



**Database Management System Lab  
Project Report  
On  
“Password Generator ”**

In partial fulfillment  
For the award of degree of  
**“Bachelor of Technology”**  
**(Computer Science with AI-ML)**

**Submitted To:**  
Ms. Neny Pandel  
(Assistant Professor)

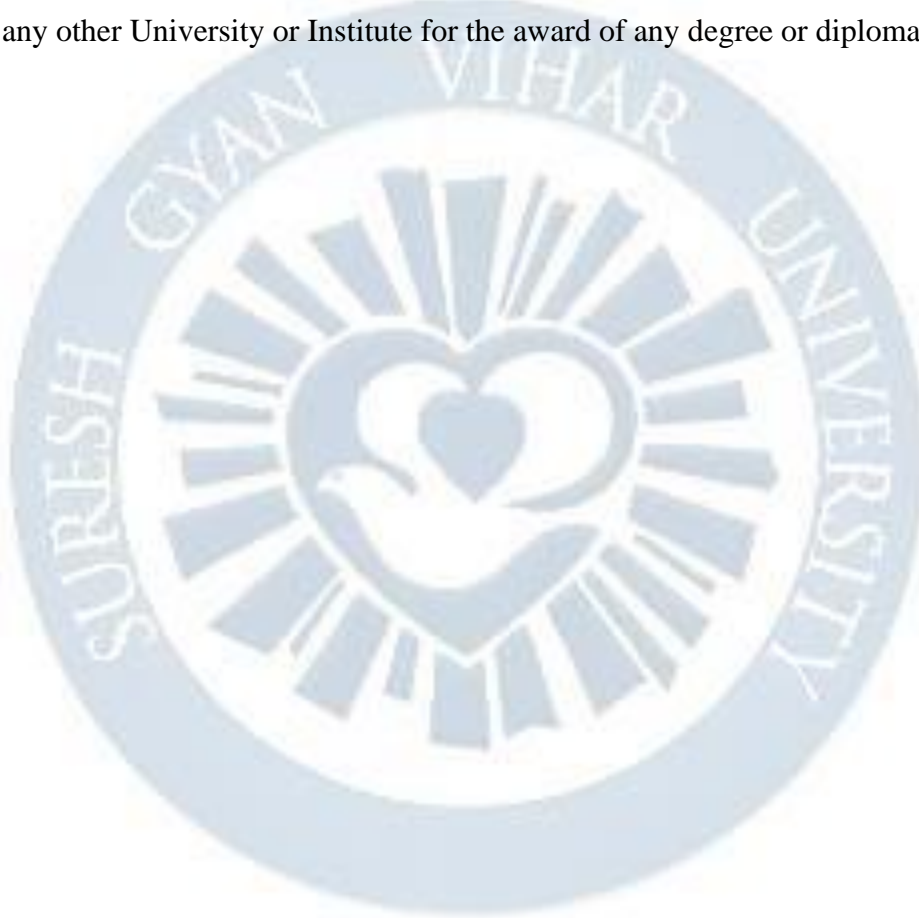
**Submitted By:**  
Anwar Alam  
(100846)

**Department of Computer Engineering & Information Technology  
SGVU Jaipur(302017)**

# DECLARATION

I hereby declare that the project work entitled “**DATABASE MANAGEMENT SYSTEM**” submitted to the SGVU Jaipur, is a record of an original work done by me under the guidance of **Ms. Neny Pandel** , Assistance Professor, Dept. of Computer Engineering And Information Technology , Gyan Vihar School of Engineering and Technology, SGVU.

This project work is submitted in the partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science & Engineering. This result embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.



Student's Sign

Submitted To:  
Ms. Neny Pandel  
(Assistant Professor)

# **CERTIFICATE**

This is to certify that the people report entitled “DATABASE MANAGEMENT SYSTEM” Submitted to Suresh Gyan Vihar University, Jaipur in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology, is an authentic and original work carried out by ANWAR ALAM (100846) under my guidance.

The matter embodied in this project is genuine work done by the student and has not been submitted whether to this university or to any university for the fulfilment of the requirement of any course of study.

