in Part 2	✓	–	–
Part 2 recall took longer	–	–	–
Required more attempts in Part 1 or Part 2	–	–	–
Participant less likely to try recalling from memory	–	–	✓

Impact of Composition Policy

Moving from a 1class8 to a 3class12 policy increased the time it took to create the password, measured from the first keystroke to the last keystroke in the password box ($p = .014$). Table 7.11 details metrics of the password-creation process. It also impacted participant sentiment about password creation, as shown in Table 7.13. The 3class12 policy led participants to report password creation as significantly more annoying (both $p < .001$) and significantly more difficult ($p < .001$ and $p = .003$, respectively).

Passwords created under a 3class12 policy were more secure than those created under 1class8, but these security gains were somewhat futile because participants were less likely to remember their password. Compared to 1class8, participants who made 3class12 passwords were less likely to successfully recall their password during Part 2 ($p = .025$). Across conditions, 81.3% of 1class8 participants successfully recalled their password during Part 2, whereas 75.0% of 3class12 participants did. The policy did not significantly impact any other recall metrics.

Impact of the Amount of Feedback

Increasing the amount of feedback increased the time it took to create the password ($p = .011$). We observed an interaction between the amount of feedback and the stringency; increasing the amount of feedback in

high-stringency conditions led to a greater time increase ($p = .048$).

To understand how participants change their password during creation, we examined the number of deletions, which we defined as a participant removing characters from their password that they added in a prior keystroke. Increasing amounts of feedback led to significantly more deletions ($p < .001$), implying that the feedback causes participants to change their password-creation decisions. For instance, the median number of deletions for 1c8-None was zero, while the median number for 1c8-Std-H was 9. We observed two significant interaction effects. For high-stringency conditions, an increased amount of feedback led to even more deletions ($p = .002$).

Increasing the amount of feedback negatively affected participant sentiment. It led participants to report password creation as more annoying (both $p < .001$) and more difficult ($p < .001$ and $p = .003$, respectively). It also led participants to report password creation as less fun ($p = .025$). For each sentiment, roughly 10%–15% of the participants in that condition moved from agreement to disagreement, or vice versa.

Even though increasing the amount of feedback led to significantly more secure passwords, it did not significantly impact any of our recall metrics.

Impact of Stringency

Although increasing the scoring stringency led participants to create longer passwords, varying between medium and high stringency did not cause them to take significantly longer to do so, nor did it impact participant sentiment about password creation. When we added an additional low stringency level in Experiment 2, however, participants who saw increased stringency took longer to create a password ($p < .001$) and deleted more characters during creation ($p < .001$). They also took longer to recall their password during Part 1 ($p = 0.010$) and were less likely to try recalling their password solely from memory ($p = .002$), though stringency did not significantly impact other recall metrics.

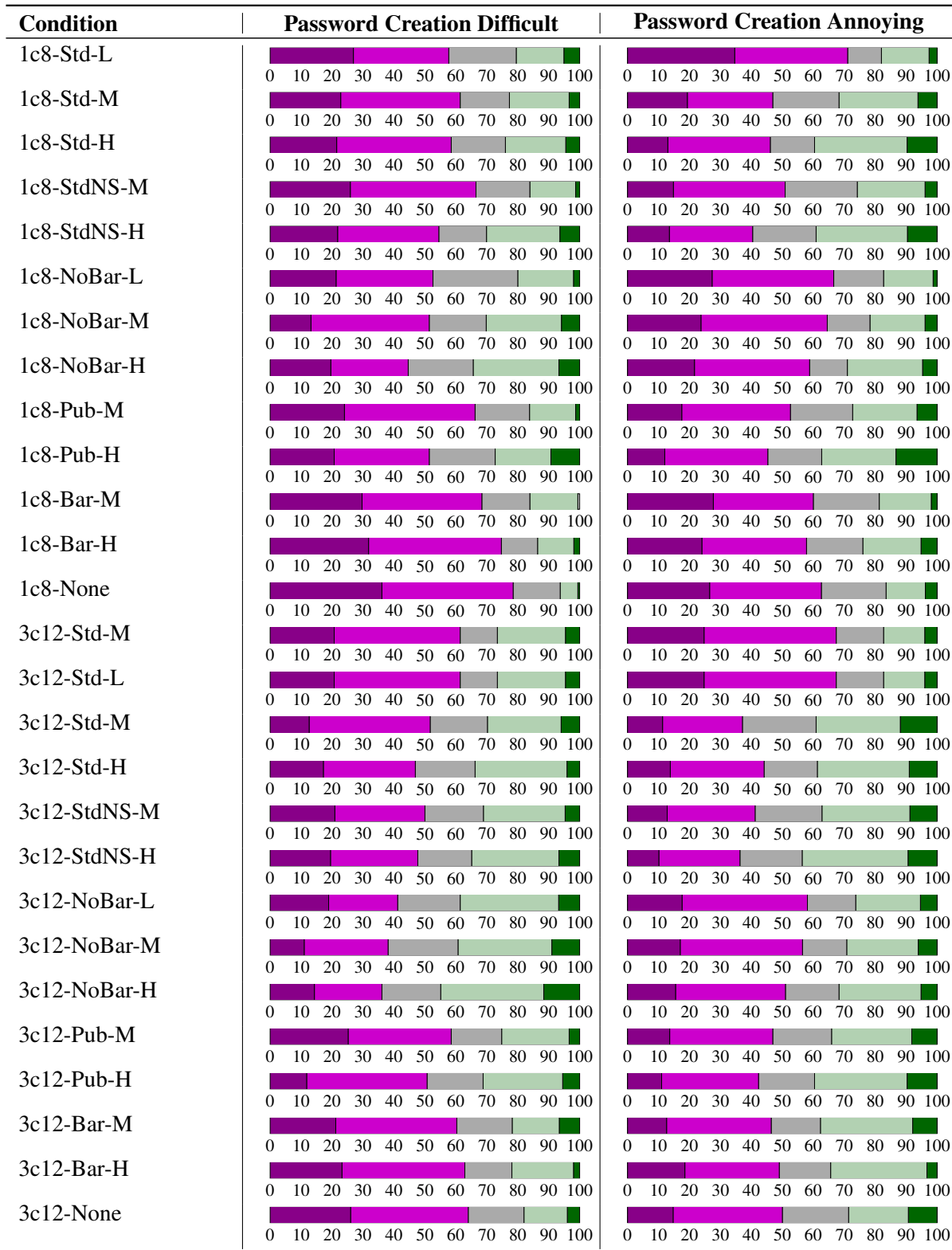
Increasing the stringency greatly impacted participant sentiment. It led participants to perceive password creation as more annoying, more difficult, and less fun ($p < .001$, $p < .001$, $p = .010$, respectively). It also caused participants to be more likely to say the bar helped them create a stronger password ($p = .027$), less likely to believe the bar was accurate ($p < .001$), and less likely to find it important that the bar gives them a high score ($p = .006$). Increasing levels of stringency made participants more likely to say the text feedback led them to create a different password than they would have otherwise ($p = .010$), but also less likely to believe they learned something new from the text feedback ($p < .001$).

7.5.6 Interface Element Usage and Reactions

In this section, we discuss participants' usage of, and reaction to, the different aspects of data-driven text feedback, as well as the colored bar.

Text Feedback

Participants reacted positively to the text feedback, as detailed in Table 7.14. Most participants (61.7%) agreed or strongly agreed that the text feedback made their password stronger. Similarly, 76.9% disagreed or strongly disagreed that the feedback was not informative, and 48.7% agreed or strongly agreed that they created a different password than they would have otherwise because of the text feedback. Higher stringency



Strongly Disagree Disagree Neutral Agree Strongly Agree

Table 7.13: Participants' agreement that creating a password was difficult or annoying.

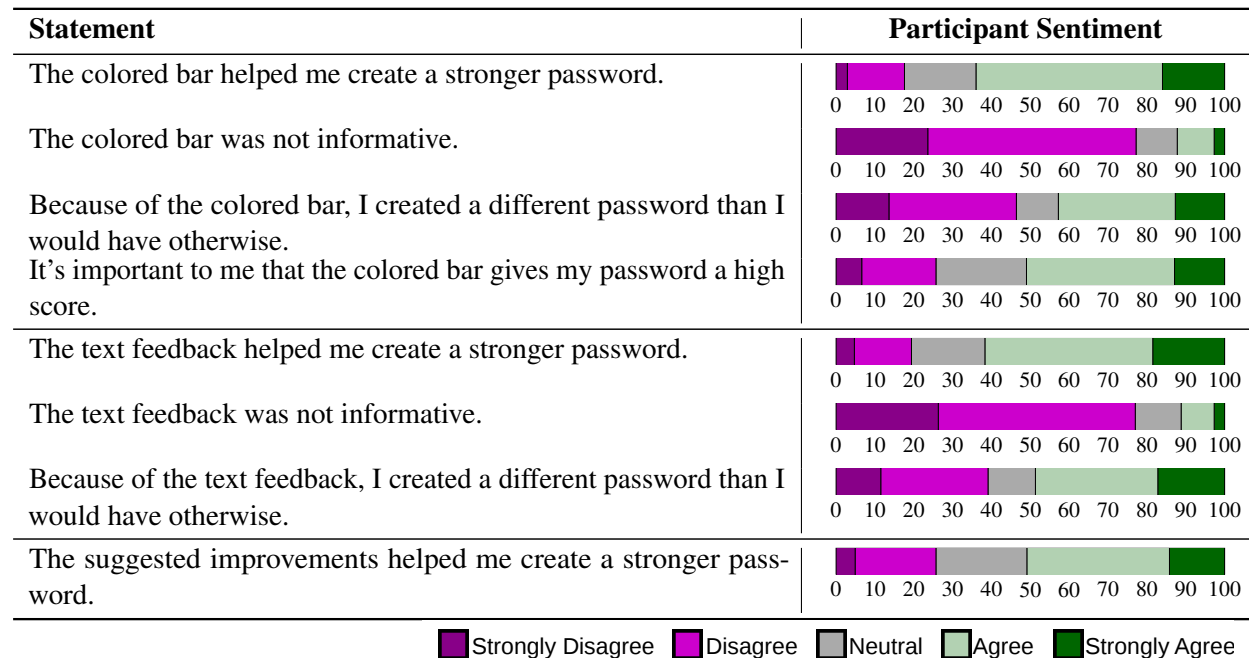


Table 7.14: Participants' agreement with statements about the meter's color bar, text feedback, and suggested improvements.

participants were more likely to say they created a different password ($p = .022$), but no other dimension had a significant impact.

Although most participants (68.5%) selected “no” when we asked if they learned “something new about passwords (your password, or passwords in general) from the text feedback,” 31.5% selected “yes.” Participants commonly said they learned about moving capital letters, digits, and symbols to less predictable locations from the meter (e.g., “I didn't know it was helpful to capitalize an internal letter.”). Many participants also noted that the meter's requests not to use dictionary entries or words from Wikipedia in their password taught them something new. One of these participants noted learning “that hackers use Wikipedia.” Requests to include symbols also resonated. As one participant wrote, “I didn't know previously that you could input symbols into your passwords.”

Participants also took the text feedback as an opportunity for reflection on their password-creation strategies. One participant learned “that I tend to use full words which isn't good,” while another learned “don't base the password off the username.” Participants exercised many of the features of our feedback, including participants who “learned to not use L33T to create a password (exchanging letters for predictable numbers).” Some participants also learned about password reuse, notably that “people steal your passwords in data breaches and they try to use it to access other accounts.”

Suggested Improvement

When participants in applicable conditions showed their password or clicked “see your password with our improvements,” they would see the suggested improvement. Across conditions, 37.8% of participants clicked the “show password” checkbox. Participants who made a 3class12 password, saw a higher-stringency meter,

or who saw less feedback were more likely to show their password on screen ($p < .001$, $p = .006$, and $p = .022$, respectively). While most participants who saw a suggested improvement did so as a consequence of checking “show password & detailed explanations,” 8.7% of participants in applicable conditions specifically clicked the “see your password with our improvements” button. Higher stringency made participants more likely to show the suggested improvement ($p = .003$).

When we asked in the survey whether participants in those conditions had seen a suggested improvement, 34.8% selected “yes,” 55.6% selected “no,” and 9.6% chose the “I don’t remember” option. Nearly all participants who said they did not see a suggested improvement indeed were never shown a suggested improvement because they never showed their password or clicked “see your password with our improvements.” Among participants who said they saw a suggested improvement, 81.5% said that suggested improvements were useful, while 18.5% said they were not. A slight majority (50.9%) of these participants agreed or strongly agreed that the suggested improvement helped them make a stronger password. This help, however, did not often come in the form of adopting the precise suggestion offered. In each condition that offered a suggested improvement, at most 7% of participants used a password that the meter had suggested verbatim.

