Password Assistance

Moritz Horsch, Johannes Braun, and Johannes Buchmann[1]

Abstract: For decades, users are not able to realize secure passwords for their user accounts at Internet services. Users' passwords need to fulfil general security requirements and the password requirements of services. Furthermore, users need to cope with different password implementations at services. Finally, users need to perform a multitude of tasks to properly manage their large password portfolios. This is practically impossible.
In this paper, we introduce the vision of a *password assistant*. It supports users in all duties and tasks with regard to their passwords, from the creation of secure passwords to the recovery of them in case of loss. Moreover, it provides an extensive automatization of all password tasks that reduces the users' efforts and activities to deal with passwords to a minimum. A password assistant enables high security for passwords as well as improves their ease of use. First, we provide a detailed description of the problem of users to realize secure passwords for their accounts in practice. Second, we outline the vision of a password assistant, describe its technical foundation, and introduce the related open-source project starting to realize the first password assistant.

Keywords: Passwords, password assistant, password backup, password change, password emergency access, password generation, password management, password problem, password security, password synchronization.

1 Introduction

User accounts at Internet services contain a multitude of personal and privacy-sensitive data such as emails, documents, pictures, and payment information. To protect this data, access to user accounts is usually restricted by password-based authentication. The variety of known attacks against passwords, like brute-force [KKM$^+$12, WAdMG09], dictionary [Bo12, WACS10], and social engineering [CADaP13], makes it indispensable to use *secure* passwords.

However, users are not able to realize secure passwords for their user accounts for decades [HvO12, THB14]. First, users need to generate and manage passwords according to general security requirements. The passwords must be hard to guess, should not be reused across accounts, and finally changed regularly [Un17]. This requires users to memorize a large password portfolio, which is practically impossible [FHvO14b]. Second, users need to cope with different password implementations at services on the Internet. Services have different password requirements as well as different password interfaces and procedures for login, password change, and password reset [BP10]. This make the management of a password portfolio a very cumbersome task, particularly with regard to changing pass-

[1] Technische Universität Darmstadt, Hochschulstraße 10, 64289 Darmstadt, {horsch | jbraun | buchmann}@cdc.informatik.tu-darmstadt.de

words regularly [GC17]. To address both issues, we need to provide a solution that relieves users from memorizing and manually managing their password portfolios.

A first step towards a practical solution is to create passwords randomly and store them on a user device, as done by password managers. However, this approach poses the problem that the passwords might get lost and not be available on all user devices, particularity in urgent or even emergency situations [KSC10, SB14]. As long as we do not provide a secure and usable solution comprising all aspects of passwords, users are not able to use secure passwords for their accounts without restrictions. The aspects of passwords include the security requirements and conditions of services as well as all the password tasks of users from the generation of a password to it's recovery in case of loss.

In this paper, we introduce the vision of a *password assistant*. It supports users in all their tasks with passwords. It assists in generating proper passwords, in making them available on all devices, in creating a backup, in recovering them in case loss, and so on. A password assistant provides an extensive automatization of all password tasks and thus reduces the efforts and activities of users to deal with passwords to a minimum. This allows to enable high security for passwords as well as to improve the ease of use of passwords. A password assistant is solely realized on the user-side and completely transparent to services. We lunched the open-source *password assistance project* to realize the password assistant.

The remainder of the paper is structured as follows: Section 2 contains related work. In Section 3, we provide our first contribution, a detail description of the problem of users to realize secure passwords for their accounts. In Section 4, we present our second contribution and outline our vision of a password assistant. Moreover, we describe the technical foundation of its realization. Section 5 concludes this paper.

2 Related work

There exist various proposals to force or rather encourage users to generate secure passwords: Password requirements [SKK+10, KSK+11], passwords meters [UKK+12, ESM+13], system-assigned passwords [SKK+12], passphrases [KSS07], password generators [AHJS15]. However, they do not address the issue of keeping the passwords. This is crucial as nobody can memorize a multitude of secure passwords.

Moreover, there exist numerous proposals to keep the generated passwords: memorizing passwords (using repetition [BS14], cues [AWS15, BBD13] and, training videos [HAAWS17]), reusing passwords for different types of accounts [FHvO14b], and password managers [KSC10]. Only password managers provide a practical solution with regard to securely keeping a large password portfolio. But, they also raise the issues of password recoverability, availability, and accessibility [KSC10, HBMB17].