**Ms. Neny Pandel**  
**Assistant Professor**

**Dr. Sohit Agarwal**  
**HOD, CEIT**

# ACKNOWLEDGEMENT

I would like to express my profound gratitude to Ms. Neny Pandel of CEIT department, and Mr. Sohit Agarwal of Suresh Gyan Vihar University for their contributions to the completion of my project titled **“DATABASE MANAGEMENT SYSTEM”**.

I would like to express my special thanks to our mentor **Ms. Neny Pandel** for her time and efforts she provided throughout the year. Her useful advice and suggestions were really helpful to me during the project's completion. In this aspect, I am eternally grateful to her.

I would like to acknowledge that this project was completed entirely by me and not by someone else.

Student Name: ANWAR ALAM  
SID No.: 100846  
Sem: 5th

## **Page Index**

<b>1.ABSTRACT.....</b>	<b>6</b>
<b>2. INTRODUCTION.....</b>	<b>7</b>
<b>3. WORKING.....</b>	<b>8</b>
<b>4.REUIREMENTS.....</b>	<b>9</b>
<b>5.BUILDING THE GENERATOR.....</b>	<b>10</b>
<b>6.INTRODUCTION TO IMAGE.....</b>	<b>11</b>
<b>7.ADVANTAGES &amp; DISADVANTAGES.....</b>	<b>12-13</b>
<b>8.DATA FLOW DIAGRAM.....</b>	<b>14</b>
<b>9.SOURCE CODE.....</b>	<b>15</b>
<b>10.OUTPUT.....</b>	<b>16</b>
<b>11.FEATURES.....</b>	<b>17</b>
<b>12.HISTORY OF PASSWORD.....</b>	<b>18-28</b>
<b>12.CONCLUSION.....</b>	<b>29</b>

# ABSTRACT

## Abstract:

With the increasing prevalence of online services and the growing concern over cybersecurity, the need for robust and unique passwords has never been greater. SecurePass is a cutting-edge password generator designed to address these concerns by generating strong, complex, and easily memorable passwords. This abstract provides an overview of SecurePass, its features, and its potential benefits.

SecurePass employs a hybrid approach that combines the security of random character generation with the memorability of passphrases. The generator takes into account user preferences, such as password length, character types (letters, numbers, symbols), and even user-defined keywords. This customization ensures that users can create passwords tailored to their security needs while maintaining ease of use.

## KEY FEATURES OF SECUREPASS:

**Customizable Parameters:** Users can specify password length, character types (uppercase letters, lowercase letters, numbers, symbols), and even include meaningful keywords to make the password more memorable.

**Secure Randomness:** SecurePass utilizes strong cryptographic algorithms to generate random characters, making the generated passwords resistant to brute force attacks.

**Passphrase Option:** Users have the choice to generate passwords as passphrases, which consist of randomly selected words or word combinations. Passphrases are easier to remember than traditional complex passwords while maintaining security.

**Strength Meter:** SecurePass provides a visual strength meter that indicates the complexity and security level of the generated password, helping users make informed choices.

**Offline Mode:** For added security, SecurePass can be used offline, reducing the risk of exposure to potential online threats.

**Cross-Platform Compatibility:** SecurePass is available as a web application, desktop software, and mobile app, ensuring accessibility across various devices and platforms.

**Password Management:** SecurePass offers an optional built-in password manager that securely stores and organizes generated passwords.

By combining the convenience of user customization with robust security measures, SecurePass aims to empower individuals and organizations to create and manage strong, unique passwords effortlessly. This advanced password generator represents a crucial step towards enhancing online security, safeguarding personal and sensitive information from potential cyber threats.

In an era where password breaches are becoming increasingly common, SecurePass offers a reliable solution for individuals and businesses seeking to fortify their digital defenses. Its innovative approach to password generation strikes a balance between security and usability, making it a valuable tool in the ongoing battle against cyberattacks.

## INTRODUCTION

In today's digital age, password security is paramount. With the increasing prevalence of online accounts, protecting your sensitive information is crucial. A password generator is a valuable tool that helps individuals and organizations create strong and secure passwords.

Password generators are software or online tools designed to generate complex and random passwords that are difficult for hackers to guess or crack. These passwords typically include a mix of uppercase and lowercase letters, numbers, and special characters. The primary purpose of a password generator is to enhance security by creating passwords that are virtually impossible for attackers to predict through common techniques like dictionary attacks or brute force attacks.