We asked participants who found the suggested improvements useful to explain why. They wrote that seeing a suggested improvement “helps you modify what you already have instead of having to think of something absolutely new” and “breaks you out of patterns you might have when creating passwords.” Participants particularly liked that the suggested improvement was a modification of their password, rather than an entirely new one, because it “may help spark ideas about tweaking the password versus having to start from scratch.” As one participant summarized, “It actually offers some help instead of just shutting you down by essentially saying ‘no, not good enough, come up with something else.’ It’s very helpful.”

Participants who did not find it useful expressed two streams of reasoning. The first stream concerned memorability. One participant explained, “I’m more likely to forget a password if I don’t use familiar techniques.” while another wrote, “I already have a certain format in mind when I create my password to help me memorize it and I don’t like to stray from that.” The second stream of reasoning concerned the trustworthiness of the “algorithm that creates those suggestions.” As one participant succinctly explained, “I don’t trust it. I don’t want a computer knowing my passwords.”

Modal Windows

Clicking a “(why?)” link next to any of the up to three pieces of feedback in any condition with text feedback opened the specific-advice modal. Few participants in our experiment clicked on “(why?)” links, and therefore few participants saw the specific-advice modal. Only 1.0% of participants across all conditions clicked on one of these links.

In contrast, 8.4% of participants looked at the generic-advice modal, though 3class12 participants were less likely to do so ($p = .025$). Participants could arrive at the generic-advice modal by clicking “how to make strong passwords” or clicking “(why?)” next to admonitions against password reuse. Participants arrived at the generic-advice modal about evenly through these two methods.

Colored Bar

We also analyzed participants’ reactions to the colored bar. All three dimensions impacted how much of the colored bar participants filled. Participants who were assigned the 3class12 policy or saw more feedback filled more of the bar, while participants whose passwords were rated more stringently unsurprisingly filled

less (all $p < .001$). Few participants completely filled the bar (estimated guess numbers 10^{18} and 10^{24} in medium and high stringency, respectively). The median participant often filled half to two-thirds of the bar, depending on the stringency. For instance, for 1c8-Std-M, only 16.5% completely filled the bar, but 51.7% filled at least two-thirds, and 73.1% filled at least half. Table 7.15 details these behaviors by condition.

Overall, participants found the colored bar useful. The majority of participants (64.0%) agreed or strongly agreed that the colored bar helped them create a stronger password, 42.8% agreed or strongly agreed that the bar led them to make a different password than they would have otherwise, and 77.2% disagreed or strongly disagreed with the statement that the colored bar was not informative. Participants also looked to the colored bar for validation; 50.9% of participants agreed or strongly agreed that it is important that the colored bar gives their password a high score. High-stringency participants were less likely to care about receiving a high score ($p = .025$). With increasing amounts of feedback, participants were more likely to care about receiving a high score ($p = .002$), more likely to say that the bar helped them create a stronger password ($p < .001$) that was different than they would have otherwise ($p < .001$). They were also less likely to believe the bar was not informative ($p = .024$).

Participants mostly felt the colored bar accurately scored their password. Across conditions, 68.2% of participants said they felt the bar scored their password's strength accurately, while 23.6% felt the bar gave their password a lower score than it deserved. An additional 4.2% of participants felt the bar gave their password a higher score than it deserved, while 4.0% did not remember how the bar scored their password. Participants were less likely to believe the rating was accurate in the more stringent conditions ($p < .001$).

We also tested removing the colored bar while keeping all text feedback. Removing the colored bar caused participants to be more likely to return for Part 2 of the study ($p = .020$), but did not impact any other objective security or usability metrics. Removing the colored bar did impact participant sentiment, however. Participants who did not see a bar found password creation more annoying and difficult (both $p < .001$).

7.6 Discussion

In this chapter, we described our design and evaluation of a password meter that provides detailed, data-driven feedback. Using a combination of artificial neural networks and nearly two dozen advanced heuristics to score passwords, as well as giving users detailed text explanations of what parts of their particular password is predictable, our meter gives both more accurate and more actionable information to users.

We found that our password-strength meter made 1class8 passwords harder to guess without significantly impacting memorability. Text feedback led to more secure 1class8 passwords than a colored bar alone, whereas colored bars alone are the type of meter widely deployed today [51]. Notably, leaving the detailed text feedback but removing the colored bar did not significantly impact the security of the passwords participants created. Combined with our finding that most people do not feel compelled to fill the bar, this suggests that the visual metaphor has only marginal impact when detailed text feedback is also present. As a result, we highly recommend the use of a meter that provides detailed text feedback for common password-composition policies like 1class8. From our results, we recommend that the meter offer potentially sensitive feedback when the user shows his or her password on screen. The suggested improvement did seem to help some participants, but its overall effect was not strong and some participants did not trust suggestions from a computer. While its inclusion does not seem to hurt, we would consider it optional. Similarly, although the generic-advice modal was visited more than the specific-advice modal, only a fraction of participants looked at it. Because not all users need to learn the basics of making strong passwords and because most

Table 7.15: The degree to which participants “filled” the colored bar that displayed their password strength.

Set	% of Bar Filled			% Participants Who Filled Bar...			
	Mean	σ	Median	1/3rd Full	1/2 Full	2/3rds Full	Completely
1c8-Std-L	83.9	20.0	91.4	96.8	92.9	83.3	38.5
1c8-Std-M	67.2	25.1	67.4	90.3	73.1	51.7	16.6
1c8-Std-H	55.0	23.4	51.3	81.2	53.2	28.6	7.1
1c8-StdNS-M	65.8	24.2	69.2	90.3	71.6	52.9	9.7
1c8-StdNS-H	54.5	24.4	51.9	80.8	51.3	28.8	7.1
1c8-Pub-M	62.6	24.2	61.8	90.9	63.0	44.8	9.1
1c8-Pub-H	54.5	23.7	51.9	76.7	53.3	34.7	5.3
1c8-Bar-M	58.8	22.8	56.9	83.9	61.3	38.1	5.8
1c8-Bar-H	44.5	20.0	41.2	69.5	33.8	14.9	0.6
3c12-Std-L	95.4	10.1	100.0	100.0	99.3	97.3	72.7
3c12-Std-M	80.2	17.6	84.2	98.7	94.0	80.1	17.9
3c12-Std-H	64.7	18.8	62.5	96.6	80.0	37.9	5.5
1c8-StdNS-M	78.5	18.0	80.2	98.0	91.2	80.4	18.2
1c8-StdNS-H	65.6	17.7	64.4	98.7	83.2	43.0	6.0
3c12-Pub-M	78.9	18.3	80.2	99.3	93.9	77.6	19.7
3c12-Pub-H	63.0	17.5	63.1	95.8	77.8	43.8	3.5
3c12-Bar-M	75.8	19.0	77.7	97.4	89.4	70.2	15.2
3c12-Bar-H	61.2	16.2	61.4	96.0	76.2	37.7	1.3

users understand why dictionary words are bad as a major component of a password [181], it is reasonable that only a handful of users would need to engage with these features. We recommend that they be included to help these users.

In contrast to prior work that found scoring stringency to be crucial for password meters [182], we only observed a significant effect for 3class12 passwords, and the effect size was small. Note that our meter used far more advanced methods to score passwords more accurately than the basic heuristics tested in that prior work. Because the high-stringency setting negatively impacted some usability metrics, we recommend our medium setting.

Our recommendations differ for 3class12 passwords. The meter had minimal impact on the security of 3class12 passwords. While the meter introduced few usability disadvantages, suggesting that it may not hurt to include the meter, we would not recommend it nearly as strongly as for 1class8 passwords.

To spur adoption of data-driven password meters, we are releasing our meter’s code open-source.⁴

⁴Available at https://github.com/cupslab/password_meter

Chapter 8

Conclusion and Future Work

8.1 Conclusions and Lessons Learned

In this thesis, I have used data-driven methods to support users' password-creation decisions. I began by discussing our large-scale, online study of password-strength meters (Chapter 3). We found that meters that score passwords using even basic heuristics can improve password security without negatively impacting usability, as long as those meters show password strength visually and score passwords more stringently than the meters used in the wild at that time. While this first step showed that users can successfully be nudged towards better passwords by well-designed interventions, the very minimal feedback we gave users in this early part only vaguely reflected how people create passwords and how attackers try to guess passwords.

Moving towards building better interventions, I spent the subsequent chapters of the thesis delving into the techniques of attackers and the habits of users through large-scale data analysis. Through extensive experiments, we modeled how attackers would guess different sets of passwords using widely used conceptual approaches to guessing passwords, each in a number of configurations (Chapter 4). We then compared these models to the performance of an expert password-forensics firm we contracted. While the password-guessing models used previously in the literature far underestimated the vulnerability of passwords to guessing by a skilled professional, we proposed methods for considering multiple well-configured approaches in parallel to model adversarial guessability, finding this method to be a relatively conservative proxy for guessing by a skilled professional.

To better understand users, we first used crowdsourcing to manually reverse engineer passwords, undoing substitutions and transformations, as well as separating passwords into their component parts (Chapter 5). We used these reverse-engineered passwords to document the predictability of different behaviors in how users structure passwords, incorporate semantically meaningful content, and approach the password-creation process. We then sought to better understand users by studying how their perceptions of password security matched our models of password-guessing techniques (Chapter 6). We found that users' perceptions of what individual password characteristics make a password more secure mostly matched our guessing models, yet participants' perceptions of attackers differed greatly from one person to another. As a result, many participants believed passwords that were likely insecure in reality would be strong enough to resist an attack.

In the final section of my thesis (Chapter 7), I described the iterative design and two-part evaluation of our data-driven password-strength meter. This meter integrates our previous insights about how attackers guess passwords by using two complementary ways to measure a password's strength. It also gives users detailed

text feedback about how to improve their specific password, in addition to proposing concrete modifications to their password as a suggested improvement. We found that this meter led participants to create stronger passwords without being significantly less likely to remember their password. In addition, most of our participants reported that the text feedback was informative and helped them create stronger passwords.

Below, I discuss key insights about passwords and usable security that became clear over the course of this thesis.

8.1.1 Users' Well-Ensconced Approaches

Across the user studies discussed in this thesis, in addition to a companion laboratory study of password creation [184], I learned from myriad users about their password-creation strategies. Various users told me about numerous different methods they had developed for making strong passwords, and these methods combined professional security advice they had learned at work, folk wisdom they had learned informally, and their own intuition and creativity. Participants described schemes they had developed and that they believed to be secure. Some unfortunately were insecure (e.g., using “temp1234” for all accounts), others appeared relatively secure (e.g., substantially mangling the names of minor characters from obscure mythological texts that the participant associated with the purpose of the account), and others mirrored the advice of security experts (e.g., making up unique sentences and mangling them).

A key takeaway from synthesizing all of these stories and strategies is that many users have well-developed, albeit not always secure, methods of coping with passwords. Having likely made hundreds of passwords in their life [70], a user has much experience making and coping with [169] passwords, although not necessarily much feedback on their strategy. In essence, users could potentially learn that their password-creation and password-management strategies are poorly thought out if they suffer a data breach, but they will not necessarily come to such a conclusion. Any ad-hoc, post-mortem investigation of “being hacked” will likely not elucidate that the account was compromised, for instance, because the user reused his or her password on a different site that suffered a data breach. Similarly, for all but the largest breaches, users would likely not learn that, against best-practice advice, the service that was breached hashed passwords insecurely using MD5. In contrast, users who have “never been hacked” despite adopting poor security practices or widely reusing passwords can mistakenly assume that their system is working. In some ways, their system may be working, although the risk of a future compromise is high.

8.1.2 The Mismatched Incentives of Professional Password Advice

To this point, much of the information users have learned about passwords is from their own experiences creating passwords for accounts. Unfortunately, mismatched incentives often prevent users from receiving reasonable advice about the entire ecosystem, leading to their rational rejection of security advice [90]. For instance, users are required to create accounts for many systems where the value of the account is minimal or nonexistent to the user (e.g., making an account to read a newspaper article).

System administrators are incentivized, however, to demand that users make complex and unique passwords for their own system. In the case of a data breach, a system administrator who let users know that it is acceptable to make predictable passwords and reuse them across meaningless accounts would likely be blamed. Furthermore, system administrators recommend that users take steps that are based on outdated assumptions, and some of these steps (e.g., regularly changing their password) may be detrimental to security in practice [41, 205].

The litany of onerous requirements related to passwords is an opportunity to shift blame to the user [3] for not following practices that are mostly impossible in practice. In its place, user-centered guidance for reasonably coping with the authentication ecosystem is necessary. I believe the meter I discussed in Chapter 7 to be a first step in that direction.

8.1.3 Mismatch Between Reality and Perception

In the course of making potentially hundreds of passwords in their lives, the participants in our studies had spent substantial time engaging with password creation. While we found that a number of their perceptions of how different password characteristics relate to password security matched reality (Chapter 6), they held important misconceptions that led to poor security in practice [181, 184]. The importance of matching users' perceptions to the reality of a security or privacy ecosystem is therefore a major takeaway from the work in this thesis.