Storing passwords in the cloud is the usual approach to (1) make passwords available on all devices, (2) have a backup in case of device loss, (3) provide an emergency access to trusted persons in urgent situations. However, storing passwords in the cloud bears the risk of server breaches (e.g. as happened in case of LastPass [La11, La15] and OneLogin

[On17]) and finally to the leakage of the users' entire password portfolios. There are also first solutions to change passwords automatically [La17, Da17, MBV16], but they struggle with the different password implementations at services.

3 The password problem

In this section, we detail the problem of users to realize secure passwords for their accounts in practice. We examine three aspects of the problem: First, we analyze in Section 3.1 security requirements for passwords. Second, in Section 3.2 we describe the conditions of services for passwords. Finally, we enumerate the various password tasks of users to realize secure passwords with respect to the security requirements and the service conditions.

3.1 Security requirements

In this section, we describe our attacker model and specify security requirements for passwords to thwart the attacker and finally realize secure passwords for online accounts.

We consider an attacker who knows the user's username and tries to obtain the user's passwords to access his account at a service. We expect that the attacker can perform an *online guessing* attack by repeatedly attempting to log in to the account with a different password. We also expect that the attacker can perform an *offline guessing* attack by systematically trying all possible passwords and compare them to the salted and hashed user password. The preliminary for this attack is that the attacker had compromised the password database of the service and had stolen the hashed password and the used salt [MC13]. The attacker can run brute-force attacks, dictionary attacks [MT79], and highly optimized attacks based on probabilistic models [MYLL14, WAdMG09, NS05, MUS$^+$16].

We expect that the attacker does not simply obtain the password in plain-text, neither form the service nor form the user. Consequently, we expect that services implement at least a basic protection for the stored passwords and store them in a salted and hashed manner [FHvO14a]. Moreover, we expect that a user does not divulge his password through social engineering [HMS16], phishing [Ah17], or malware [FFC$^+$11].

With respect to the attacker's capabilities, we now define requirements for passwords:

R1 *Brute-force-resistant password.* The password must have a security level of at least 128 bits to withstand brute-force attacks [Bl]. If not feasible, e.g. due to the service's password requirements, the highest possible security level should be used.

R2 *Individual password.* The password for each account must be unique. It should neither be used across accounts nor reused (even not partially) for password changes. This prevents an attacker from using a stolen password to access other accounts.

R3 *Changing password.* The password must be changed (1) on a regular basis and (2) immediately after a security incident at an account or a service. This invalidates a compromised password as well as addresses the issue of “aging of cryptography” (e.g. the hash function used for storing the passwords becomes insecure).

3.2 Service conditions

In practice the usage of passwords is bounded by the conditions of services. In the following, we outline these conditions.

Password requirements Password requirements are fixed rules with respect to the password length and the allowed and/or required characters. Services implement password requirements to guide users to select strong passwords and to prevent incompatibilities with their password implementations. In practice, users can only select passwords for their accounts that comply with the password requirements of services. Various studies have documented that the requirements are highly diverging between services [ATJK12, BP10, FH10, Fu07, Fu11, WW15]. Due to missing official standards, it is very unlikely that this situation will change in the future.

Password interfaces and procedures Beside the selection of passwords, also the usage of passwords is bound by services. A password at a service can only be used through the service’s interfaces. In practice, this is the HTML form for login, password change, and password reset at the service’s website. Moreover, the interaction with these interfaces is predefined by certain procedures. The password interface and password procedures at services are quite different. This makes particularly a regular password change a very cumbersome task. For instance, in case of Google or ebay, users are redirected to the login form to conform the knowledge of the current password when they want to change their password. But, at ebay users need to enter the current password at the password change form again, summing up to enter the current password three times to actually change it (at the login, before changing, and at the password change form). In comparison, wikipedia users just need to do it once, during the login.

3.3 Password tasks

In this section, we enumerate the password tasks of users to realize secure passwords for their accounts. These tasks ranging from the generation of a password to its recovery in case of loss. In turns out that it is practically impossible to do these tasks manually.

We consider the following two real-world usage conditions of Internet users: First, users have numerous accounts. Due to security requirement R1 and R2 users have a large portfolio of passwords, which is practically impossible to memorize. To this end, we assume that users store their password on their devices. Yet, storing passwords instead of memorizing them bears the risk of loss, e.g., in situations where the device on which the passwords are stored is damaged, lost, or stolen. This makes the creation of backups of the passwords indispensable. Second, users have multiple devices. Therefore, it is necessary that the passwords are available on all their devices, i.e. get synchronized between them. Otherwise, users cannot access their accounts at anytime and from anywhere. Addressing these issues are crucial for users [KSC10, SB14, ZY14].