### HERE ARE SOME KEY ASPECTS OF PASSWORD GENERATORS:

**Security:** Password generators prioritize security by creating passwords that are highly resistant to hacking attempts. These passwords are often long and incorporate various character types, making them challenging for malicious actors to decipher.

**Complexity:** Password generators can produce passwords with a high degree of complexity, ensuring that they meet stringent security requirements. Complex passwords are essential for protecting sensitive data and accounts.

**Customization:** Many password generators allow users to customize the generated passwords according to specific requirements. Users can specify the length of the password, whether it includes special characters, and other criteria.

**Ease of Use:** Password generators are typically user-friendly and accessible to individuals with varying levels of technical expertise. They simplify the process of creating strong passwords.

**Storage:** Password generators often come with built-in features for securely storing generated passwords. Some offer password management functions, helping users keep track of multiple passwords across different accounts.

**Offline and Online:** Password generators can be either offline software or online web-based tools. Users can choose the one that best fits their needs and preferences.

**Password Strength Assessment:** Some password generators provide a strength assessment for the generated passwords, helping users gauge the security level of the password before using it.

In summary, password generators are indispensable tools for maintaining online security. They play a vital role in safeguarding personal and sensitive information by creating strong, complex, and unique passwords. As cyber threats continue to evolve, using a password generator is a proactive step in ensuring the safety of digital assets and accounts.

## WORKING

Creating a password generator can be a useful and fun coding project. Here's a simple Python password generator as an example:

```
python
import random
import string

def generate_password(length=12, use_digits=True, use_special_chars=True):
    # Define character sets
    characters = string.ascii_letters
    if use_digits:
        characters += string.digits
    if use_special_chars:
        characters += string.punctuation

    # Generate the password
    password = ''.join(random.choice(characters) for _ in range(length))

    return password

# Example usage:
password = generate_password()
print(password)
In this example:
```

We import the random and string modules to help generate the password.

The generate\_password function takes three parameters: length (default 12), use\_digits (default True), and use\_special\_chars (default True).

We define the character set based on the options passed to the function. By default, it includes uppercase and lowercase letters, digits, and special characters.

The function uses a list comprehension to generate a password by randomly selecting characters from the character set.

Finally, it returns the generated password.

You can call the generate\_password function to create passwords with different settings. For example:

```
python
password = generate_password(length=16, use_digits=False, use_special_chars=False)
print(password)
```

This will generate a 16-character password without digits or special characters.



## REQUIREMENT

Certainly, I can help you create a password generator. A password generator is a tool that generates random and strong passwords according to certain criteria, such as length, character types, and complexity. Here's a simple Python script for a password generator:

```
python
import random
import string

def generate_password(length=12, use_digits=True, use_special_chars=True):
    characters = string.ascii_letters
    if use_digits:
        characters += string.digits
    if use_special_chars:
        characters += string.punctuation

    password = ''.join(random.choice(characters) for _ in range(length))
    return password

if __name__ == "__main__":
    length = int(input("Enter the desired password length: "))
    use_digits = input("Include digits? (yes/no): ").lower() == "yes"
    use_special_chars = input("Include special characters? (yes/no): ").lower() == "yes"

    password = generate_password(length, use_digits, use_special_chars)
    print("Generated Password: ", password)
```

This script allows you to specify the desired length of the password and whether you want to include digits and special characters. It then generates a random password based on your input.

### To run this script:

Make sure you have Python installed on your system.

Copy the code into a Python file (e.g., password\_generator.py).

Run the script, and it will prompt you for the desired password length and character types.

You can customize this script further based on your specific requirements, such as adding more character types or constraints. Additionally, there are many existing libraries and online tools for password generation if you need more advanced features.

## BUILDING THE GENERATOR

Creating a password generator is a useful and relatively simple programming project. You can implement it in various programming languages, but I'll provide you with a Python example using a basic randomization approach. This example will generate a strong random password based on your specifications.

```
python
import random
import string

def generate_password(length=12, use_digits=True, use_special_chars=True):
    # Define character sets based on user preferences
    lowercase_chars = string.ascii_lowercase
    uppercase_chars = string.ascii_uppercase
    digit_chars = string.digits if use_digits else ""
    special_chars = "!@#%&*()_+~[]{}|;:'.<>?/" if use_special_chars else ""

    # Combine character sets based on preferences
    all_chars = lowercase_chars + uppercase_chars + digit_chars + special_chars

    if not all_chars:
        raise ValueError("You must select at least one character set for the password.")

    # Ensure the password length is valid
    if length < 1:
        raise ValueError("Password length must be at least 1.")

    # Generate the password
    password = "".join(random.choice(all_chars) for _ in range(length))

    return password

# Example usage:
password = generate_password(length=16, use_digits=True, use_special_chars=True)
print(password)
```