Many of users' misconceptions about password security seem to stem from their prior interactions with security tools and interfaces. Current tools to help users create better passwords are lacking. They rate weak passwords as strong, and vice versa [51], giving users false confidence in the security of some passwords that are actually predictable. These meters also provide feedback that is not very helpful and may even lead to misconceptions about security [184]. Even though password reuse is a major security threat [49], these tools often gloss over such ecosystem-level concerns. As a result, users are left with unusable security.

In the case of helping users create passwords, I found that iteratively designing a meter using data-driven methods, as well as presenting data-driven, specific feedback to users, can help them create better passwords (Chapter 7). This data-driven approach can potentially be extended to many other areas of usable security and privacy, such as helping users make online privacy decisions.

8.1.4 Real-World Considerations in Modeling Attackers

In the process of modeling how attackers guess passwords (Chapter 4), we learned important lessons about the importance of investigating not just how attackers behave, but why they do so. Using the method standard in academic analyses, modeling a password's security by counting the number of guesses the software tool took to guess it, we found that Hashcat's wordlist mode appeared to lag behind other password-guessing approaches on our graphs. This apparent lag is caused by Hashcat's ordering of guesses, as well as its lack of support for filtering a guess based on non-compliance with a particular password-composition policy.

Despite what these graphs show, Hashcat is very widely used in practice, and we would highly recommend it to someone attempting to crack passwords. Understanding why requires taking a step back to consider the overall ecosystem. In many cases, the percentage of passwords cracked at the end of an attacker's cracking session, rather than how quickly each was cracked, matters most. Statistical approaches that produce an optimal ordering of guesses according to their model of how passwords are chosen are, compared to Hashcat, extremely slow in generating guesses. Because this guess-generation process can be a major time factor in an offline attack that also involves hashing candidate guesses, Hashcat's approach of very quickly generating guesses, some of which are non-optimal or even impossible, can result in a more successful attack overall.

Therefore, when modeling security, it is important to consider both what an optimal attacker might do, as well as what attackers are likely to do in practice given the availability of certain types of data, economics, and rational coping mechanisms, among other factors. Accounting for both scenarios is key to producing more accurate estimates of security.

8.2 Future Work

In this section, we outline promising directions for research to improve practical authentication for users, particularly in the near-term future.

8.2.1 Improve the Ecosystem

On the one hand, an overall ecosystem in which users memorize hundreds of distinct, complex passwords is unsustainable [70]. On the other hand, despite regular claims over the last decade that the password is dead [113, 131], it is unlikely that passwords will entirely disappear in the foreseeable future because of their advantages over alternative authentication methods [24]. In essence, rather than killing the password, it may be worth re-envisioning the role of the password in authentication. Passwords are currently being asked to do more than they are capable of [161].

Using a single-sign-on system, using a password manager, and enabling two-factor authentication have the potential to greatly improve practical authentication security while retaining many of the benefits of passwords. As we outline below, though, both infrastructural and usability questions remain about making these systems fully usable and fully practical.

Infrastructure

Infrastructural changes could mitigate current issues with single-sign-on systems and password managers. In a study of the OpenID single-sign-on system, researchers found that users lack trust in such systems because they are a single point of failure and can reduce the user's privacy when used to access accounts with personally sensitive information [173]. Research into secure, yet privacy-preserving, single-sign-on systems is therefore crucial in spurring adoption of such systems. Communicating intelligibly to users about the privacy properties of such systems, which are likely to be complex, is an additional challenge.

Password managers could also be designed in a principled way. Currently, an exploit targeting a password manager could release all of a user's passwords, and a software crash without commensurate backups could lock a user out of his or her accounts. One could imagine rigorously designing a password manager using state-of-the-art techniques in software engineering to minimize the software's trusted computing base and prove correct important properties about its operation. In concert with rigorous usability testing, such an approach could enable widespread adoption of password managers. Furthermore, with a more centralized authentication ecosystem based on single-sign-on systems or password managers, two-factor authentication would become even more valuable in helping users protect a larger number of accounts.

User Behavior and Perceptions

While building the infrastructure to support secure, yet practically private, authentication throughout an ecosystem is a crucial first step, understanding how this ecosystem will impact subsequent user behavior is equally important. For instance, if a user relies on a single-sign-on system for all of his or her accounts, the security of that single-sign-on password is crucial. If users are not sufficiently supported to understand the importance of this password, no amount of engineering will improve security throughout the ecosystem.

8.2.2 Evaluating The Role of Password-Composition Policies

Onerous password-composition policies have traditionally been employed to push users towards stronger passwords [156]. These policies require that passwords contain certain characteristics in the hopes that, in the process of including these, users make hard-to-predict choices. Password-composition policies are thus an indirect, unprincipled means of pushing users towards what are hoped to be stronger passwords. Researchers have argued in the past that password-composition policies may not be the ideal method for thinking about password strength; they instead argued that restricting the popularity of a password on a particular system would be a better conception [149].

One might instead want to decide whether to allow a password based on how predictable it is. Doing so has historically involved multiple challenges, including questions about how well models correspond to passwords' predictability to attackers and how one could compactly encode these models on the client's machine. While using principled models of password guessing [38] or password selection [107] had been proposed previously, these models could not be transferred to the client's machine.

Recent advances in password-strength estimation entirely on the client side [130, 199] and our comparison of our guessing models to professionals (Chapter 4), however, suggest that estimating the predictability of a password on the client side, without enforcing any password-composition policy, is a possible way forward. We found in Chapter 7 that our standard password-strength meter matched with a 1class8 password-composition policy can lead users to make passwords that are somewhat similar in strength to those created under a 3class12 policy, though introducing the meter introduced many of the same usability disadvantages of a 3class12 policy. In light of these results, I believe further evaluating the role of password-composition policies vis-à-vis advanced client-side password checking is worth future investigation.

8.2.3 Natural Language and Passwords

One of our goals in Chapter 5 was to understand the role of semantic content, including words and phrases, in passwords by modeling substrings from the passwords using natural-language corpora. We found fairly wide use of words and phrases from natural language in passwords. Furthermore, prior work found the inclusion of natural-language data to improve password guessing [104, 197]. Surprisingly, our recent work on using neural networks to guess passwords found that including natural language data actually made guessing less accurate [130]. Further investigation could leverage our reverse-engineered passwords from Chapter 5 to better understand whether natural-language corpora could also improve guessing by neural networks [130], potentially by adding an additional layer to the neural network.

8.2.4 Automatically Modeling Targeted Attacks

In Chapter 4, we modeled adversarial password guessing in a large-scale, trawling attack. One of the limitations of this approach, though, is that it does not consider attacks targeted to a particular individual. As discussed in Chapter 2, users often include meaningful names and dates, in addition to things they like, in their passwords. While tools like Wordhound [81] from the password-cracking community attempt to create a user-specific or site-specific wordlist by scraping the internet based on keywords, our models of password guessing could be improved by better quantifying and including the extent to which password guessing can be targeted to an individual. Such a model of password guessing that automatically evaluates the extent to which password guessing could be targeted to a particular password from a particular individual would be

particularly helpful for removing a blind spot in current proactive password checking. This model potentially could also take into account the degree to which that user has reused a variant of a candidate password, as well as for which accounts, in estimating security.

8.2.5 Improvements to the Meter

The codebase for the meter described in Chapter 7 is already stable despite its complexity in adding many new features beyond current password-strength meters. However, the additional features and improvements listed below would support wider adoption of this meter.

Accessibility

We have taken initial steps in testing our meter with screen readers and labeling graphical components, yet the meter would greatly benefit from further accessibility enhancements. These enhancements would include adding explanatory text exclusively to assist users accessing the meter via screen readers, testing of color shades to better support color-blind users, and easy ways to increase the size of text and buttons for users with difficulty seeing.

Compression

While we have been cognizant throughout the design process of keeping the file sizes small for our data-driven meter, a few megabytes of data must be transferred to clients. While our meter is still smaller in size than many modern webpages, one could take additional steps to further reduce the meter's size. For instance, some of the dictionaries and wordlists we package with the meter could potentially be reduced in size by automatically accounting for the similarity among entries, instead bundling smaller wordlists with more advanced transformations (similar to mangling rules). This approach, however, would necessitate additional client-side computation, which may impede performance.

Automatic Updates

We designed our meter to reflect the patterns in widely distributed password sets at the time we created it. However, as the way users pick passwords changes over time in response to additional education, requirements, and new cultural references, our meter could benefit from a system for automatically updating itself. One could imagine the meter updating its wordlists and weightings of different characteristics when fed recent password breaches and potentially scrapes of news websites to capture pop-culture references. Such an approach would involve substantial additional architecture. One could also imagine enabling the meter to automatically customize itself given an organization's password corpus, although the security risks of potentially leaking information about the organization's password corpus likely outweigh the benefits of having more accurate scoring.

Additional Caching

Currently, the meter caches the scoring and feedback for passwords the user has typed in during a particular section to improve performance. If the user deletes a character, for instance, the meter does not recompute password strength. However, in the common case of a user adding additional characters to their existing

candidate password, individual meter functions (e.g., dictionary lookups) do not currently rely on cached data. Results of looking up a password's substrings or evaluating other characteristics could potentially be cached, though using more memory to save computation may not necessarily be beneficial for overall performance and responsiveness.

Balancing Browser Support and Multi-Thread Support

To improve the responsiveness of the user interface, our meter scores passwords using neural networks [130] in a separate thread using the WebWorker framework [133]. We have considered also moving heuristic scoring to a separate thread, but doing so introduces a complex tradeoff. The WebWorker framework and other techniques of writing multi-threaded code in the browser are not fully supported by all platforms. On these platforms, the lack of WebWorker support is not completely problematic because we can still score passwords using advanced heuristics. Moving forward, there remain open questions about how best to balance having the meter work on as many platforms as possible, yet doing so as efficiently as possible. One possible solution would be to perform multi-threaded computation only if the meter detects in a session that the browser fully supports it.

Bibliography

- [1] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. A comparison of machine learning techniques for phishing detection. In *Proc. APWG eCrime Researchers Summit*, 2007.
- [2] Steven Van Acker, Daniel Hausknecht, Wouter Joosen, and Andrei Sabelfeld. Password meters and generators on the web: From large-scale empirical study to getting it right. In *Proc. CODASPY*, 2015.
- [3] Anne Adams and Martina Angela Sasse. Users are not the enemy. *CACM*, 42(12):40–46, 1999.
- [4] Anne Adams, Martina Angela Sasse, and Peter Lunt. Making passwords secure and usable. In *Proc. HCI on People and Computers*, 1997.
- [5] Alexa.com. The top 500 sites on the web. <http://www.alexa.com/topsites>, 2016.
- [6] Hazim Almuhiemedi, Florian Schaub, Norman Sadeh, Idris Adjerid, Alessandro Acquisti, Joshua Gluck, Lorrie Cranor, and Yuvraj Agarwal. Your location has been shared 5,398 times! A field study on mobile app privacy nudging. In *Proc. CHI*, 2015.
- [7] Julio Angulo, Simone Fischer-Hübner, Tobias Pulls, and Erik Wästlund. Usable transparency with the data track: A tool for visualizing data disclosures. In *Proc. CHI Extended Abstracts*, 2015.
- [8] Farzaneh Asgharpour, Debin Lu, and L. Jean Camp. Mental models of computer security risks. In *Proc. WEIS*, 2007.
- [9] Adam J. Aviv and Dane Fichter. Understanding visual perceptions of usability and security of Android’s graphical password pattern. In *Proc. ACSAC*, 2014.
- [10] Dylan Ayrey. HashcatJS. <https://github.com/praetorian-inc/hashcatJS>, 2016.
- [11] Rebecca Balebako, Jaeyeon Jung, Wei Lu, Lorrie Faith Cranor, and Carolyn Nguyen. “Little brothers watching you:” Raising awareness of data leaks on smartphones. In *Proc. SOUPS*, 2013.
- [12] Chris Baraniuk. Ashley Madison: Two women explain how hack changed their lives. *BBC* <http://www.bbc.co.uk/news/technology-34072762>, August 27, 2015.
- [13] Bob Beeman. Using “grep” (a Unix utility) for solving crosswords and word puzzle. <http://www.bee-man.us/computer/grep/grep.htm#web2>, 2004, retrieved November 2010.
- [14] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B*, 57(1):289–300, 1995.