Moreover, we are focusing on a pure user-side solution for realizing secure passwords. Against this background, we do not consider means of reseting passwords provided by services to solve the loss of passwords, like sending a new password to a recovery email address or mobile number. Note that such reset mechanism have proven not to be reliable and secure [BBC$^+$15, GKMP17]. We address the issue of password loss by creating a backup. This approach has the two advantages that (1) it does not rely on services properly implementing password reset procedures and (2) it leaves the user in control of his passwords.

In the following, we enumerate the password task of users:

T1 *Password generation*. The first password task of a user is to generate a password. It must comply with security requirement R1 and R2. In brief, it must be brute-force-resistant and not already been used for another account. Moreover, it must comply with the password requirements of the service. If the password is too short, too long, or does not contain all expected characters, it will be rejected by the service.

T2 *Password storage*. The second task for a user is to store the password. Beside the simple action of saving the password in a file or designed application like an password manager, the user must also protect the stored password from unauthorized access. This is usually done by another password, a so-called *master password*, which leads to three subtasks: First, the user needs to generate a master password with respect to security requirement R1 and R2. Second, the user needs to memorize it. Third, the user should change it regularly to fulfil security requirement R3.

T3 *Password change*. The third user task is to change the password regularly. This is deduced from security requirement R3. Changing the password of an account consists of three subtasks: First, the user must generate a new password, which leads to T1. Second, he needs to log in to the account and change the password. This can only be done through the individual—and often different—password interface and procedure of the service. Third, the user must store the new password, which leads to T2. Note that this task must be performed for the password of each account of the user, making this a very cumbersome task.

T4 *Password availability*. The forth password task of a user is to make his password available on all his devices. This is necessary to be able to access the account at anytime and form anywhere. In case the user changes the password the task must be repeated. This is necessary to always have the current password on all devices.

T5 *Password recoverability*. The fifth task of a user is to create a backup of his password. This is essential to recover it in situations where his device—on which the password is stored—is damaged, lost, or stolen. The creation of a password backup includes three subtasks: First, the user must protect the backup from unauthorized access. The protection must have the same security level as used for the stored password. Second, the user must place the backup at a secure location to protect it from loss as well. With respect to malware or physical damage like fire a different and distant location is important, e.g. at a friend's place. Third, the user must keep the password backup up-to-date as a backup containing an outdated password is useless. Creating and updating a backup should be done immediately after a password is stored. This can happen frequently due to security requirement R3 demanding to change passwords regularly.

T6 *Password accessibility*. The last password task of a user is to provide an emergency access to his password. It enables the user to grant access to his password to a trusted person in urgent or even emergency situations. For instance, after a robbery or natural disaster, it is essential that a user can ask a friend to log in to his accounts and send an email or get a copy of his identity card stored at a cloud storage provider. To set up and maintain the emergency access the user needs to perform three subtasks: First, the user needs to make the password available for the trusted person. Second, the user needs to ensure that the trusted person always have the current user's password. Having an outdated password is useless. Third, the user must limit the emergency access to the password for the trusted person and must approve the access when actually done.

These tasks allow users to realize secure passwords for their accounts. They cover the real-world conditions of users and all aspects of passwords. However, manually performing these tasks is practically impossible for users. These is extensively documented by various user studies [LV07, MKV+13, ZMR10, KSK+11, SKD+14, CJE+06, DBC+14, GF06, SKK+10, SB14, UNB+15, Bo12, KKM+12, FHvO14b, FH07, HWF05, YBAG04, ZH99, SKD+16].

4 Password assistant

In the previous section, we detailed the problem of users to realize secure passwords for online accounts. The security requirements lead to generate and manage a large portfolio of passwords. The various service conditions require to figure out how a password must look like and how it can be managed at each service. Finally, all the passwords tasks need to be performed to realize secure passwords. This is practically impossible for users.

We propose a *password assistant* to solve this problem. It supports users in all their password tasks, from the creation of a secure password to the recovery of it in case of loss. Moreover, it provides an extensive automatization of these tasks which reduces the efforts and activities of users to deal with passwords to a minimum. The password assistant provides four key features: First, an automatic generation of secure and valid passwords for users. Second, an automatic and proactive change of their passwords. Third, a synchronization of the passwords between all the devices of a user. Fourth, a secure backup of the passwords as well as a built-in revocation mechanism and an emergency access for backups. These four features provide a comprehensive and ubiquitous assistance for users in the usage of passwords. It makes the generation and management of passwords as easy as possible and finally enables users to realize secure passwords for their accounts without restrictions. In the following sections, we describe the technical foundation of the four features in detail.