This code defines a `generate_password` function that takes three parameters:

**length:** The length of the password you want to generate (default is 12).

**use\_digits:** Whether to include digits in the password (default is True).

**use\_special\_chars:** Whether to include special characters in the password (default is True).

You can customize the password length and character set by calling the `generate_password` function with your desired parameters.

Remember that strong passwords are typically longer and include a mix of uppercase letters, lowercase letters, digits, and special characters. Adjust the parameters accordingly to match your security requirements.

Here's an example of generating a 16-character password with digits and special characters included: `bash`

## INTRODUCTION TO IMAGE



## ADVANTAGES & DISADVANTAGES

Password generators are tools or programs that create strong and random passwords for various online accounts and security purposes. They offer several advantages and disadvantages:

### ADVANTAGES:

**Enhanced Security:** Password generators create complex, unique passwords that are difficult for hackers to guess or crack, enhancing the security of your online accounts.

**Randomness:** They generate passwords with a high degree of randomness, reducing the risk of patterns or easily guessable combinations.

**Convenience:** Password generators save time and effort by automatically creating strong passwords, eliminating the need for users to come up with their own.

**Complexity:** They can include a mix of uppercase and lowercase letters, numbers, and special characters, making it more challenging for attackers to decipher the password.

**Protection Against Dictionary Attacks:** Password generators often avoid common words and phrases, reducing vulnerability to dictionary attacks.

**Customization:** Many password generators allow you to customize password length and character composition to meet specific security requirements.

**Multi-Platform Use:** Password generators can be used across various platforms, including websites, mobile apps, and password management tools.

### DISADVANTAGES:

**Dependency:** Relying solely on a password generator can lead to a situation where users don't remember their passwords, requiring them to store passwords digitally or use a password manager, which can introduce its own security risks.

**Limited Memorability:** Generated passwords are often complex and challenging to memorize, leading users to write them down or store them insecurely.

**Compatibility Issues:** Some websites or systems may have specific password requirements that password generators do not always adhere to, causing usability issues.

**Loss of Control:** Users may feel they have less control over their passwords' security when relying on generators, as the process is automated.

