CS808: Computer Security Fundamentals

4.6: Article: Password Security

Done: View

Password Security

When considering the security of passwords we need to address how passwords are generated, stored and attacked. In this article we will explore each of these in turn. Note that when we discuss passwords, we are referring to alphanumeric text based passwords.

Password Generation

Password generation can be challenging. Too simple makes it easy to recall, but also easy to attack. Too random and it makes it difficult to attack, but also difficult to recall.

In generating passwords you can be shown password strength meters. Take a moment and consider what a strong or secure password means to you, are there specific criteria? Often people will suggest a mix of upper and lower case letters, special characters and numbers as a “strong” password. Generally these work on the basis of calculating the entropy of the password.

Claude Shannon created the concept of entropy as applied to information compression. It is a measure of how few bits can be used to represent the data. The more random the data is, the more bits required to represent it. This concept was extrapolated to determine the redundancy in a given language, and then to determine the redundancy in a password. The idea being that a more random password has higher entropy and takes more processing power to crack.

Entropy is measured in bits and a randomly chosen password with length k bits, has 2^k possible values. Since in binary we have only two possible values for a bit (1 or 0) this provides the 2. If we look only at binary then k represents the entropy of the password. However, we are not using binary, we are using the English language and thus there are more possible values for a character.

If a password has a length l characters, and it’s chosen at random from an alphabet of n characters, then it has an entropy of n^l
To then transpose this into the entropy in bits we need to apply a log base 2 i.e. $log_2 n^l$ This provides the entropy of a password.

However this assumes we select randomly from the password space available. The password space is the set of all possible passwords. As users, we do not pick random passwords. In fact we are incredibly predictable in general. Have a look at the top passwords from password database compromises in 2020 here Note how password and 123456 are incredibly common, despite knowledge this is an insecure password.

There is a clear limitation of using entropy in this way, as it can skew the apparent security of a password. Take the example password123. I expect everyone recognises this is not a good password as it is easily guessed. However, the entropy calculation comes out at approx. 41 bits which implies reasonable strength.

How then do we calculate entropy of a password which is not randomly selected from the password space? If we knew all the passwords that people choose and the distribution of them, this would be relatively easy, but we do not.

As a result, a different option for measuring entropy resulted from further work by Shannon.

Shannon performed an experiment looking at the entropy of a 27 letter alphabet- lower case English plus a space character. People were asked to guess the next letter given the previous letter. It was observed it is difficult to guess the first letter of a string, but it gets progressively easier the more letters you have.

In fact, the result of this was that the first character contributed around 4.7 bits, the second to the eighth character 2.3 bits, and then any further characters contributed 1.5 bits each to overall entropy.

As you may deduce, the application to passwords followed this structure. 4.7 bits for the first character, characters 2-8 contributing 2.3 bits and further characters 1.5 bits. The sum of these values provide an estimate of the entropy of the user chosen password.

Once more, this is not necessarily an ideal representation as the purpose of the experiment was not focussed on passwords specifically.

These two approaches, whilst flawed, are generally the best we have. Indeed, the initial calculation tends to be the basis of strength meters on the internet.

In generating passwords guidance is beginning to change to recognise the human aspect of using passwords. In particular, recommendations such as forcing frequent password changes, adding requirements such as minimum length, plus a mix of lower and upper, plus special characters and numbers place undue burden on users and as we have discussed this can result in poor password management.

Instead, have a look at this guidance from the National Cyber Security Centre which recommends three random words: https://www.ncsc.gov.uk/blog-post/three-random-words-or-thinkrandom-0 This is more memorable, and emphasises length over randomness.

One final option for passwords which is worth mentioning is the use of a one-time password. A one-time password (OTP) is a random password valid for a single session only, ordinarily with a time limit by which you must use it. You may have come across this through use of bank security OTP devices such as those shown in Figure 1

Figure 1

Storing Passwords

When a user registers for a service, they often are required to provide a username and password. In order to authenticate the user in future, this password needs to be stored by the service for future reference. Clearly we should not store this password 'in the clear' that is in a plaintext format where if the database were compromised any attacker could simply read the passwords without further effort.

Instead we look at the process of using password hashes and salts.

Password hashes are effectively cryptographic hashes, but often have mechanisms inbuilt to slow the process of generating a hash. As you will see when we come to the types of attacks on passwords, slowing the computation of the hash makes it more challenging for an attacker. One example of such a hashing function is bcrypt which is based on the blowfish cipher.

Hashing the password for storage ensures the plaintext password is never stored on the server or in the database. It also means, so long as an appropriate hashing function is used, that it is a one way function making it difficult to determine the original password from the hash value alone. You will see as we discuss the different aspects that there are a range of mechanisms which improve the security of the password in storage.