The password assistant is realized as an application running on all user devices. It stores the users' secrets on their devices, protect them by a master password (cf. T2), and provides various conveniences features like filling out login forms, and so forth.

4.1 Automatic generation of brute-force-resistant and valid passwords

Users need to generate brute-force-resistant passwords for their accounts while coping with the different passwords requirements (cf. T1). The password assistant solves this challenge by generating such passwords automatically for them. Users just need to provide the URL of the service. The password generation is realized by means of a password generator that generates random passwords in accordance with the individual password requirements of services. To ensure the compliance with the services' password requirements, we make the individual password requirements of services available to the password assistant and we provide optimal fallback password-composition rules for the case that no explicit requirements are available. Thus, the password assistant automatically creates passwords with the best-possible security level and acceptance rate.

The password generation is realized by four building blocks:

- *A standardized description of password requirements that can be processed by the password assistant.* We developed the Password Requirements Markup Language (PRML) which is used to create Password Requirements Descriptions (PRDs) for services [HSH$^+$16]. A PRD allows to specify the different password lengths, allowed character sets, and further password restrictions of a service. It provides all necessary information to automatically generate a brute-force-resistant password compliant to the password requirements of a service.

- *The automatic generation of PRDs for the numerous services that already exists on the Internet.* We developed the Password Requirements Crawler (PRC), an application that extracts password requirements from a service's website and generates the corresponding PRD. We coped with the different expressions of password requirements by using *Natural Language Processing* techniques and used the PRC to create PRDs of 185,696 services.

- *The distribution of the PRDs to make them available to the password assistant.* We developed the PRD Distribution Service (PRDDS), a central service where the password assistant can search for and retrieve PRDs.

- *Optimal fallback password-composition rules for the password assistant.* Based on the password requirements of 185,696 services, we developed optimal fallback password-composition rules that the password assistant can use in case that the password requirements for a service are partially or entirely not available. By using letters and number and 22 characters we generate passwords with a security level of 130 bits and an acceptance rate by services of 76.6% [HBB17].

4.2 Automatic and proactive password change

The third user task is to change passwords regularly (cf. T3). The password assistant solves this challenge by changing passwords on behalf of users automatically. Users neither need to create a new password nor to log in to their accounts. The password assistant does this for users, e.g. on a weekly basis. The key pillar of this solution is a standardized description of services' password requirements, procedures, and interfaces. This facilitates the password assistant, among other things, to log in to an account and change the password.

The password change is realized by three building blocks:

- *A uniform description of the password policies*. Based on PRML, we developed the Password Policy Markup Language (PPML) to create Password Policy Descriptions (PPDs) for services [HSH^{+}16]. A PRD allows to describe the different password interfaces and procedures of services and make them available to the password assistant.
- *A support tool to create PPDs for existing services*. We developed a tool that records password procedures, like logging into an online account or changing a password, and creates a corresponding PPD. Our tool makes the creation of PPDs as easy as possible and is particularly designed for non-technical users.
- *The distribution of the PPDs to make them available to the password assistant*. We enhanced our solution developed for PRDs and developed a central service for distributing PPDs, where the password assistant can search for and retrieve PPDs.

Beside changing passwords automatically for users, this feature allows the password assistant to make the usage of secure passwords in practice as easy as possible. A problem of secure passwords is to currently enter them on other devices. A typical situation is accessing an account from another device that does not belong to the user and thus his password is not available on the device and must be manually entered. Beside the inconvenience for the user, this is very error-prone particularly in case the password consists of ambiguous characters [SKD^{+}16]. Moreover, there is the threat of malware on the other device. The password assistant solves this problem by temporary changing the account password to a shorter and less complex one so that it can easily be entered on the other device. After service usage, the assistant automatically changes the password back to a secure one.

4.3 Passwordless and scalable password synchronization

Users need to make their passwords available on all devices (cf. T4). The password assistant uses the PALPAS scheme [HHB15] to accomplish this task in a secure way. PALPAS makes the users' passwords available on all their devices without storing them on servers or user devices. To this end, adversaries cannot steal the passwords by compromising servers. Moreover, users can prevent adversaries from stealing their passwords from their devices in common situations like theft or loss.