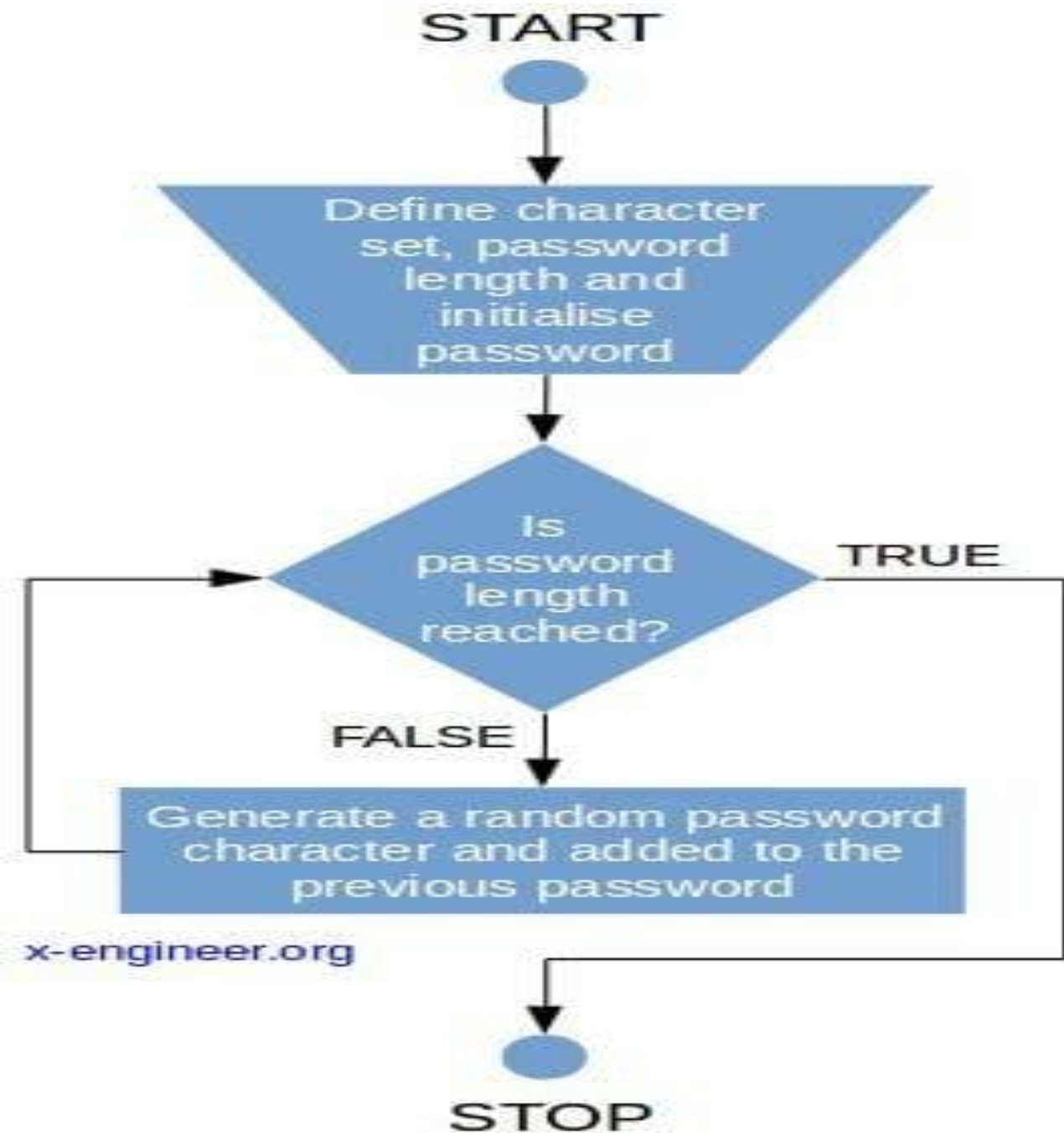
**Initial Setup:** Setting up a password generator or integrating it into existing workflows may require some effort and technical knowledge.

**Risk of Exposure:** If you use a password generator in an environment or application that isn't secure, the generated password could be intercepted or compromised during transmission.

**Forgotten Passwords:** Generated passwords can be difficult to remember, and if you lose access to the generator or the stored passwords, it may be challenging to regain access to your accounts.

In summary, password generators are valuable tools for enhancing online security, but they should be used in conjunction with secure storage methods (like password managers) and proper security practices to mitigate their disadvantages and ensure overall account security.