- [15] Francesco Bergadano, Bruno Crispo, and Giancarlo Ruffo. Proactive password checking with decision trees. In *Proc. CCS*, 1997.
- [16] André Bergholz, Gerhard Paa, Frank Reichartz, Siehyun Strobel, and Jeong-Ho Chang. Improved phishing detection using model-based features. In *Proc. CEAS*, 2008.
- [17] Adam J. Berinsky, Gregory A. Huber, and Gabriel S. Lenz. Evaluating online labor markets for experimental research: Amazon.com’s Mechanical Turk. *Political Analysis*, 20:351–368, 2012.
- [18] Chandrasekhar Bhagavatula, Blase Ur, Kevin Iacovino, Su Mon Kywe, Lorrie Faith Cranor, and Marios Savvides. Biometric authentication on iPhone and Android: Usability, perceptions, and influences on adoption. In *Proc. USEC*, 2015.
- [19] Alex Biryukov, Daniel Dinu, , and Dmitry Khovratovich. Version 1.2 of Argon2. <https://password-hashing.net/submissions/specs/Argon-v3.pdf>, July 8, 2015.
- [20] Matt Bishop and Daniel V. Klein. Improving system security via proactive password checking. *Computers & Security*, 14(3):233–249, 1995.
- [21] Joseph Bonneau. The Gawker hack: How a million passwords were lost. *Light Blue Touchpaper Blog*, December 2010. <http://www.lightbluetouchpaper.org/2010/12/15/the-gawker-hack-how-a-million-passwords-were-lost/>.
- [22] Joseph Bonneau. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *Proc. IEEE SP*, 2012.
- [23] Joseph Bonneau. Statistical metrics for individual password strength. In *Proc. WSP*, 2012.
- [24] 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 *Proc. IEEE SP*, 2012.
- [25] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. Passwords and the evolution of imperfect authentication. *CACM*, 58(7):78–87, June 2015.
- [26] Joseph Bonneau, Mike Just, and Greg Matthews. What’s in a name? Evaluating statistical attacks on personal knowledge questions. In *Proc. FC*, 2010.
- [27] Joseph Bonneau and Sören Preibusch. The password thicket: technical and market failures in human authentication on the web. In *Proc. WEIS*, 2010.
- [28] Joseph Bonneau and Stuart Schechter. Towards reliable storage of 56-bit secrets in human memory. In *Proc. USENIX Security*, 2014.
- [29] Joseph Bonneau and Ekaterina Shutova. Linguistic properties of multi-word passphrases. In *Proc. USEC*, 2012.
- [30] Joseph Bonneau and Rubin Xu. Of contraseñas, sysmawt, and mímă: Character encoding issues for web passwords. In *Proc. W2SP*, 2012.

- [31] Jon Brodtkin. 10 (or so) of the worst passwords exposed by the LinkedIn hack. *Ars Technica*, June 2012.
- [32] Kay Bryant and John Campbell. User behaviours associated with password security and management. *Australasian Journal of Information Systems*, 14(1), 2006.
- [33] Michael Buhrmester, Tracy Kwang, and Samuel D. Gosling. Amazon’s Mechanical Turk: A new source of inexpensive, yet high-quality, data? *Perspectives on Psychological Science*, 6(1):3–5, 2011.
- [34] Mark Burnett. Today I am releasing ten million passwords. <https://xato.net/today-i-am-releasing-ten-million-passwords-b6278bbe7495#.s11zbdb8q>, February 9, 2015.
- [35] William E. Burr, Donna F. Dodson, and W. Timothy Polk. Electronic authentication guideline. Technical report, NIST, 2006.
- [36] Dell Cameron. Apple knew of iCloud security hole 6 months before Celebgate. *The Daily Dot*, September 24 2014. <http://www.dailydot.com/technology/apple-icloud-brute-force-attack-march/>.
- [37] Carnegie Mellon University. Password guessability service. <https://pgs.ece.cmu.edu>, 2015.
- [38] Claude Castelluccia, Markus Dürmuth, and Daniele Perito. Adaptive password-strength meters from Markov models. In *Proc. NDSS*, 2012.
- [39] Jon M. Chang. Passwords and email addresses leaked in Kickstarter hack attack. *ABC News*, Feb 17, 2014. <http://abcnews.go.com/Technology/passwords-email-addresses-leaked-kickstarter-hack/story?id=22553952>.
- [40] Sonia Chiasson, Alain Forget, Elizabeth Stobert, P. C. van Oorschot, and Robert Biddle. Multiple password interference in text passwords and click-based graphical passwords. In *Proc. CCS*, 2009.
- [41] Sonia Chiasson and Paul C van Oorschot. Quantifying the security advantage of password expiration policies. *Designs, Codes and Cryptography*, 77(2):401–408, 2015.
- [42] Hsien-Cheng Chou, Hung-Chang Lee and Hwan Jeu Yu, Fei-Pei Lai, Kuo-Hsuan Huang, and Chih-Wen Hsueh. Password cracking based on learned patterns from disclosed passwords. *International Journal of Innovative Computing, Information and Control*, 2013.
- [43] Yiannis Chrysanthou. Modern password cracking: A hands-on approach to creating an optimised and versatile attack. Master’s thesis, Royal Holloway, University of London, 2013.
- [44] Jan De Clercq. Resetting the password of the KRBTGT active directory account, 2014. <http://windowsitpro.com/security/resetting-password-krbtgt-active-directory-account>.
- [45] Frederick G. Conrad, Mick P. Couper, Roger Tourangeau, and Andy Peytchev. The impact of progress indicators on task completion. *Interacting with Computers*, 22(5):417–427, 2010.

- [46] Sunny Consolvo, Jaeyeon Jung, Ben Greenstein, Pauline Powledge, Gabriel Maganis, and Daniel Avrahami. The wi-fi privacy ticker: improving awareness & control of personal information exposure on wi-fi. In *Proc. Ubicomp*, 2010.
- [47] Lorrie Faith Cranor, Pedro Giovanni Leon, and Blase Ur. A large-scale evaluation of U.S. financial institutions' standardized privacy notices. *ACM TWEB*, 10(3):17, 2016.
- [48] curlyboi. Hashtopus. <http://hashtopus.nech.me/manual.html>, 2009-.
- [49] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and XiaoFeng Wang. The tangled web of password reuse. In *Proc. NDSS*, 2014.
- [50] Mark Davies. The corpus of contemporary American English: 425 million words, 1990–present. <http://corpus.byu.edu/coca/>, 2008.
- [51] Xavier de Carné de Carnavalet and Mohammad Mannan. From very weak to very strong: Analyzing password-strength meters. In *Proc. NDSS*, 2014.
- [52] Alexander De Luca, Alina Hang, Emanuel von Zezschwitz, and Heinrich Hussmann. I feel like I'm taking selfies all day! Towards understanding biometric authentication on smartphones. In *Proc. CHI*, 2015.
- [53] Matteo Dell'Amico and Maurizio Filippone. Monte Carlo strength evaluation: Fast and reliable password checking. In *Proc. CCS*, 2015.
- [54] Matteo Dell'Amico, Pietro Michiardi, and Yves Roudier. Password strength: An empirical analysis. In *Proc. INFOCOM*, 2010.
- [55] Jega Anish Dev. Usage of botnets for high speed md5 hash cracking. In *Proc. INTECH*, 2013.
- [56] Martin M. A. Devillers. *Analyzing Password Strength*. PhD thesis, Radboud University Nijmegen, 2010.
- [57] Julie S. Downs, Mandy B. Holbrook, Steve Sheng, and Lorrie Faith Cranor. Are your participants gaming the system? Screening Mechanical Turk workers. In *Proc. CHI*, 2010.
- [58] Chris Duckett. Login duplication allows 20m Alibaba accounts to be attacked. *ZDNet*, February 5, 2016. <http://www.zdnet.com/article/login-duplication-allows-20m-alibaba-accounts-to-be-attacked/>.
- [59] Geoffrey B. Duggan, Hilary Johnson, and Beate Grawemeyer. Rational security: Modelling everyday password use. *International Journal of Human-Computer Studies*, 70(6):415 – 431, 2012.
- [60] Markus Dürmuth, Fabian Angelstorf, Claude Castelluccia, Daniele Perito, and Abdelberi Chaabane. OMEN: Faster password guessing using an ordered markov enumerator. In *Proc. ESSoS*, 2015.
- [61] Markus Dürmuth, Abdelberi Chaabane, Daniele Perito, and Claude Castelluccia. When privacy meets security: Leveraging personal information for password cracking. *CoRR*, 2013. <http://arxiv.org/pdf/1304.6584.pdf>.

- [62] David Eargle, John Godfrey, Hsin Miao, Scott Stevenson, Richard Shay, Blase Ur, and Lorrie Cranor. You can do better — motivational statements in password-meter feedback. *Proc. SOUPS Posters*, 2015.
- [63] Serge Egelman, Lorrie Faith Cranor, and Abdur Chowdhury. An analysis of p3p-enabled web sites among top-20 search results. In *Proc. ICEC*, 2006.
- [64] Serge Egelman, Marian Harbach, and Eyal Peer. Behavior ever follows intention?: A validation of the security behavior intentions scale (SeBIS). In *Proc. CHI*, 2016.
- [65] 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 *Proc. CHI*, 2013.
- [66] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In *Proc. OSDI*, 2010.
- [67] Sascha Fahl, Marian Harbach, Yasemin Acar, and Matthew Smith. On the ecological validity of a password study. In *Proc. SOUPS*, 2013.
- [68] Dinei Florêncio and Cormac Herley. A large-scale study of web password habits. In *Proc. WWW*, 2007.
- [69] Dinei Florêncio, Cormac Herley, and Paul C. van Oorschot. An administrator’s guide to internet password research. In *Proc. USENIX LISA*, 2014.
- [70] Dinei Florêncio, Cormac Herley, and Paul C. van Oorschot. Password portfolios and the finite-effort user: Sustainably managing large numbers of accounts. In *Proc. USENIX Security*, 2014.
- [71] Alain Forget, Sonia Chiasson, P. C. van Oorschot, and Robert Biddle. Improving text passwords through persuasion. In *Proc. SOUPS*, 2008.
- [72] John Fox and Sanford Weisberg. *An R Companion to Applied Regression (Online Appendix)*. Sage Publications, second edition, 2011. <https://socserv.socsci.mcmaster.ca/jfox/Books/Companion/appendix/Appendix-Cox-Regression.pdf>.
- [73] Simson Garfinkel and Heather Richter Lipford. Usable security: History, themes, and challenges. *Synthesis Lectures on Information Security, Privacy, and Trust*, 2014.
- [74] Shirley Gaw and Edward W. Felten. Password management strategies for online accounts. In *Proc. SOUPS*, 2006.
- [75] Megan Geuss. Mozilla: Data stolen from hacked bug database was used to attack Firefox. *Ars Technica* <http://arstechnica.com/security/2015/09/mozilla-data-stolen-from-hacked-bug-database-was-used-to-attack-firefox/>, September 4, 2015.
- [76] Jeffrey Goldberg. Defining password strength. In *Proc. Passwords*, 2013.

- [77] Dan Goodin. Hackers expose 453,000 credentials allegedly taken from Yahoo service. *Ars Technica*, July 2012. <http://arstechnica.com/security/2012/07/yahoo-service-hacked/>.
- [78] Dan Goodin. Why passwords have never been weaker—and crackers have never been stronger. *Ars Technica*, August 2012. <http://arstechnica.com/security/2012/08/passwords-under-assault/>.
- [79] Dan Goodin. Anatomy of a hack: How crackers ransack passwords like “qeadzcvrsfxv1331”. *Ars Technica*, May 2013. <http://arstechnica.com/security/2013/05/how-crackers-make-minced-meat-out-of-your-passwords/>.
- [80] Dan Goodin. “thereisnofatebutwhatwemake”—turbo-charged cracking comes to long passwords. *Ars Technica*, August 2013. <http://arstechnica.com/security/2013/08/thereisnofatebutwhatwemake-turbo-charged-cracking-comes-to-long-passwords/>.
- [81] Dan Goodin. Meet wordhound, the tool that puts a personal touch on password cracking. *Ars Technica*, August 2014.
- [82] Dan Goodin. Once seen as bulletproof, 11 million+ Ashley Madison passwords already cracked. *Ars Technica* <http://arstechnica.com/security/2015/09/once-seen-as-bulletproof-11-million-ashley-madison-passwords-already-cracked/>, September 10, 2015.
- [83] Google. Web 1T 5-gram version 1, 2006. <http://www ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T13>.
- [84] Beate Grawemeyer and Hilary Johnson. Using and managing multiple passwords: A week to a view. *Interacting with Computers*, 23(3), June 2011.
- [85] Haible. gperf. <https://www.gnu.org/software/gperf/>, 2010-.
- [86] S.M. Taiabul Haque, Matthew Wright, and Shannon Scielzo. A study of user password strategy for multiple accounts. In *Proc. CODASPY*, 2013.
- [87] Marian Harbach, Markus Hettig, Susanne Weber, and Matthew Smith. Using personal examples to improve risk communication for security & privacy decisions. In *Proc. CHI*, 2014.
- [88] Eiji Hayashi, Nicolas Christin, Rachna Dhamija, and Adrian Perrig. Use your illusion: Secure authentication usable anywhere. In *Proc. SOUPS*, 2008.
- [89] Alan Henry. Five best password managers. *LifeHacker*, January 11, 2015. <http://lifelifehacker.com/5529133/>.
- [90] Cormac Herley. So long, and no thanks for the externalities: The rational rejection of security advice by users. In *Proc. NSPW*, 2009.
- [91] Shiva Houshmand, Sudhir Aggarwal, and Randy Flood. Next gen PCFG password cracking. *IEEE TIFS*, 10(8):1776–1791, Aug 2015.