PALPAS generates a password for an online account from a high entropy secret and a random salt value. The secret is shared by all user devices and the salt differs for each account to achieve individual passwords for accounts. Only the salt values are stored on a synchronization server but not the secret. The salt values on the one hand enable all user devices to generate the same passwords, while on the other hand they are useless for adversaries because they are statistically independent of the passwords. When creating new or changing passwords, the corresponding salts are added or updated at the synchronization server. Each time a password is needed, e.g. for a login, the corresponding salt for the account is retrieved from the synchronization server and the password is generated. In this way, any changes of the user's passwords are immediately available on all of his devices.

The user authentication at the synchronization server is realized using certificates. This has the advantage that also no passwords for user authentication are stored on the server. A certificate to access the salts at the server is stored on each user device and protected by a user-chosen master password. As such passwords usually not that secure, the password assistant uses a PBKDF to derive a strong encryption key from the master password. But, as this only provides a first line of defense, the password assistant also encrypts the secret with an one-time-pad. The one-time-pad key is stored on the synchronization server. In case of device loss, a user can use one of his other devices and revoke the certificate as well as the one-time-pad key. This makes it impossible to recover the secret from a stolen device or access the salts at the synchronization server.

4.4 Update-tolerant and revocable password backup

The last two user task are the creation of password backups and the provision of an emergency access to the passwords (cf. T5 and T6). The password assistant makes use of the PASCO scheme [HBMB17], which provides a secure and usable backup solution for PALPAS. It ensures that users never lose their password portfolio by providing recoverability of the essential PALPAS data that is stored on the user device.

PASCO provides three key features: First, password backups do not have to be updated even when the user's passwords change. Consequently, users need to create a backup only once and can keep it completely offline in secure, different, and physical isolated location which minimizes the risk of compromise and loss. Second, PASCO backups have a built-in revocation mechanism. It allows users to completely invalidate a backup in case they lose control over it. The revocation mechanism works without having access to the backup itself and guarantees that no passwords can be leaked from it once revoked. Third, PASCO backups provide a fully controllable emergency access. Users can authorize someone else to access a backup and obtain a set of pre-defined passwords in emergency situations.

PASCO uses a separate backup device to store a backup of the PALPAS data. We consider the backup device to be a tamper resistant device that provides secure storage, user authentication, and basic cryptographic algorithms. Similar to a user device it stores the PALPAS secret encrypted by a one-time-pad. Furthermore, it has its own certificate for the PALPAS synchronization server.

The backup device is protected by a user-chosen PIN. To prevent guessing attacks, it has a retry counter for the PIN. After five wrong PIN entries the device erases all stored data. We successfully implemented PASCO using a smart card as a backup device [HBMB17].

Moreover, the password assistant use the PASCO scheme to provide an emergency access to the users' passwords. This is realized in the following way. Beside storing the PALPAS data, the backup device is capable of generating passwords using the PALPAS password generation procedure [HHB15]. Furthermore, the PALPAS synchronization server is equipped with a fine granular access control system for the salts. This allows a user to specify different access rules to the salts. While, one backup device may have access to all salts, another device can only access the salts for the user's mail account. A user now can create a backup devices and place it at at friend's location. In case of an emergency, the user can call the friend and tell him the PIN. The friend is than capable to generate the user's password(s) and access his accounts. Note that even this backup must not be updated even when the user change his passwords. By this the password assistant provides a secure and usable password backup and emergency access solution to users.

5 Conclusion

The problem of realizing secure passwords for online accounts remains for decades. We presented an extensive analysis of this problem to identify all relevant aspects and propose a solution. We identified the security requirements, the services conditions, and the various password tasks of users to realize secure passwords. While existing solutions only consider parts of these aspects, we presented the first ubiquitous solution: a password assistant. It automatically generates brute-force-resistant and valid passwords, it automatically changes them, it makes them available on all user devices without storing them in the cloud, it provides a secure and usable backup solution, and finally it offers a fully controllable emergency access to the passwords. This is solely realized on the client-side and completely transparent to services, which is key for a practical solution as we cannot rely on any changes on the service-side regarding the implementation of passwords.

To realize our vision of the password assistant, we lunched the *password assistance project*. It will provide an open-source implementation of the technical foundation of the password assistant. The individual parts can be used by themselves as well as integrated into existing solutions. For instance, all existing password generators can use PRDs to automatically generate valid passwords. Finally, the combination of all parts build the password assistant.

References

[Ah17] Ahmed Aleroud and Lina Zhou. Phishing environments, techniques, and countermeasures: a survey. *Computers and Security*, 2017.