## DATA FLOW DIAGRAM



## SOURCE CODE

```
'''
Password Generator
-----
'''

import secrets
import string

def create_pw(pw_length=12):
    letters = string.ascii_letters
    digits = string.digits
    special_chars = string.punctuation

    alphabet = letters + digits + special_chars
    pwd = ""
    pw_strong = False

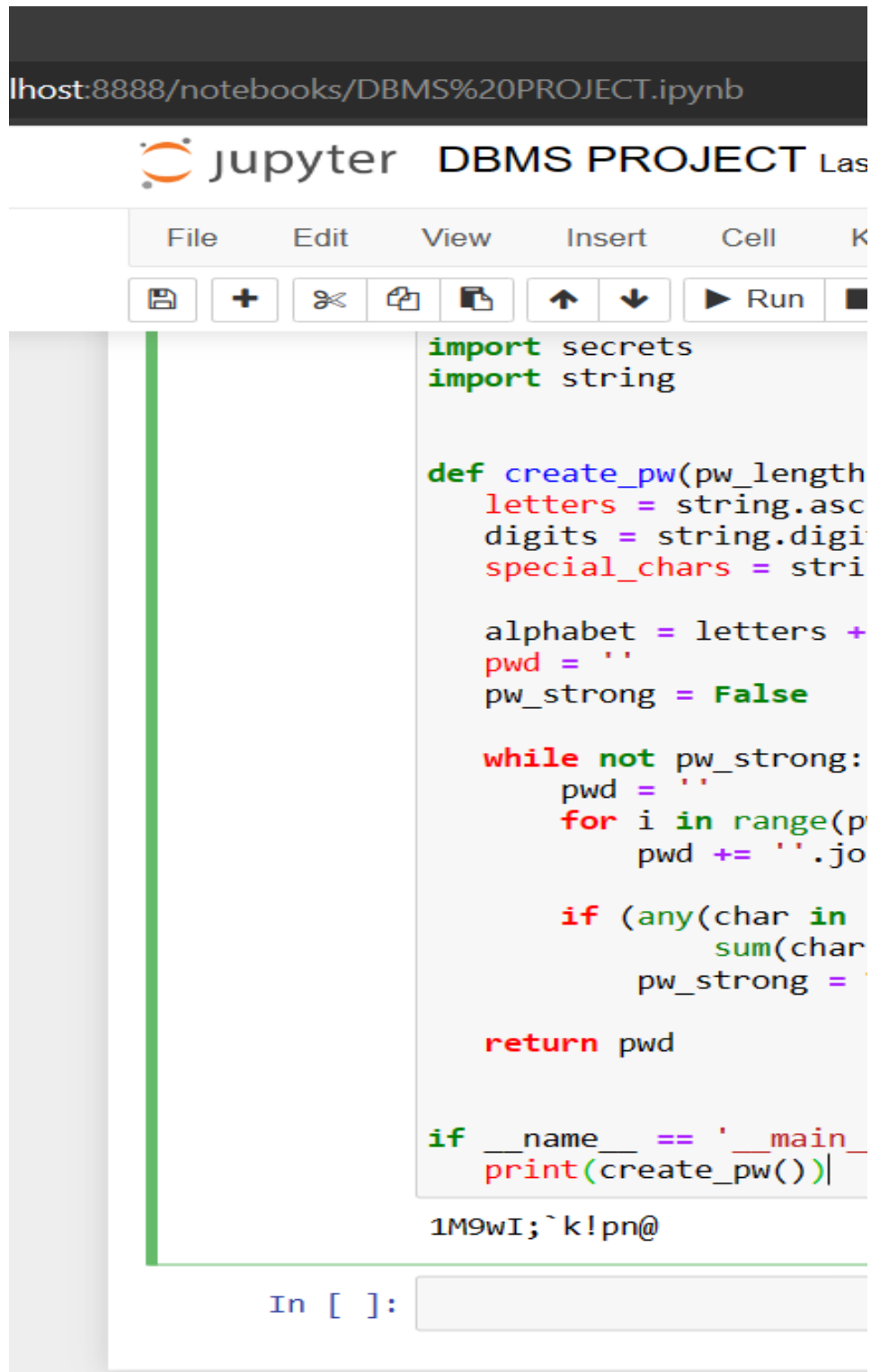
    while not pw_strong:
        pwd = ""
        for i in range(pw_length):
            pwd += "".join(secrets.choice(alphabet))

        if (any(char in special_chars for char in pwd) and
            sum(char in digits for char in pwd) >= 2):
            pw_strong = True

    return pwd

if __name__ == '__main__':
    print(create_pw())
```

# OUTPUT



The screenshot shows a Jupyter Notebook window titled "jupyter DBMS PROJECT". The address bar at the top displays "localhost:8888/notebooks/DBMS%20PROJECT.ipynb". The notebook interface includes a menu bar with "File", "Edit", "View", "Insert", "Cell", and "Kernel" options. Below the menu is a toolbar with icons for saving, adding new cells, cutting, copying, pasting, and running code. The main area of the notebook contains a Python script for generating a strong password. The script imports the 'secrets' and 'string' modules, defines a 'create\_pw' function that builds a character set from letters, digits, and special characters, and uses a loop to generate a password until it meets strength criteria. The script concludes with a main block that prints the generated password. The output of the script, "1M9wI;`k!pn@", is displayed below the code cell.

```
import secrets
import string

def create_pw(pw_length):
    letters = string.ascii_letters
    digits = string.digits
    special_chars = string.punctuation

    alphabet = letters + digits + special_chars
    pwd = ''
    pw_strong = False

    while not pw_strong:
        pwd = ''
        for i in range(pw_length):
            pwd += secrets.choice(alphabet)

        if (any(char in special_chars for char in pwd) and
            sum(char in digits for char in pwd) > 2):
            pw_strong = True

    return pwd

if __name__ == '__main__':
    print(create_pw(10))
```