- [92] Jun Ho Huh, Seongyeol Oh, Hyounghshick Kim, Konstantin Beznosov, Apurva Mohan, and S. Raj Rajagopalan. Surpass: System-initiated user-replaceable passwords. In *Proc. CCS*, 2015.
- [93] Troy Hunt. The science of password selection. Blog Post, July 2011. <http://www.troyhunt.com/2011/07/science-of-password-selection.html>.
- [94] Imperva. Consumer password worst practices, 2010. http://www.imperva.com/docs/WP_Consumer_Password_Worst_Practices.pdf.
- [95] Philip Inglesant and M. Angela Sasse. The true cost of unusable password policies: Password use in the wild. In *Proc. CHI*, 2010.
- [96] InsidePro. PasswordsPro. <http://www.insidepro.com/eng/passwordspro.shtml>, 2003-.
- [97] Iulia Ion, Marc Langheinrich, Ponnurangam Kumaraguru, and Srdjan Čapkun. Influence of user perception, security needs, and social factors on device pairing method choices. In *Proc. SOUPS*, 2010.
- [98] Iulia Ion, Rob Reeder, and Sunny Consolvo. “. . .no one can hack my mind”: Comparing expert and non-expert security practices. In *Proc. SOUPS*, 2015.
- [99] Panagiotis G. Ipeirotis. Demographics of Mechanical Turk. Technical report, New York University, 2010.
- [100] Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. Quality management on Amazon Mechanical Turk. In *Proc. HCOMP*, New York, NY, USA, 2010.
- [101] Blake Ives, Kenneth R. Walsh, and Helmut Schneider. The domino effect of password reuse. *CACM*, 47(4):75–78, April 2004.
- [102] Markus Jakobsson and Mayank Dhiman. The benefits of understanding passwords. In *Proc. HotSec*, 2012.
- [103] Ari Juels and Ronald L. Rivest. Honeywords: Making password-cracking detectable. In *Proc. CCS*, 2013.
- [104] Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Richard Shay, Tim 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 *Proc. IEEE SP*, 2012.
- [105] Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with Mechanical Turk. In *Proc. CHI*, 2008.
- [106] Saranga Komanduri. *Modeling the adversary to evaluate password strength with limited samples*. PhD thesis, Carnegie Mellon University, 2015.
- [107] Saranga Komanduri, Richard Shay, Lorrie Faith Cranor, Cormac Herley, and Stuart Schechter. Telepath-words: Preventing weak passwords by reading users’ minds. In *Proc. USENIX Security*, 2014.

- [108] Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L. Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Serge Egelman. Of passwords and people: Measuring the effect of password-composition policies. In *Proc. CHI*, 2011.
- [109] KoreLogic. "Crack Me If You Can" - DEFCON 2013. <http://contest-2013.korelogic.com, 2010->.
- [110] KoreLogic. "Crack Me If You Can" - DEFCON 2010. <http://contest-2010.korelogic.com/rules.html, 2010>.
- [111] KoreLogic. Pathwell topologies. *KoreLogic Blog*, 2014. https://blog.korelogic.com/blog/2014/04/04/pathwell_topologies.
- [112] KoreLogic. "Analytical Solutions: Password Recovery Service. <http://contest-2010.korelogic.com/prs.html, 2015>.
- [113] Munir Kotadia. Gates predicts death of the password. CNET News, February 25, 2004.
- [114] Naveen Kumar. Password in practice: An usability survey. *Journal of Global Research in Computer Science*, 2(5), 2011.
- [115] Cynthia Kuo, Sasha Romanosky, and Lorrie Faith Cranor. Human selection of mnemonic phrase-based passwords. In *Proc. SOUPS*, 2006.
- [116] John Leyden. Office workers give away passwords for a cheap pen. *The Register*, 2003.
- [117] Rui Li, Shengjie Wang, Hongbo Deng, Rui Wang, and Kevin Chen-Chuan Chang. Towards social user profiling: Unified and discriminative influence model for inferring home locations. In *Proc. KDD*, 2012.
- [118] Zhigong Li, Weili Han, and Wenyuan Xu. A large-scale empirical analysis of Chinese web passwords. In *Proc. USENIX Security*, 2014.
- [119] Jialiu Lin, Shahriyar Amini, Jason Hong, Norman Sadeh, Janne Lindqvist, and Joy Zhang. Expectation and purpose: Understanding users' mental models of mobile app privacy through crowdsourcing. In *Proc. UbiComp*, 2012.
- [120] Jialiu Lin, Bin Liu, Norman Sadeh, and Jason I. Hong. Modeling users' mobile app privacy preferences: Restoring usability in a sea of permission settings. In *Proc. SOUPS*, 2014.
- [121] George Loewenstein and Emily Celia Haisley. The economist as therapist: Methodological ramifications of 'light' paternalism. In *The Foundations of Positive and Normative Economics*. Oxford University Press, 2008.
- [122] Dylan Love. Apple on iCloud breach: It's not our fault hackers guessed celebrity passwords. *International Business Times*, September 2 2014. <http://www.ibtimes.com/apple-icloud-breach-its-not-our-fault-hackers-guessed-celebrity-passwords-1676268>.

- [123] Jerry Ma, Weining Yang, Min Luo, and Ninghui Li. A study of probabilistic password models. In *Proc. IEEE SP*, 2014.
- [124] David Malone and Kevin Maher. Investigating the distribution of password choices. In *Proc. WWW*, 2012.
- [125] Simon Marechal. Automatic wordlist mangling rule generation. *Openwall Blog*, 2012. <http://www.openwall.com/presentations/Passwords12-Mangling-Rules-Generation/>.
- [126] 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 *Proc. CCS*, 2013.
- [127] B. Dawn Medlin and Joseph A. Cazier. An empirical investigation: Health care employee passwords and their crack times in relationship to HIPAA security standards. *IJHISI*, 2(3), 2007.
- [128] Eric Medvet, Engin Kirda, and Christopher Kruegel. Visual-similarity-based phishing detection. In *Proc. SecureComm*, 2008.
- [129] William Melicher, Darya Kurilova, Sean M. Segreti, Pranshu Kalvani, Richard Shay, Blase Ur, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Michelle L. Mazurek. Usability and security of text passwords on mobile devices. In *Proc. CHI*, 2016.
- [130] William Melicher, Blase Ur, Sean M. Segreti, Saranga Komanduri, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Fast, lean, and accurate: Modeling password guessability using neural networks. In *Proc. USENIX Security*, 2016.
- [131] David A. Milman. Death to passwords. ComputerWorld. http://blogs.computerworld.com/17543/death_to_passwords, 2010.
- [132] Robert Morris and Ken Thompson. Password security: A case history. *CACM*, 22(11), 1979.
- [133] Mozilla Developer. Using Web workers. https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers, Accessed 2016.
- [134] MWR InfoSecurity. MWR InfoSecurity, 2014. <https://www.mwrinfosecurity.com/>.
- [135] Arvind Narayanan and Vitaly Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *Proc. CCS*, 2005.
- [136] Gilbert Notoatmodjo and Clark Thomborson. Passwords and perceptions. In *Proc. AISC*, 2009.
- [137] Openwall. Wordlists. <http://download.openwall.net/pub/wordlists/>, 2015.
- [138] Colin Percival. Stronger key derivation via sequential memory-hard functions. <http://www.tarsnap.com/scrypt/scrypt.pdf>, 2009.
- [139] Alexander Peslyak. John the Ripper. <http://www.openwall.com/john/>, 1996.

- [140] PHDays. “Hash Runner” - Positive Hack Days. <http://2013.phdays.com/program/contests/>, 2013.
- [141] Pieroxy. lz-string: Javascript compression, fast! <http://pieroxy.net/blog/pages/lz-string/index.html>, Accessed 2016.
- [142] Robert W. Proctor, Mei-Ching Lien, Kim-Phuong L. Vu, E. Eugene Schultz, and Gavriel Salvendy. Improving computer security for authentication of users: Influence of proactive password restrictions. *Behavior Research Methods, Instruments, & Computers*, 34(2):163–169, 2002.
- [143] Niels Provos and David Mazieres. A future-adaptable password scheme. In *Proc. USENIX ATC*, 1999.
- [144] Emilee Rader, Rick Wash, and Brandon Brooks. Stories as informal lessons about security. In *Proc. SOUPS*, 2012.
- [145] Ashwini Rao, Birendra Jha, and Gananand Kini. Effect of grammar on security of long passwords. In *Proc. CODASPY*, 2013.
- [146] Rapid7. LinkedIn passwords lifted, retrieved September 2012. <http://www.rapid7.com/resources/infographics/linkedin-passwords-lifted.html>.
- [147] Joel Ross, Lilly Irani, M. Six Silberman, Andrew Zaldivar, and Bill Tomlinson. Who are the crowd-workers?: Shifting demographics in Mechanical Turk. In *Proc. CHI Extended Abstracts*, 2010.
- [148] Norman Sadeh, Alessandro Acquisti, Travis D. Breaux, Lorrie Faith Cranor, Aleecia M. McDonald, Joel Reidenberg, Noah A. Smith, Fei Liu, N. Cameron Russell, Florian Schaub, Shomir Wilson, James T. Graves, Pedro Giovanni Leon, Rohan Ramanath, and Ashwini Rao. Poster: Towards usable privacy policies: Semi-automatically extracting data practices from websites’ privacy policies. In *Proc. SOUPS Extended Abstracts*, 2014.
- [149] Stuart Schechter, Cormac Herley, and Michael Mitzenmacher. Popularity is everything: a new approach to protecting passwords from statistical-guessing attacks. In *Proc. HotSec*, 2010.
- [150] Bruce Schneier. MySpace passwords aren’t so dumb. <http://www.wired.com/politics/security/commentary/securitymatters/2006/12/72300>, 2006.
- [151] Bruce Schneier. Password advice. http://www.schneier.com/blog/archives/2009/08/password_advice.html, August 2009.
- [152] Bruce Schneier. Choosing secure passwords. Schneier on Security https://www.schneier.com/blog/archives/2014/03/choosing_secure_1.html, March 3, 2014.
- [153] SCOWL. Spell checker oriented word lists. <http://wordlist.sourceforge.net>, 2015.
- [154] Richard Shay, Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Blase Ur, Timothy Vidas, Lujio Bauer, Nicholas Christin, and Lorrie Faith Cranor. Correct horse battery staple: Exploring the usability of system-assigned passphrases. In *Proc. SOUPS*, 2012.

- [155] Richard Shay, Saranga Komanduri, Adam L. Durity, Phillip (Seyoung) Huh, Michelle L. Mazurek, Sean M. Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Can long passwords be secure and usable? In *Proc. CHI*, 2014.
- [156] Richard Shay, Saranga Komanduri, Adam L. Durity, Phillip (Seyoung) Huh, Michelle L. Mazurek, Sean M. Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Designing password policies for strength and usability. *ACM TISSEC*, 18(4):13, 2016.
- [157] Richard Shay, Saranga Komanduri, Adam L. Durity, Phillip (Seyoung) Huh, Michelle L. Mazurek, Sean M. Segreti, Blase Ur, Lujo Bauer, Lorrie Faith Cranor, and Nicolas Christin. Can long passwords be secure and usable? In *Proc. CHI*, 2014.
- [158] Richard Shay, Saranga Komanduri, Patrick Gage Kelley, Pedro Giovanni Leon, Michelle L. Mazurek, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Encountering stronger password requirements: User attitudes and behaviors. In *Proc. SOUPS*, 2010.
- [159] Steven K. Shevell, editor. *The Science of Color*. Elsevier, 2003.
- [160] Supriya Singh, Anuja Cabraal, Catherine Demosthenous, Gunela Astbrink, and Michele Furlong. Password sharing: Implications for security design based on social practice. In *Proc. CHI*, 2007.
- [161] Daniel J. Solove and Woodrow Hartzog. Should the FTC kill the password? The case for better authentication. Bloomberg BNA Privacy & Security Law Report 1353, 2015.
- [162] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on SSH. In *Proc. USENIX Security Symposium*, 2001.
- [163] Andreas Sotirakopoulos, Ildar Muslukov, Konstantin Beznosov, Cormac Herley, and Serge Egelman. Motivating users to choose better passwords through peer pressure. *Proc. SOUPS Posters*, 2011.
- [164] SpiderOak. Zero knowledge cloud solutions. <https://spideroak.com/>, 2016.
- [165] Jeffrey M. Stanton, Kathryn R. Stam, Paul Mastrangelo, and Jeffrey Jolton. Analysis of end user security behaviors. *Computers & Security*, 24(2):124–133, 2005.
- [166] Jens Steubbe. Hashcat. <http://hashcat.net/oclhashcat-plus/>, 2009.
- [167] Jens Steube. Mask Attack. https://hashcat.net/wiki/doku.php?id=mask_attack, 2009-.
- [168] Jens Steube. Rule-based Attack. https://hashcat.net/wiki/doku.php?id=rule_based_attack, 2009-.
- [169] Elizabeth Stobert and Robert Biddle. The password life cycle: User behaviour in managing passwords. In *Proc. SOUPS*, 2014.
- [170] Elizabeth Stobert and Robert Biddle. Expert password management. In *Proc. Passwords*, 2015.
- [171] Stricture Consulting Group. Password audits. <http://stricture-group.com/services/password-audits.htm>, 2015.