[AHJS15] Steven Van Acker; Daniel Hausknecht; Wouter Joosen; Andrei Sabelfeld. Password Meters and Generators on the Web: From Large-Scale Empirical Study to Getting It Right. In *Proc. CODASPY*. ACM, 2015.

[ATJK12] Bander AlFayyadh; Per Thorsheim; Audun Jøsang; Henning Klevjer. Improving usability of password management with standardized password policies. In *Proc. SAR-SSI*, 2012.

[AWS15] Mahdi Nasrullah Al-Ameen; Matthew K. Wright; Shannon Scielzo. Towards Making Random Passwords Memorable: Leveraging Users' Cognitive Ability Through Multiple Cues. In *Proc. CHI*. ACM, 2015.

[BBC$^+$15] Joseph Bonneau; Elie Bursztein; Ilan Caron; Rob Jackson; Mike Williamson. Secrets, Lies, and Account Recovery: Lessons from the Use of Personal Knowledge Questions at Google. In *Proc. WWW*. ACM, 2015.

[BBD13] Jeremiah Blocki; Manuel Blum; Anupam Datta. Naturally Rehearsing Passwords. In *Proc. ASIACRYPT*. Springer, 2013.

[Bl] BlueKrypt. Cryptographic Key Length Recommendation. `https://www.keylength.com`.

[Bo12] Joseph Bonneau. The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. In *Proc. SP*. IEEE, 2012.

[BP10] Joseph Bonneau; Sören Preibusch. The Password Thicket: Technical and Market Failures in Human Authentication on the Web. In *Proc. WEIS*, 2010.

[BS14] Joseph Bonneau; Stuart E. Schechter. Towards Reliable Storage of 56-bit Secrets in Human Memory. In *Proc. USS*. USENIX Association, 2014.

[CADaP13] Claude Castelluccia; Chaabane Abdelberi; Markus Dürmuth; aniele Perito. When Privacy meets Security: Leveraging personal information for password cracking. *CoRR*, abs/1304.6584, 2013.

[CJE$^+$06] Luke St. Clair; Lisa Johansen; William Enck; Matthew Pirretti; Patrick Traynor; Patrick D. McDaniel; Trent Jaeger. Password Exhaustion: Predicting the End of Password Usefulness. In *Proc. ICISS*. Springer, 2006.

[Da17] Dashlane, Inc. Best Password Manager, Free Form Filler, Secure Digital Wallet, 2017. `https://www.dashlane.com`.

[DBC$^+$14] Anupam Das; Joseph Bonneau; Matthew Caesar; Nikita Borisov; XiaoFeng Wang. The Tangled Web of Password Reuse. In *Proc. NDSS*. The Internet Society, 2014.

[ESM$^+$13] Serge Egelman; Andreas Sotirakopoulos; Ildar Muslukhov; Konstantin Beznosov; Cormac Herley. Does my password go up to eleven?: the impact of password meters on password selection. In *Proc. CHI*. ACM, 2013.

[FFC$^+$11] Adrienne Porter Felt; Matthew Finifter; Erika Chin; Steve Hanna; David Wagner. A survey of mobile malware in the wild. In *Proc. SPSM@CCS*, 2011.

[FH07] Dinei A. F. Florêncio; Cormac Herley. A Large Scale Study of Web Password Habits. In *Proc. WWW*. ACM, 2007.

[FH10] Dinei Florêncio; Cormac Herley. Where do security policies come from? In *Proc. SOUPS*. USENIX Association, 2010.

[FHvO14a] Dinei Florêncio; Cormac Herley; Paul C. van Oorschot. An Administrator's Guide to Internet Password Research. In *Proc. LISA*. USENIX Association, 2014.

[FHvO14b] Dinei Florêncio; Cormac Herley; Paul C. van Oorschot. Password Portfolios and the Finite-Effort User: Sustainably Managing Large Numbers of Accounts. In *Proc. USS*. USENIX Association, 2014.

[Fu07] Steven Furnell. An assessment of website password practices. *Computers and Security*, 26(7-8), 2007.

[Fu11] Steven Furnell. Assessing password guidance and enforcement on leading websites. *Computer Fraud and Security*, 2011(12), 2011.

[GC17] Kristen K. Greene; Yee-Yin Choong. Must I, can I? I don't understand your ambiguous password rules. *Inf. & Comput. Security*, 25(1), 2017.

[GF06] Shirley Gaw; Edward W. Felten. Password management strategies for online accounts. In *Proc. SOUPS*. USENIX Association, 2006.