In [ ]:

1M9wI;`k!pn@



## FEATURES

Creating a password generator with Python is a straightforward task. Here's a simple example of a password generator with some customizable features:

```
python
import random
import string

def generate_password(length=12, include_lowercase=True, include_uppercase=True,
include_digits=True, include_special_chars=True):
    characters = ""

    if include_lowercase:
        characters += string.ascii_lowercase
    if include_uppercase:
        characters += string.ascii_uppercase
    if include_digits:
        characters += string.digits
    if include_special_chars:
        characters += string.punctuation

    if not characters:
        return "Invalid configuration. No character set selected."

    password = ''.join(random.choice(characters) for _ in range(length))
    return password

# Example usage:
password = generate_password(length=16, include_lowercase=True, include_uppercase=True,
include_digits=True, include_special_chars=True)
print(password)
```

In this code, you can customize the password generation by specifying the desired length and whether to include lowercase letters, uppercase letters, digits, and special characters. The `generate_password` function constructs the character set based on your choices and then generates a random password of the specified length.

You can adjust the function's parameters to fit your requirements, such as changing the default length or excluding certain character sets if needed.

# HISTORY OF PASSWORD

A password, sometimes called a passcode (for example in Apple devices),[1] is secret data, typically a string of characters, usually used to confirm a user's identity.[1] Traditionally, passwords were expected to be memorized,[2] but the large number of password-protected services that a typical individual accesses can make memorization of unique passwords for each service impractical.[3] Using the terminology of the NIST Digital Identity Guidelines,[4] the secret is held by a party called the claimant while the party verifying the identity of the claimant is called the verifier. When the claimant successfully demonstrates knowledge of the password to the verifier through an established authentication protocol,[5] the verifier is able to infer the claimant's identity.

In general, a password is an arbitrary string of characters including letters, digits, or other symbols. If the permissible characters are constrained to be numeric, the corresponding secret is sometimes called a personal identification number (PIN).

Despite its name, a password does not need to be an actual word; indeed, a non-word (in the dictionary sense) may be harder to guess, which is a desirable property of passwords. A memorized secret consisting of a sequence of words or other text separated by spaces is sometimes called a passphrase. A passphrase is similar to a password in usage, but the former is generally longer for added security.[6]

## HISTORY

Passwords have been used since ancient times. Sentries would challenge those wishing to enter an area to supply a password or watchword, and would only allow a person or group to pass if they knew the password. Polybius describes the system for the distribution of watchwords in the Roman military as follows:

The way in which they secure the passing round of the watchword for the night is as follows: from the tenth maniple of each class of infantry and cavalry, the maniple which is encamped at the lower end of the street, a man is chosen who is relieved from guard duty, and he attends every day at sunset at the tent of the tribune, and receiving from him the watchword—that is a wooden tablet with the word inscribed on it – takes his leave, and on returning to his quarters passes on the watchword and tablet before witnesses to the commander of the next maniple, who in turn passes it to the one next to him. All do the same until it reaches the first maniples, those encamped near the tents of the tribunes. These latter are obliged to deliver the tablet to the tribunes before dark. So that if all those issued are returned, the tribune knows that the watchword has been given to all the maniples, and has passed through all on its way back to him. If any one of them is missing, he makes inquiry at once, as he knows by the marks from what quarter the tablet has not returned, and whoever is responsible for the stoppage meets with the punishment he merits.[7]

Passwords in military use evolved to include not just a password, but a password and a counterpassword; for example in the opening days of the Battle of Normandy, paratroopers of the U.S. 101st Airborne Division used a password—flash—which was presented as a challenge, and answered with the correct response—thunder. The challenge and response were changed every three days. American paratroopers also famously used a device known as a "cricket" on D-Day in place of a password system as a temporarily unique method of identification; one metallic click given by the device in lieu of a password was to be met by two clicks in reply.[8]