- [172] Wayne C. Summers and Edward Bosworth. Password policy: The good, the bad, and the ugly. In *Proc. WISICT*, 2004.
- [173] San-Tsai Sun, Eric Pospisil, Ildar Muslukhov, Nuray Dindar, Kirstie Hawkey, and Konstantin Beznosov. What makes users refuse web single sign-on?: An empirical investigation of OpenID. In *Proc. SOUPS*, 2011.
- [174] Terms of Service; Didn't Read. <http://tosdr.org/>.
- [175] Richard H. Thaler and Cass R. Sunstein. *Nudge: Improving decisions about health, wealth, and happiness*. Yale University Press, 2008.
- [176] Michael Toomim, Travis Kriplean, Claus Pörtner, and James Landay. Utility of human-computer interactions: Toward a science of preference measurement. In *Proc. CHI*, 2011.
- [177] Trustwave. 2014 business password analysis. *Password Research*, 2014.
- [178] Trustwave Spiderlabs. eHarmony password dump analysis, June 2012. <http://blog.spiderlabs.com/2012/06/eharmony-password-dump-analysis.html>.
- [179] Trustwave Spiderlabs. SpiderLabs/KoreLogic-Rules. <https://github.com/SpiderLabs/KoreLogic-Rules>, 2012.
- [180] Janice Y. Tsai, Serge Egelman, Lorrie F. Cranor, and Alessandro Acquisti. The effect of online privacy information on purchasing behavior: An experimental study. *Information Systems Research*, 22(2):254–268, June 2011.
- [181] Blase Ur, Jonathan Bees, Sean M. Segreti, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Do users' perceptions of password security match reality? In *Proc. CHI*, 2016.
- [182] Blase Ur, Patrick Gage Kelly, Saranga Komanduri, Joel Lee, Michael Maass, Michelle Mazurek, Timothy Passaro, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. How does your password measure up? The effect of strength meters on password creation. In *Proc. USENIX Security*, August 2012.
- [183] Blase Ur, Saranga Komanduri, Richard Shay, Stephanos Matsumoto, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Patrick Gage Kelley, Michelle L. Mazurek, and Timothy Vidas. Poster: The art of password creation. In *Proc. IEEE SP Posters*, 2013.
- [184] Blase Ur, Fumiko Noma, Jonathan Bees, Sean M. Segreti, Richard Shay, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. "I added '!' at the end to make it secure": Observing password creation in the lab. In *Proc. SOUPS*, 2015.
- [185] Blase Ur, Sean M. Segreti, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Saranga Komanduri, Darya Kurilova, Michelle L. Mazurek, William Melicher, and Richard Shay. Measuring real-world accuracies and biases in modeling password guessability. In *Proc. USENIX Security*, 2015.
- [186] Anthony Vance, David Eargle, Kirk Ouimet, and Detmar Straub. Enhancing password security through interactive fear appeals: A web-based field experiment. In *Proc. HICSS*, 2013.

- [187] Ashlee Vance. If your password is 123456, just make it HackMe. New York Times, <http://www.nytimes.com/2010/01/21/technology/21password.html>, 2010.
- [188] Rafael Veras, Christopher Collins, and Julie Thorpe. On the semantic patterns of passwords and their security impact. In *Proc. NDSS*, 2014.
- [189] Rafael Veras, Julie Thorpe, and Christopher Collins. Visualizing semantics in passwords: The role of dates. In *Proc. VizSec*, 2012.
- [190] Melanie Volkamer and Karen Renaud. Mental models – general introduction and review of their application to human-centred security. In *Number Theory and Cryptography*, volume 8260 of *Lecture Notes in Computer Science*, pages 255–280. 2013.
- [191] Emanuel von Zezschwitz, Alexander De Luca, and Heinrich Hussmann. Survival of the shortest: A retrospective analysis of influencing factors on password composition. In *Proc. INTERACT*, 2013.
- [192] Emanuel von Zezschwitz, Alexander De Luca, and Heinrich Hussmann. Honey, I shrunk the keys: Influences of mobile devices on password composition and authentication performance. In *Proc. NordiCHI*, 2014.
- [193] Kim-Phuong L. Vu, Robert W. Proctor, Abhilasha Bhargav-Spantzel, Bik-Lam (Belin) Tai, and Joshua Cook. Improving password security and memorability to protect personal and organizational information. *IJHCS*, 65(8):744–757, 2007.
- [194] Rick Wash. Folk models of home computer security. In *Proc. SOUPS*, 2010.
- [195] Matt Weir. The RockYou 32 million password list top 100. <http://reusablesec.blogspot.com/2009/12/rockyou-32-million-password-list-top.html>, December 2009.
- [196] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proc. CCS*, 2010.
- [197] Matt Weir, Sudhir Aggarwal, Breno de Medeiros, and Bill Glodek. Password cracking using probabilistic context-free grammars. In *Proc. IEEE SP*, 2009.
- [198] Dan Wheeler. zxcvbn: Realistic password strength estimation. <https://blogs.dropbox.com/tech/2012/04/zxcvbn-realistic-password-strength-estimation/>, 2012.
- [199] Dan Lowe Wheeler. zxcvbn: Low-budget password strength estimation. In *Proc. USENIX Security*, 2016.
- [200] Wikipedia. Wikimedia downloads. <http://dumps.wikimedia.org/>.
- [201] Craig E. Wills and Mihajlo Zeljkovic. A personalized approach to web privacy: awareness, attitudes and actions. *Information Management & Computer Security*, 19(1):53–73, 2011.
- [202] Jianxin Jeff Yan. A note on proactive password checking. In *Proc. NSPW*, 2001.
- [203] Yulong Yang, Janne Lindqvist, and Antti Oulasvirta. Text entry method affects password security. In *Proc. LASER*, 2014.

- [204] Shiva Houshmand Yazdi. Analyzing password strength and efficient password cracking. Master's thesis, The Florida State University, 2011.
- [205] Yinqian Zhang, Fabian Monrose, and Michael K Reiter. The security of modern password expiration: An algorithmic framework and empirical analysis. In *Proc. CCS*, 2010.
- [206] Leah Zhang-Kennedy, Sonia Chiasson, and Robert Biddle. Password advice shouldn't be boring: Visualizing password guessing attacks. In *Proc. eCRS*, 2013.
- [207] Moshe Zviran and William J. Haga. Password security: An empirical study. *J. Mgt. Info. Sys.*, 15(4), 1999.

Appendices

Appendix A

Surveys from “How Does Your Password Measure Up...”

A.1 Survey Questions

A.1.1 Day 1 Survey

Page 1

We will now ask you some questions about the password you just created. After you finish this survey, we will ask you to reenter your password, at which point you will receive the MTurk confirmation code. Don't worry if you've forgotten your password; you will still receive the code!

1. Creating a password that meets the requirements given in this study was annoying.
 Strongly disagree Disagree Neutral Agree Strongly agree NA
2. Creating a password that meets the requirements given in this study was difficult.
 Strongly disagree Disagree Neutral Agree Strongly agree NA
3. Creating a password that meets the requirements given in this study was fun.
 Strongly disagree Disagree Neutral Agree Strongly agree NA
4. If I changed my real email account password to the password I created in this study, it would make my email account less secure.
 Strongly disagree Disagree Neutral Agree Strongly agree NA

Page 2

All questions on this page are about the password strength meter (above) that you saw when creating your password.

1. The password strength meter helped me create a stronger password that I would have otherwise.
 Strongly disagree Disagree Neutral Agree Strongly agree NA

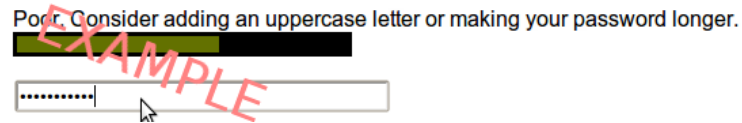


Figure A.1: Example meter shown to participants.

2. The password strength meter was not informative.
 - Strongly disagree Disagree Neutral Agree Strongly agree NA
3. I created a different password than I would have otherwise based on feedback from the password strength meter.
 - Strongly disagree Disagree Neutral Agree Strongly agree NA
4. It's important to me that the password strength meter gives my password a high score.
 - Strongly disagree Disagree Neutral Agree Strongly agree NA
5. I did not understand how the password strength meter rated my password.
 - Strongly disagree Disagree Neutral Agree Strongly agree NA
6. I think the password strength meter gave an incorrect score of my password's strength.
 - Strongly disagree Disagree Neutral Agree Strongly agree NA
7. I often see password meters when I'm creating new online accounts.
 - Strongly disagree Disagree Neutral Agree Strongly agree NA
8. I would have created the same password without the password strength meter.
 - Strongly disagree Disagree Neutral Agree Strongly agree NA
9. The password strength meter was annoying.
 - Strongly disagree Disagree Neutral Agree Strongly agree NA

Page 3

1. While creating your password, how many times did you look at the password strength meter?
 - Not at all
 - Once or twice
 - Three to five times
 - More than five times
 - I watched it the whole time while I was creating my password
2. Please explain how the password strength meter helped you create a better password, or explain why it was not helpful.
3. How did the password meter impact the password you chose?
 - It had no impact
 - I typed a password and then made it longer to increase the password meter score

- I typed a password and then added some special characters to increase the password meter score
- I typed a password and then deleted some or all of what I typed and tried again to increase the password meter score
- Since I saw the meter I tried harder to pick a good password that would be likely to have a high password meter score
- I tried multiple passwords to see what the meter would say before I picked one to submit
- Other -----

Page 3

1. What is your gender?
 - Male
 - Female
 - I prefer not to answer
2. How old are you?
3. Are you majoring in or do you have a degree or job in computer science, computer engineering, information technology, or a related field?
 - Yes
 - No
 - I prefer not to answer
4. What country do you live in?
5. Are you red-green colorblind?
 - Yes
 - No
 - I prefer not to answer

A.1.2 Day 2 Survey**Page 1**

Thank you for participating in this Carnegie Mellon University password study. The purpose of this study is to help us learn about how people use different types of password systems. Please answer the following questions honestly. There are no right or wrong answers and everyone who finishes this task completely will receive their bonus payment.

1. When you created your password for this study, which of the following did you do?
 - I reused a password I was already using for an email account
 - I modified a password I was already using for an email account
 - I reused a password I was already using for a different account
 - I modified a password I was already using for a different account
 - I created an entirely new password
 - Other -----

2. Was the password you created for this study stored by your web browser?
 No
 Yes
3. Did you write down the password you created for this study?
 No
 Yes, on paper
 Yes, electronically (stored in computer, phone, etc.)
 Other -----
4. If you wrote down your password for this study, how is it protected? (choose all that apply)
 I did not protect it
 I stored it in an encrypted file
 I hid it
 I stored it on a computer or device protected with another password
 I locked up the paper
 I always keep the password with me
 I wrote down a reminder instead of the actual password
 I keep the paper in an office or room that only I use
 I stored it on a computer or device that only I use
 Other -----

Page 2

1. Have you written down your real email account password?
 No
 Yes, on paper
 Yes, electronically (stored in computer, phone, etc.)
 Other
2. If you wrote down your real email account password, how is it protected? (choose all that apply)
 I did not protect it
 I stored it in an encrypted file
 I hid it
 I stored it on a computer or device protected with another password
 I locked up the paper
 I always keep the password with me
 I wrote down a reminder instead of the actual password
 I keep the paper in an office or room that only I use
 I stored it on a computer or device that only I use
 Other
3. To how many people have you given your real email account password?
4. Remembering the password I created for this study was difficult.
 Strongly disagree Disagree Neutral Agree Strongly agree NA

5. Remembering the password I use for my main email account is difficult.
 Strongly disagree Disagree Neutral Agree Strongly agree NA

6. The password I created for this study is weaker or less secure than a password I would use for one of my real accounts.
 Strongly disagree Disagree Neutral Agree Strongly agree NA

Appendix B

Surveys and Full Data from “Do Users’ Perceptions...”

B.1 Survey Questions

B.1.1 Part 1

This study consists of 5 parts.

Please answer the following demographic questions.

1. What is your gender?
 - Male
 - Female
 - Other
 - Prefer not to answer
2. What is your age?
3. Have you ever held a job or received a degree in computer science or any related technology field?
 - Yes
 - No
4. Are you either a computer security professional or a student studying computer security?
 - Yes
 - No

B.1.2 Part 2 (Repeated 26 times)

The next 26 questions will ask you to compare pairs of passwords and rate which is more secure, or whether they are equally secure. As you think about these comparisons, imagine that you use this password for an important account that is protecting information that you care a lot about.