[GKMP17] Nethanel Gelernter; Senia Kalma; Bar Magnezi; Hen Porcilan. The Password Reset MitM Attack. In *Proc. SP*. IEEE, 2017.

[HAAWS17] S. M. Taiabul Haque; Mahdi Nasrullah Al-Ameen; Matthew Wright; Shannon Scielzo. Learning system-assigned passwords (up to 56 bits) in a single registration session with the methods of cognitive psychology. In *Proc. USEC*. The Internet Society, 2017.

[HBB17] Moritz Horsch; Johannes Braun; Johannes Buchmann. The Wide Diversity of Password Requirements – And how to cope with it. In *(in submission)*, 2017.

[HBMB17] Moritz Horsch; Johannes Braun; Dominique Metz; Johannes Buchmann. Update-tolerant and Revocable Password Backup. In *Australasian Conference on Information Security and Privacy (ACISP)*, Lecture Notes in Computer Science. Springer, 2017.

[HHB15] Moritz Horsch; Andreas Hülsing; Johannes Buchmann. PALPAS - PAsswordLess PAssword Synchronization. In *International Conference on Availability, Reliability and Security (ARES)*. IEEE Computer Society, 2015.

[HMS16] Christian Happ; André Melzer; Georges Steffgen. Trick with treat - Reciprocity increases the willingness to communicate personal data. *Computers in Human Behavior*, 61, 2016.

[HSH$^+$16] Moritz Horsch; Mario Schlipf; Stefen Haas; Johannes Braun; Johannes Buchmann. Password Policy Markup Language. In *Open Identity Summit (OID)*, Lecture Notes in Informatics. German Informatics Society, 2016.

[HvO12] Cormac Herley; Paul C. van Oorschot. A Research Agenda Acknowledging the Persistence of Passwords. *Security and Privacy*, 10(1), 2012.

[HWF05] J. Alex Halderman; Brent Waters; Edward W. Felten. A Convenient Method for Securely Managing Passwords. In *Proc. WWW*. ACM, 2005.

[KKM$^+$12] Patrick Gage Kelley; Saranga Komanduri; Michelle L. Mazurek; Richard Shay; Timothy Vidas; Lujo Bauer; Nicolas Christin; Lorrie Faith Cranor; Julio Lopez. Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms. In *Proc. SP*. IEEE, 2012.

[KSC10] Ambarish Karole; Nitesh Saxena; Nicolas Christin. A Comparative Usability Evaluation of Traditional Password Managers. In *Proc. ICISC*. Springer, 2010.

[KSK$^+$11] Saranga Komanduri; Richard Shay; Patrick Gage Kelley; Michelle L. Mazurek; Lujo Bauer; Nicolas Christin; Lorrie Faith Cranor; Serge Egelman. Of passwords and people: measuring the effect of password-composition policies. In *Proc. CHI*. ACM, 2011.

[KSS07] Mark Keith; Benjamin Shao; Paul John Steinbart. The usability of passphrases for authentication: An empirical field study. *International Journal of Man-Machine Studies*, 65(1), 2007.

[La11] LastPass Corporate. LastPass Security Notification, Mai 2011. `https://blog.lastpass.com/2011/05/lastpass-security-notification.html/`.

[La15] LastPass Corporate. LastPass Security Notification, Juni 2015. `https://blog.lastpass.com/2015/06/lastpass-security-notice.html/`.

[La17] LastPass Corporate. LastPass - The Last Password You Have to Remember, 2017. `https://lastpass.com`.

[LV07] Michael D. Leonhard; V. N. Venkatakrishnan. A Comparative Study of Three Random Password Generators. In *Proc. EIT*. IEEE, 2007.

[MBV16] Peter Mayer; Hermann Berket; Melanie Volkamer. Enabling Automatic Password Change in Password Managers Through Crowdsourcing. In *Proc. PASSWORDS*. Springer, 2016.

[MC13] Dennis Mirante; Justin Cappos. Understanding password database compromises. *TR-CSE-2013-02*, 2013.

[MKV$^+$13] Michelle L. Mazurek; Saranga Komanduri; Timothy Vidas; Lujo Bauer; Nicolas Christin; Lorrie Faith Cranor; Patrick Gage Kelley; Richard Shay; Blase Ur. Measuring password guessability for an entire university. In *Proc. CSS*. ACM, 2013.

[MT79] Robert Morris; Ken Thompson. Password Security - A Case History. *CACM*, 22(11), 1979.