Salts

Another way of ensuring a password is robust to cracking when stored in a database is to make use of a salt. A salt is a long pseudo-random string (generated using a cryptographically secure pseudorandom generator) which is prepended or appended to a password before it is hashed. This means that if two or more users have the same password, they will have a different password hash due to the inclusion of a salt. This again makes it more challenging for an attacker to crack a password as we will see below. To ensure a password hash is correct at the time of authentication, the server needs to store the salt in the database with the password. Note that the salt does not need to be kept secret, it is providing a randomisation of the password which results in specific attacks becoming ineffective. Passwords within a database should use different salts since having two identical passwords with the same salt would mean the same hash value, which is contrary to the purpose of the salt. A salt should also be sufficiently long so as to avoid an attacker being able to calculate all possible salts. For example, an 8 bit salt would be much too small. You may wish to use a salt of the same length as the size of your hash function output, e.g. SHA256 would use salts of length 256 bits.

Three Strikes and You Are Out

Often to mitigate an attacker attempting many passwords in an effort to gain access to the system, a system will implement a maximum number of attempts before the user is locked out of their account. The number of unsuccessful attempts can vary, but often it is three or five. The limitation here is that users forget their passwords legitimately and it can be cumbersome to reset the password.

Checking the Password for Authentication Process

Making use of hash functions and salts means when a user authenticates the following general process is followed:

- The provided password is combined with salt and hashed
- The hash value is compared with the stored hash value
- If they match, the user is successfully authenticated

Attacking Passwords

There are a range of different attacks which can be used to "crack" a password.

Brute Force

In a brute force attack the attacker attempts all possible passwords, which is generated from the password space. This can be completed online, where a tool or attacker provides direct entry in real time. Alternatively, this can be completed offline. In this situation the attacker has access to a set of credentials, e.g. from a compromised database of passwords. It is possible to buy such data sets, and older data sets of compromised passwords are readily available on the web. These are often gathered through attacking systems using attacks such as SQLi. You may even have received a communication from a company asking you to reset your password due to a suspected or known compromise. Clearly brute force could be time consuming, and is not the most effective approach.

Password Guessing

It should be noted that if an individual is known to the attacker, or there is information available to the attacker about a target individual then guessing the password can become easier. A common example is often portrayed in film and TV where in an effort to break into a laptop they try combinations of relatives' birthdays and favourite pets. All this is simply to say be careful in how you generate passwords and share personal information.

Another area for password guessing is where default username and passwords (such as admin and password123) which should be changed on purchase of a device or installation of software are left as is. This means an attacker with knowledge of the default configuration for a given device or system may be able to exploit it.

Dictionary Attack

A dictionary attack is more sophisticated than a brute force attack. It makes use of a pre-populated list of possible passwords. This can be generated from common password lists, compromised password lists, and words from a dictionary as we typically think of it. It can make use of lists of pop culture references such as movies, and need not be in English alone. Using such a list is much more likely to be successful since we are reducing the potential password space to more probable passwords. Dictionary attacks can be completed online as detailed above, or offline with a compromised database of usernames and passwords (or password hashes).

Note that if passwords are hashed and the attacker is attempting to determine the plaintext value, then the attacker must compute hash values of the plaintext dictionary values. Multiple hash functions can be used to calculate hash values for a single potential password. This allows the attacker to compare the computed hash with the hash in the compromised dataset to determine the original input. This is where salts can help mitigate attacks- by randomising the hash values an attacker needs to know the salt for that password in order to compute the hash value. Note also that assuming each user has a different salt, the same password will have different hash values due to the different salt.

Pre-computed Hash Tables

As one might imagine, an attacker can precompute different hash values using a range of common hash functions and a dictionary of passwords. This means when it comes to trying to crack passwords, they are able to use a lookup table to determine if they have the original password. If they find a hash value in the compromised passwords which matches a pre-computed hash then they can locate the password used to generate that value. As above, this assumes there is no salt, or a very poorly implemented salt.

There is a more sophisticated version of pre-computed hash tables called rainbow tables, but this is beyond the scope for our purpose.

Software

As you may have guessed, there is software available to crack passwords. One real world example is John the Ripper

Should you wish to explore the software, you are advised not to use this software on anyone else's passwords.

Last modified: Wednesday, 2 November 2022, 12:50 PM

◄ 4.5: Poll: The Password Problem

Jump to...

4.7: Video: Biometrics Overview (23-24 - Update v2 Oct2023) (09:35) ►