1. In your opinion, which of the following passwords is more secure?

 p_1 p_2

- p_1 is much more secure
- p_1 is more secure
- p_1 is slightly more secure
- Both passwords are equally secure
- p_2 is slightly more secure
- p_2 is more secure
- p_2 is much more secure

2. Why?

B.1.3 Part 3 (Repeated 20 times)

For the next 20 questions, we will present a password and ask you to rate how secure you think it is, as well as how easy you think it is to remember. As you choose ratings, imagine that you use this password for an important account that is protecting information that you care a lot about.

1. Please rate the security of the following password: p_n
(very insecure) 1 2 3 4 5 6 7 (very secure)
2. Please rate the memorability of the following password: p_n
(very hard to remember) 1 2 3 4 5 6 7 (very easy to remember)

B.1.4 Part 4 (Repeated 11 times)

For the next 11 questions, we will present a strategy for creating passwords and ask you to rate how secure you think it is, as well as how easy you think it is to remember. As you choose ratings, imagine that you use this password for an important account that you care a lot about.

Consider the following password-creation strategy: $strategy_n$

1. Please rate the security of passwords created using that strategy.
(very insecure) 1 2 3 4 5 6 7 (very secure)
2. Please rate the memorability of passwords created using that strategy.
(very hard to remember) 1 2 3 4 5 6 7 (very easy to remember)

B.1.5 Part 5

The final 7 questions cover attacks against passwords.

1. In your opinion, what characteristics make a password easy for an attacker to guess?
2. In your opinion, what characteristics make a password hard for an attacker to guess?

3. Please describe the type of attacker (or multiple types of attackers), if any, whom you worry might try to guess your password.
4. As far as you know, why would an attacker try to guess your password, if at all?
5. As far as you know, how do attackers try to guess your password?
6. How many guesses (by an attacker) would a password need to be able to withstand for you to consider it secure?
7. Please explain how you arrived at the number of guesses you wrote for the previous question.

B.2 Full List of Password Pairs

PW ₁	PW ₂	Stronger	Perceived Stronger	p
MAncity123	ManCity123	PW ₁	PW ₁	< .001
uniVersal	Universal	PW ₁	PW ₁	< .001
goLD.tEeTh.	gold.teeth.	PW ₁	PW ₁	< .001
iloveyou!	iloveyou	PW ₁	PW ₁	< .001
CHAINLOW!	CHAINLOW	PW ₁	PW ₁	< .001
!angryinch	angryinch	PW ₁	PW ₁	< .001
babydoll1	babydoll	PW ₁	PW ₁	< .001
Indiana1	Indiana	PW ₁	PW ₁	< .001
michelle1	michelle	PW ₁	PW ₁	< .001
jdsprig1!	jdsprig	PW ₁	PW ₁	< .001
scotishot1!	scotishot	PW ₁	PW ₁	< .001
rredheadss1!	rredheadss	PW ₁	PW ₁	< .001
restless1	restlessu	PW ₂	PW ₂	< .001
puppydog3	puppydogv	PW ₂	Neither	n.s.
5jungle1	5junglek	PW ₂	PW ₂	< .001
bobcat01	bobc01at	PW ₂	PW ₂	< .001
spongebob01	sponge01bob	PW ₂	PW ₂	< .001
99tinkerbells	tink99erbells	PW ₂	PW ₂	< .001
\$HREEne2!\$	HRE\$Ene\$2!	PW ₂	PW ₂	.039
mbmcbg8&	mbmc&bg8	Neither	Neither	n.s.
#!madalion	mada#!lion	PW ₂	PW ₂	< .001
barbara1993	barbara7391	PW ₂	PW ₂	< .001
feliz2009	feliz5224	PW ₂	PW ₂	< .001
islandia2007	islandia3847	PW ₂	PW ₂	< .001
Thumper123	Thumper728	PW ₂	PW ₂	< .001
babig123	babig974	PW ₂	PW ₂	< .001
badboys234	badboys833	Neither	PW ₂	.001
jonny1421	jonnyrtxe	PW ₂	Neither	n.s.

PW ₁	PW ₂	Stronger	Perceived Stronger	p
brooklyn16	brooklynqy	PW ₂	Neither	n.s.
astley123	astleyabc	PW ₂	Neither	n.s.
sandman@	sandman2	PW ₁	PW ₁	< .001
:):)salasar	1989salasar	PW ₁	PW ₁	< .001
isaac^joe	isaac6joe	PW ₁	PW ₁	< .001
0849023189)*\$()@#!*(PW ₂	PW ₂	< .001
18521852	!*%@!*%@	PW ₂	PW ₂	< .001
123456789	!@\$%^&*(PW ₂	PW ₂	< .001
Caitlin1016	Treason1016	PW ₂	PW ₂	< .001
08alyssa	08tomato	PW ₂	PW ₂	< .001
jackie1234	soccer1234	Neither	PW ₂	.034
survey77†	boring77	PW ₂ †	PW ₂	< .001
ilikeamazon†	ilikecereal	PW ₂ †	PW ₂	< .001
turkturk†	gamegame	PW ₂ †	PW ₂	< .001
iloveliverpool	questionnaires	PW ₂	Neither	n.s.
computer	moldovan	PW ₂	PW ₂	< .001
Cryingwolf	Desiccated	Neither	Neither	n.s.
iknewyouweretrouble	punkygirlfairyrules	Neither	Neither	n.s.
goldenkey	jadeisfit	Neither	Neither	n.s.
iloveyou88	ieatkale88	PW ₂	Neither	n.s.
bluewater	nightgown	PW ₂	Neither	n.s.
momoffive	cornflake	PW ₁	PW ₁	< .001
loveispain	strawberry	PW ₁	PW ₁	< .001
1qaz2wsx3edc	thefirstkiss	PW ₂	PW ₁	< .001
1234567890	livestrong	PW ₂	PW ₂	< .001
qwertyuiop	bradybunch	PW ₂	Neither	n.s.
abc123def789	293070844005	PW ₂	Neither	n.s.
1b2b3b4b5b	5ndco2sjg7	PW ₂	PW ₂	< .001
asdfghjkl1	msutvsxoal	PW ₂	PW ₂	< .001
07211985†	07251985	PW ₂ †	PW ₂	< .001
11-8-1992†	4-13-1997	PW ₂ †	PW ₂	< .001
KELLY1975†	KELLY1998	PW ₂ †	PW ₂	< .001
carmen89†	laurie89	PW ₂ †	PW ₂	< .001
iloverobert†	ilovethomas	PW ₂ †	PW ₂	< .001
yolandamarina†	jessicamelisa	PW ₂ †	PW ₂	< .001
L0vemetal	Lovemetal	Neither	PW ₁	< .001
sw33th3art	sweetheart	PW ₁	PW ₁	< .001
Z@kie142	Zakie142	PW ₁	PW ₁	< .001
gas011nal0v3	gasO1InalOvE	Neither	Neither	n.s.
jan3tt3chr1st1na	janEttEchrIstIna	Neither	Neither	n.s.
p@ssw0rd	pAsswOrd	PW ₂	PW ₁	< .001
l0ngh0rns	l5ngh5rns	PW ₂	PW ₂	< .001

PW₁	PW₂	Stronger	Perceived Stronger	p
punk4life	punk7life	PW ₂	PW ₂	< .001
sk8erboy	sk2erboy	PW ₂	PW ₂	< .001
KL+MO4EVER	KL+MOFOREVER	PW ₁	PW ₁	< .001
punk4life	punkforlife	PW ₂	PW ₁	< .001
sk8erboy	skaterboy	Neither	PW ₁	< .001

Table B.1: The full list of password pairs we tested, along with the relative security of the passwords according to our models of password cracking and participants' perceptions. Significant p values indicate that there was a preference toward one password over the other.

Appendix C

Scripts/Surveys from “...Data-Driven Password Meter...”

C.1 Script for Laboratory Study

C.1.1 Introduction

Hello, my name is [name]. Thanks for coming! Before we get started, are you okay with me recording audio for our session? [await confirmation] Great.

Today, we would like your feedback about making passwords and keeping track of them. We'll be using your feedback to inform the design of a tool to help people make better passwords.

First, some risks and benefits.

Potential Risks: Talking about your own password can be a sensitive subject, so please don't say anything you wouldn't want shared outside of this session. I will not ask what any of your actual passwords are, nor should you reveal that information.

Benefits: You will be compensated with a \$20 Amazon gift card. However, you will not receive any other benefits from participating in this study.

You are allowed to leave at any time. If you still consent to being part of this study, please say, “Yes.” [await verbal confirmation] Great.

The purpose of this study is to inform the design of tools to help users like you create better passwords. We'll start with some general questions about how you create and remember passwords. Then, I'll show you a slideshow of designs for password tools that companies and researchers have created. Afterwards, we'll go through a hands-on portion, where you can try out some tools. In all portions of the study, please feel encouraged to point out aspects you like about designs, as well as aspects you don't like. There is not a particular design we hope you'll like better than the others; we're most interested in your honest and blunt feedback for everything we show you.

C.1.2 Part I – Open, Abstract Discussion of Password Creation

1. First, when did you last create or change a password? Why?
2. What are your goals when you are creating a password?

3. What makes a password secure? [Follow-up question if relevant] How did you learn what makes a password secure?
4. Have any particular sources of information been particularly helpful in learning how to make secure passwords?
5. Have you seen any news articles or TV programs where they talk about password advice? What have you learned from them?
6. Have you ever received feedback on the screen where you are creating a password? Is this feedback helpful?
7. (If not mentioned) Are you familiar with password meters, which rate the strength of your passwords as you create them? What do you think of them?
8. When you make a password, what kinds of threats, if any, are you trying to protect against?
9. How do you keep track of all your passwords? [Follow-up question if relevant] Where did you learn to do that?

C.1.3 Part II – Password Feedback Slideshow

We’ve been talking a bit about password feedback, but now let’s actually look at some designs for giving feedback. I’m going to go through a slideshow of some example designs.

As we’re looking at each of these, I want you to give me your thoughts and feedback about each one. Please tell me what you like about it, as well as what you don’t like about it?

See Figure C.1 for the slideshow images.

C.1.4 Part III – Meter Testing

Okay, next is the hands-on portion. I’d like each of you to spend about 5 minutes trying passwords on two different prototype tools to get a feel for how they work. [give laptop, show them tabs with prototypes, and identify the nomenclature] As you do that, please jot down some brief notes on the pros and cons sheets we’ve provided for each meter. Feel free to add any other comments as well. Try some of the same passwords on both. In addition, at the end, I’ll give you a sheet of sample passwords to try to make sure you’ve seen different features of the meter. Unlike some experiments, there isn’t one design we secretly hope you prefer, so please give us your honest and candid feedback on both. If you have any questions about what you should do, I’d be happy to help.

[Allow at least 5 minutes for the meter testing session. Keep going until the participant seems to lose interest.]

Okay, now we’re going to talk a little bit about the prototypes.

1. What did you like about each prototype?
2. What did you dislike about each prototype?
3. What would you change?

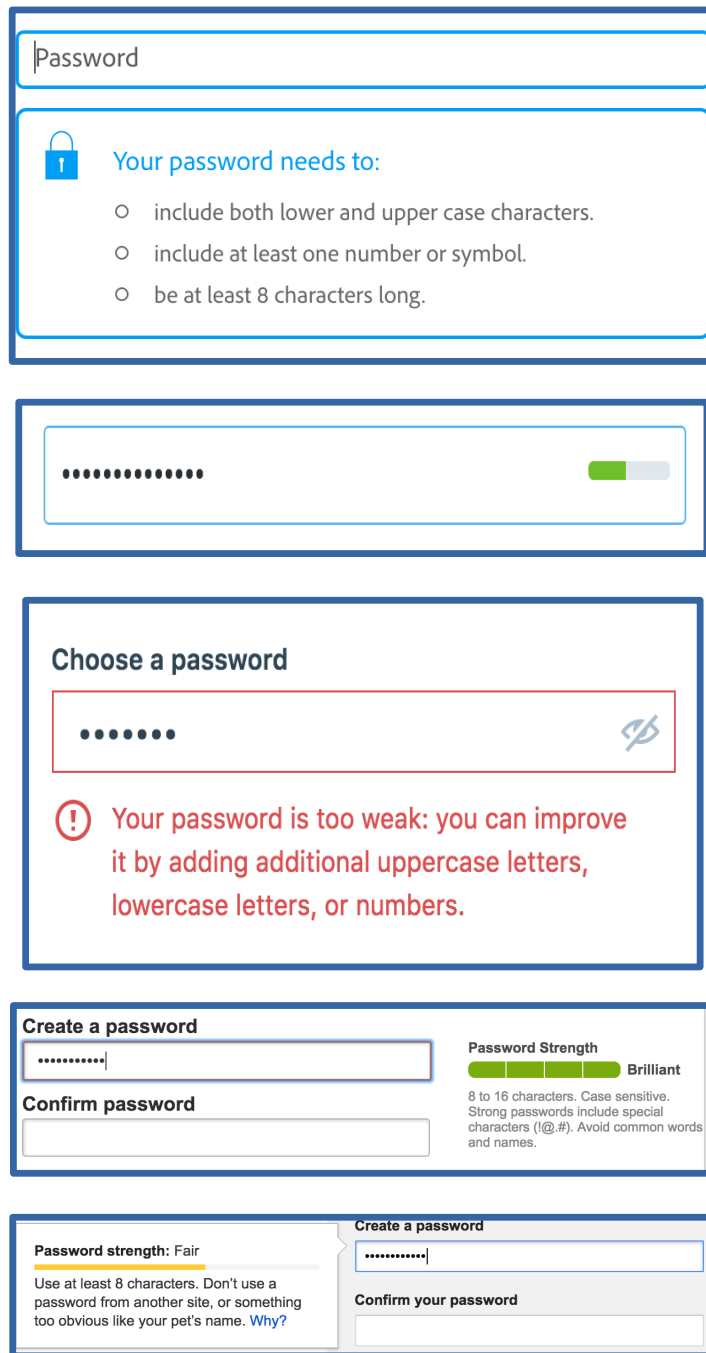


Figure C.1: Images we showed in the meter slideshow. They are respectively screenshots from password meters used on Adobe, Twitter, Wordpress, AOL Mail, and Google.