[MUS$^+$16] William Melicher; Blase Ur; Sean M. Segreti; Saranga Komanduri; Lujo Bauer; Nicolas Christin; Lorrie Faith Cranor. Fast, Lean, and Accurate: Modeling Password Guessability Using Neural Networks. In *Proc. USS*. USENIX Association, 2016.

[MYLL14] Jerry Ma; Weining Yang; Min Luo; Ninghui Li. A Study of Probabilistic Password Models. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2014.

[NS05] Arvind Narayanan; Vitaly Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *Proc. CCS*. ACM, 2005.

[On17] OneLogin, Inc. Security Incident, Mai 2017. `https://www.onelogin.com/blog/may-31-2017-security-incident`.

[SB14] Elizabeth Stobert; Robert Biddle. The Password Life Cycle: User Behaviour in Managing Passwords. In *Proc. SOUPS*. USENIX Association, 2014.

[SKD$^+$14] Richard Shay; Saranga Komanduri; Adam L. Durity; Phillip (Seyoung) Huh; Michelle L. Mazurek; Sean M. Segreti; Blase Ur; Lujo Bauer; Nicolas Christin; Lorrie Faith Cranor. Can long passwords be secure and usable? In *Proc. CHI*. ACM, 2014.

[SKD$^+$16] Richard Shay; Saranga Komanduri; Adam L. Durity; Phillip (Seyoung) Huh; Michelle L. Mazurek; Sean M. Segreti; Blase Ur; Lujo Bauer; Nicolas Christin; Lorrie Faith Cranor. Designing Password Policies for Strength and Usability. *TISSEC*, 18(4), 2016.

[SKK$^+$10] Richard Shay; Saranga Komanduri; Patrick Gage Kelley; Pedro Giovanni Leon; Michelle L. Mazurek; Lujo Bauer; Nicolas Christin; Lorrie Faith Cranor. Encountering stronger password requirements: user attitudes and behaviors. In *Proc. SOUPS*. ACM, 2010.

[SKK$^+$12] Richard Shay; Patrick Gage Kelley; Saranga Komanduri; Michelle L. Mazurek; Blase Ur; Timothy Vidas; Lujo Bauer; Nicolas Christin; Lorrie Faith Cranor. Correct horse battery staple: exploring the usability of system-assigned passphrases. In *Proc. SOUPS*. ACM, 2012.

[THB14] Viktor Taneski; Marjan Hericko; Bostjan Brumen. Password security - No change in 35 years? In *Proc. MIPRO*. IEEE, 2014.

[UKK$^+$12] 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. USS*. USENIX Association, 2012.

[Un17] United States Computer Emergency Readiness Team (US-CERT). National Cyber Awareness System - Tips, 2017. `https://www.us-cert.gov/ncas/tips`.

[UNB$^+$15] Blase Ur; Fumiko Noma; Jonathan Bees; Sean M. Segreti; Richard Shay; Lujo Bauer; Nicolas Christin; Lorrie Faith Cranor. ”I Added ’!’ at the End to Make It Secure”: Observing Password Creation in the Lab. In *Proc. SOUPS*. USENIX Association, 2015.

[WACS10] Matt Weir; Sudhir Aggarwal; Michael P. Collins; Henry Stern. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proc. CCS*. ACM, 2010.

[WAdMG09] Matt Weir; Sudhir Aggarwal; Breno de Medeiros; Bill Glodek. Password Cracking Using Probabilistic Context-Free Grammars. In *Proc. SP*. IEEE, 2009.

[WW15] Ding Wang; Ping Wang. The Emperor’s New Password Creation Policies: An Evaluation of Leading Web Services and the Effect of Role in Resisting Against Online Guessing. In *Proc. ESORICS*. Springer, 2015.

[YBAG04] Jeff Jianxin Yan; Alan F. Blackwell; Ross J. Anderson; Alasdair Grant. Password Memorability and Security: Empirical Results. *IEEE Security & Privacy*, 2(5), 2004.

[ZH99] Moshe Zviran; William J. Haga. Password Security: An Empirical Study. *Journal of Management Information Systems*, 15(4), 1999.

[ZMR10] Yinqian Zhang; Fabian Monrose; Michael K. Reiter. The security of modern password expiration: an algorithmic framework and empirical analysis. In *Proc. CCS*. ACM, 2010.

[ZY14] Rui Zhao; Chuan Yue. Toward a secure and usable cloud-based password manager for web browsers. *Computers and Security*, 46, 2014.