Strength of Passwords

This appendix is informative.

This appendix uses the word “password” for ease of discussion. Where used, it should be interpreted to include passphrases, PINs, and passwords.

Introduction

Passwords are a widely used form of authentication despite concerns about their use from both a usability and security standpoint [Persistence]. Humans have a limited ability to memorize complex, arbitrary secrets, so they often choose passwords that can be easily guessed. To address the resultant security concerns, online services have introduced rules to increase the complexity of these passwords. The most notable form is composition rules, which require users to choose passwords that are constructed using a mix of character types (e.g., at least one digit, uppercase letter, and symbol). However, analyses of breached password databases reveal that the benefit of such rules is less significant than initially thought [Policies], and the impacts on usability and memorability are severe.

The complexity of user-chosen passwords has often been characterized using the information theory concept of entropy [Shannon]. While entropy can be readily calculated for data with deterministic distribution functions, estimating the entropy for user-chosen passwords is challenging, and past efforts to do so have not been particularly accurate. For this reason, a different and somewhat more straightforward approach based primarily on password length is presented herein.

Many attacks associated with password use are not affected by password complexity and length. Keystroke logging, phishing, and social engineering attacks are equally effective on lengthy and complex passwords as they are on simple ones. These attacks are outside of the scope of this Appendix.

Length

Password length is a primary factor in characterizing password strength [Strength] [Composition]. Passwords that are too short yield to brute-force attacks and dictionary attacks. The minimum password length required depends on the threat model being addressed. Online attacks in which the attacker attempts to log in by guessing the password can be mitigated by limiting the permitted login attempt rate. To prevent an attacker (or a persistent claimant with poor typing skills) from quickly inflicting a denial-of-service attack on the subscriber by making many incorrect guesses, passwords need to be complex enough that a reasonable number of attempts can be permitted with a low probability of a successful guess, and rate limiting can be applied before there is a significant chance of a successful guess.

Offline attacks are sometimes possible when the attacker obtains one or more hashed passwords through a database breach. The ability of the attacker to determine one or more users’ passwords depends on how the password is stored. Commonly, passwords are salted with a random value and hashed, preferably using a computationally expensive algorithm. Even with such measures, the current ability of attackers to compute many billions of hashes per second in an offline environment that is not subject to rate limiting requires passwords to be orders of magnitude more complex than those expected to resist only online attacks.

Users should be encouraged to make their passwords as lengthy as they want, within reason. Since the size of a hashed password is independent of its length, there is no reason to prohibit the use of lengthy passwords (or passphrases) if the user wishes. Extremely long passwords (perhaps megabytes long) could require excessive processing time to hash, so it is reasonable to have some limit.

Complexity

Composition rules are commonly used in an attempt to increase the difficulty of guessing user-chosen passwords. However, research has shown that users respond in very predictable ways to the requirements imposed by composition rules [Policies]. For example, a user who might have chosen "password" as their password would be relatively likely to choose "Password1" if required to include an uppercase letter and a number or "Password1!" if a symbol is also required.

Users also express frustration when online services reject their attempts to create complex passwords. Many services reject passwords with spaces and various special characters. Characters that are not accepted are sometimes the result of an effort to avoid attacks that depend on those characters (e.g., SQL injection). However, an unhashed password would not be sent intact to a database, so such precautions are unnecessary. Users should also be able to include space characters to allow the use of phrases. Space characters add little to the complexity of passwords and may introduce usability issues (e.g., the undetected use of two spaces rather than one), so removing repeated spaces in typed passwords may be beneficial if initial verification fails.

Since users' password choices are often predictable, so attackers are likely to guess passwords that have previously proven successful. These include dictionary words and passwords from previous breaches, such as the "Password1!" example above. For this reason, passwords chosen by users should be compared against a blocklist of unacceptable passwords. This list should include passwords from previous breach corpuses, dictionary words used as passwords, and specific words (e.g., the name of the service itself) that users are likely to choose. Since a minimum length requirement will also govern the user's choice of passwords, this dictionary only needs to include entries that meet that requirement. As noted in Sec. 3.1.1.2, it is not beneficial for the blocklist to be excessively large or comprehensive, since its primary purpose is to prevent the use of very common passwords that might be guessed in an online attack before throttling restrictions take effect. An excessively large blocklist will likely frustrate users who attempt to choose a memorable password.

Highly complex passwords introduce a new potential vulnerability: they are less likely to be memorable and more likely to be written down or stored electronically in an unsafe manner. While these practices are not necessarily vulnerable, some methods of recording such secrets will be. This is an additional motivation for not requiring excessively long or complex passwords.

Central vs. Local Verification

While passwords that are used as a separate authentication factor are often centrally verified by the CSP's verifier, those that are used as an activation factor for a multi-factor authenticator are either verified locally by the authenticator or used to derive the authenticator output, which will be incorrect if the wrong activation factor is used. Both of these situations are referred to as "local verification."

The attack surfaces and vulnerabilities for central and local verification are very different. Accordingly, the requirements for centrally verified passwords differ from those verified locally. Centrally verified passwords require the verifier (i.e., an online resource) to store salted and iteratively hashed verification secrets for all of the subscribers' passwords. Although the salting and hashing process increases the computational effort to determine the passwords from the hashes, the verifier is an attractive target for attackers, particularly those interested in compromising an arbitrary subscriber rather than a specific one.

Local verifiers do not have the same concerns with large-scale attacks on a central online verifier but depend to a greater extent on the physical security of the authenticator and the integrity of its associated endpoint. To the extent that the authenticator stores the activation factor, that factor must be protected against physical and side-channel (e.g., power and timing analysis) attacks on the authenticator. When the activation factor is entered through the associated endpoint, the endpoint needs to be free of malware (e.g., key-logging software). Since such threats are less dependent on the length and complexity of the password, these requirements are relaxed for local verification.

Online password-guessing attacks are a similar threat to centrally and locally verified passwords. Throttling, which is the primary defense against online attacks, can be particularly challenging for local verifiers because of the limited ability of some authenticators to securely store information about unsuccessful attempts. Throttling can be performed by either keeping a count of invalid attempts in the authenticator or generating an authenticator output rejected by the CSP verifier, which does the throttling. In this case, the invalid outputs must not be evident to the attacker, who could otherwise make offline attempts until a valid-looking authenticator output appears.

Summary

Length and complexity requirements beyond those recommended here significantly increase the difficulty of using passwords and increase user frustration. As a result, users often work around these restrictions counterproductively. Other mitigations (e.g., blocklists, secure hashed storage, machine-generated random passwords, and rate limiting) are more effective at preventing modern brute-force attacks, so no additional complexity requirements are imposed.