C.1.5 Part IV – Ending

Great, just to wrap up, I have two final questions.

1. What kinds of advice, feedback, and help are most useful to you as you create a password?
2. Is there anything else you’d like to add about today’s tasks, or about passwords in general?

Thank you again for helping us out we truly value your feedback. We will now give you your \$20 Amazon gift cards. If you have any other questions, I’d be happy to answer them now. Thanks, again!

C.2 Surveys from Online Study

C.2.1 Day 1 Survey

Page 1

Thank you for creating a password. Next, we would like you to answer some survey questions.

1. Creating a password during this study was annoying.
 Strongly disagree Disagree Neutral Agree Strongly agree
2. Creating a password during this study was fun.
 Strongly disagree Disagree Neutral Agree Strongly agree
3. Creating a password during this study was difficult.
 Strongly disagree Disagree Neutral Agree Strongly agree
4. Which of the following best describes your approach to creating your password in this study?
 I reused a password that I currently use, or previously have used, for a different account.
 I modified a password that I currently use, or previously have used, for a different account.
 I created an entirely new password, but I used the same general approach that I normally do.
 I created an entirely new password, and I used a different approach than I normally do.
5. In general, I am confident in my ability to create strong passwords.
 Strongly disagree Disagree Neutral Agree Strongly agree

Page 2 (for participants who saw the colored bar)

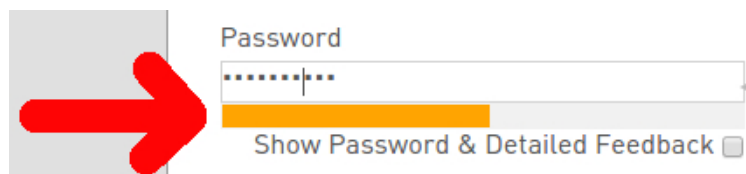


Figure C.2: Example colored bar shown to participants.

The following questions refer only to the colored bar (measuring the strength of your password) that you saw during the study. A sample is shown above.

1. The colored bar helped me create a stronger password.
 Strongly disagree Disagree Neutral Agree Strongly agree
2. The colored bar was not informative.
 Strongly disagree Disagree Neutral Agree Strongly agree
3. Because of the colored bar, I created a different password than I would have otherwise.
 Strongly disagree Disagree Neutral Agree Strongly agree
4. What is your opinion of the accuracy of the colored bar's rating?
 The colored bar's rating accurately reflected the strength of my password.
 The colored bar's rating did not accurately reflect the strength of my password; the colored bar gave my password a lower score than it deserved.
 The colored bar's rating did not accurately reflect the strength of my password; the colored bar gave my password a higher score than it deserved.
 I don't remember how the colored bar rated my password.
5. Do you have any other thoughts about the colored bar?

Page 3 (for participants who saw text feedback)

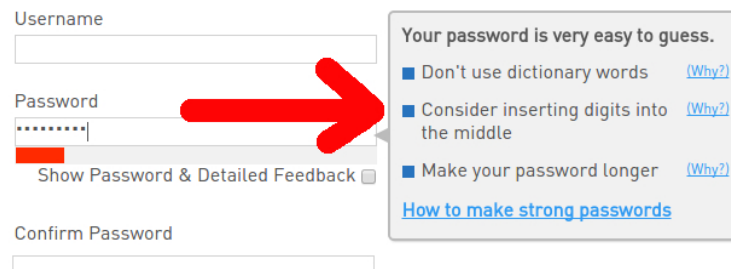


Figure C.3: Example text feedback shown to participants.

The following questions refer only to the text feedback you saw during the study. A sample is shown above. They also refer to any text feedback you saw on other screens, if you happened to click any links that were given.

1. The text feedback helped me create a stronger password.
 Strongly disagree Disagree Neutral Agree Strongly agree
2. The text feedback was not informative.
 Strongly disagree Disagree Neutral Agree Strongly agree
3. Because of the text feedback, I created a different password than I would have otherwise.
 Strongly disagree Disagree Neutral Agree Strongly agree

4. Did you learn something new about passwords (your password, or passwords in general) from the text feedback?
 Yes No
5. (If “Yes,” automatically displays) What new thing(s) did you learn?
6. (If “No,” automatically displays) Where have you learned how to make strong passwords?
7. Do you have any other thoughts about the text feedback?

Page 4 (for participants who could be shown a suggested improvements)



Figure C.4: Example suggested improvement shown to participants.

The following questions refer only to the suggested improvements to your password that you may have, or may not have, seen during the study. A sample is shown above for the password “Mypassword123”

1. Did you see suggested improvements when you were creating your password?
 Yes
 No
 I don’t remember
2. (If “Yes,” automatically displays) The suggested improvements helped me create a stronger password.
 Strongly disagree Disagree Neutral Agree Strongly agree
3. In your opinion, is it useful to see suggested improvements while you are creating a password?
 Yes No
4. (If “Yes,” automatically displays) Why?
5. (If “No,” automatically displays) Why not?
6. Do you have any other thoughts about the suggested improvements?

Page 5

1. With what gender do you identify?
 Female
 Male
 Other
 I prefer not to answer
2. How old are you?

3. Are you majoring in or do you have a degree or job in computer science, computer engineering, information technology, or a related field?
- Yes
 - No
 - I prefer not to answer

C.2.2 Day 2 Survey

Page 1

The purpose of this study is to help us learn about how people use different types of password systems. **Please answer the following questions honestly.** There are no right or wrong answers and everyone who finishes this task completely will receive their bonus payment.

1. Which of the following statements best reflects **how you entered your password on the previous screen?**
 - My password was automatically entered for me by a password manager or by my browser I typed my password in entirely from memory
 - I had written my password down on paper, and I typed it in after looking it up
 - I had saved my password electronically (e.g., in a file or on my phone), and I typed it in after looking it up
 - Other -----
2. Which of the following statements reflect **how you normally enter passwords in your daily life?** (Choose all that apply)
 - My passwords are automatically entered for me by a password manager or by my browser
 - I type my passwords in entirely from memory
 - I write my passwords down on paper, and I type them in after looking them up
 - I save my passwords electronically (e.g., in a file or on my phone), and I type them in after looking them up
 - Other -----
3. Regardless of how you entered your password on the previous screen, did you do any of the following after you created your password? (Choose all that apply)
 - I wrote my password for this study down on paper
 - I stored my password for this study electronically (e.g., in a file or on my phone)
 - I wrote down or electronically stored hints to help me remember my password for this study, but not my password itself
 - I did not do any of the above
4. What would you have done differently in creating, protecting, and remembering your password if this password were used for an account you use outside this study?
5. Do you use the password you created for this study for any other account?
 - Yes

- No
- I prefer not to answer

Appendix D

Data-Driven Meter Details

D.1 Prioritization of Feedback from Advanced Heuristics

Throughout our group meetings and during the laboratory study, we tested and iteratively updated many prioritizations of the feedback we provided users in the standard meter. For each advanced heuristic, if the associated function has feedback relevant to that particular password, it returns a non-empty string for both `publicFeedback` and `sensitiveFeedback`. If it does not have feedback, which occurs when that heuristic does not indicate a predictable pattern, it returns the empty string. We traverse the list of functions in descending priority for the first (up to) three pieces of feedback to give the user. If, however, our scoring functions rate the password such that its score fills the bar, we ignore all text feedback and tell the user that his or her password appears strong.

The list of functions that provide feedback, in descending order of priority, is as follows:

1. `contextual()`
2. `blacklist()`
3. `combinedDictCheck()`
4. `keyboardPatterns()`
5. `repeats()`
6. `identifyDates()`
7. `repeatedSections()`
8. `alphabeticSequenceCheck()`
9. `commonpwCheck()`
10. `uppercasePredictable()`
11. `digitsPredictable()`
12. `symbolsPredictable()`
13. `duplicatedCharacters()`
14. `pwLength()`

15. countSYMS()
16. countUC()
17. countDIGS()
18. countLC()
19. commonsubstringCheck()
20. structurePredictable()
21. characterClasses()

D.2 Description of Open-Source Meter Files

Our code, which we are releasing open-source,¹ comes packaged with the files listed below.

The main file in our code is `meter.htm`. Beyond the other files listed below, which are already called by `meter.htm` as appropriate, `meter.htm` depends on two common external web-development libraries. These libraries are JQuery (minified version 2.2.4 JS file used for testing) and Bootstrap (minified version 3.3.6 of both the CSS and JS file used for testing).

Both libraries are currently included in `meter.htm`. However, these lines can be removed if the meter code is included in another page that already loads **JQuery version 2.x** and **Bootstrap 3.x**. Note that Bootstrap 3.x currently requires JQuery 2.x and is not compatible with JQuery version 3.x.

All code expects all of the files listed below to be in the same directory.

D.2.1 Files for Displaying the Meter

meter.htm The main file for our demo meter. This file contains the HTML layout for our demo meter, the CSS stylesheet specifying how the different elements should be displayed, as well as all Javascript functions related to modifying the *visual* aspects of the meter.

config.css The primary configuration settings for the meter's visual design are located in this file. These settings include colors, fonts, sizes, and border radii.

config.js The primary configuration settings for the meter are located in this file. Currently, these settings revolve around the password-composition policy of the site, blacklists, and the HTML5 icons used.

D.2.2 Files Related to Calculating Heuristic Scoring

rulefunctions.js A Javascript file containing the functions used to score the password based on 21 different heuristics. The first series of functions are helper functions (mostly string and array prototypes), followed by a function to verify that a password meets the minimum requirements of a given password-composition policy. The remainder of the functions are used in our heuristic scoring.

¹Available at https://github.com/cupslab/password_meter

globalarrays.js To improve the readability of `rulefunctions.js`, this file contains large arrays of strings used by some of the heuristic scoring functions.

D.2.3 Files Related to Wordlists

lz-string.js This file, developed by a third party [141], implements Lempel-Ziv-Welch (LZW) lossless data compression in Javascript. We use this file to decompress the compressed wordlist.

englishwords-compressed.txt A compressed version of 130,517 frequently used English words taken from the intersection of the BYU Corpus of Contemporary American English (COCA) [50], the UNIX dictionary, and the 100,000 top single words (1-grams) used on Wikipedia [200].

names-compressed.txt A compressed version of 4,937 male and female names popular in the United States, per recent census data.

passwords-compressed.txt A compressed version of the 87,143 most common passwords (by frequency) seen in the Xato corpus of 10 million passwords [34].

phrases-compressed.txt The 49,927 unique entries after filtering out non-ascii and spaces from the 100,000 top 3-word phrases (3-grams) used on Wikipedia [200].

D.2.4 Files Related to Neural Networks

nn-client.min.js The main file for instantiating our artificial neural networks for calculating password guessability. This file loads the other files, below, as needed. Therefore, only this file needs to be explicitly included in the primary page.

basic_3M.info_and_guess_numbers.no_bloomfilter.json A JSON encoding of a pre-computed mapping of estimating a password's guess number from its probability by using Monte Carlo methods. This is a companion file to the one that follows.

basic_3M.weight_arch.quantized.fixed_point1000.zigzag.nospace.json A JSON encoding of the artificial neural network we computed using a 3 MB (before optimizations and compression) network in which probabilities have been quantized and stored using fixed-point encoding and ZigZag encoding. The weights have been quantized to three decimal digits, as well. This model ignores letter capitalization, which must be post-processed.

worker.min.js To calculate neural network guess numbers asynchronously (and thereby avoid causing the interface to lag), this file uses the WebWorker framework [133] to spawn separate threads for calculating guess numbers for a password using neural networks.