Passwords have been used with computers since the earliest days of computing. The Compatible Time-Sharing System (CTSS), an operating system introduced at MIT in 1961, was the first computer system to implement password login.[9][10] CTSS had a LOGIN command that requested a user password. "After typing PASSWORD, the system turns off the printing mechanism, if possible, so that the user may type in his password with privacy." [11] In the early 1970s, Robert Morris developed a system of storing login passwords

in a hashed form as part of the Unix operating system. The system was based on a simulated Hagelin rotor crypto machine, and first appeared in 6th Edition Unix in 1974. A later version of his algorithm, known as crypt(3), used a 12-bit salt and invoked a modified form of the DES algorithm 25 times to reduce the risk of pre-computed dictionary attacks.[12]

In modern times, user names and passwords are commonly used by people during a log in process that controls access to protected computer operating systems, mobile phones, cable TV decoders, automated teller machines (ATMs), etc. A typical computer user has passwords for many purposes: logging into accounts, retrieving e-mail, accessing applications, databases, networks, web sites, and even reading the morning newspaper online.

#### Choosing a secure and memorable password

The easier a password is for the owner to remember generally means it will be easier for an attacker to guess.[13] However, passwords that are difficult to remember may also reduce the security of a system because (a) users might need to write down or electronically store the password, (b) users will need frequent password resets and (c) users are more likely to re-use the same password across different accounts. Similarly, the more stringent the password requirements, such as "have a mix of uppercase and lowercase letters and digits" or "change it monthly", the greater the degree to which users will subvert the system.[14] Others argue longer passwords provide more security (e.g., entropy) than shorter passwords with a wide variety of characters.[15]

In The Memorability and Security of Passwords,[16] Jeff Yan et al. examine the effect of advice given to users about a good choice of password. They found that passwords based on thinking of a phrase and taking the first letter of each word are just as memorable as naively selected passwords, and just as hard to crack as randomly generated passwords.

Combining two or more unrelated words and altering some of the letters to special characters or numbers is another good method,[17] but a single dictionary word is not. Having a personally designed algorithm for generating obscure passwords is another good method.[18]

However, asking users to remember a password consisting of a "mix of uppercase and lowercase characters" is similar to asking them to remember a sequence of bits: hard to remember, and only a little bit harder to crack (e.g. only 128 times harder to crack for 7-letter passwords, less if the user simply capitalises one of the letters). Asking users to use "both letters and digits" will often lead to easy-to-guess substitutions such as 'E' → '3' and 'I' → '1', substitutions that are well known to attackers. Similarly typing the password one keyboard row higher is a common trick known to attackers.[19]

In 2013, Google released a list of the most common password types, all of which are considered insecure because they are too easy to guess (especially after researching an individual on social media):[20]

The name of a pet, child, family member, or significant other

Anniversary dates and birthdays

Birthplace

Name of a favorite holiday

Something related to a favorite sports team

The word "password"

Alternatives to memorization

Traditional advice to memorize passwords and never write them down has become a challenge because of the sheer number of passwords users of computers and the internet are expected to maintain. One survey concluded that the average user has around 100 passwords.[3] To manage the proliferation of passwords, some

users employ the same password for multiple accounts, a dangerous practice since a data breach in one account could compromise the rest. Less risky alternatives include the use of password managers, single sign-on systems and simply keeping paper lists of less critical passwords.[21] Such practices can reduce the number of passwords that must be memorized, such as the password manager's master password, to a more manageable number.

#### Factors in the security of a password system

The security of a password-protected system depends on several factors. The overall system must be designed for sound security, with protection against computer viruses, man-in-the-middle attacks and the like. Physical security issues are also a concern, from deterring shoulder surfing to more sophisticated physical threats such as video cameras and keyboard sniffers. Passwords should be chosen so that they are hard for an attacker to guess and hard for an attacker to discover using any of the available automatic attack schemes. See password strength and computer security for more information.[22]

Nowadays, it is a common practice for computer systems to hide passwords as they are typed. The purpose of this measure is to prevent bystanders from reading the password; however, some argue that this practice may lead to mistakes and stress, encouraging users to choose weak passwords. As an alternative, users should have the option to show or hide passwords as they type them.[22]

Effective access control provisions may force extreme measures on criminals seeking to acquire a password or biometric token.[23] Less extreme measures include extortion, rubber hose cryptanalysis, and side channel attack.

Some specific password management issues that must be considered when thinking about, choosing, and handling, a password follow.

#### Rate at which an attacker can try guessed passwords

The rate at which an attacker can submit guessed passwords to the system is a key factor in determining system security. Some systems impose a time-out of several seconds after a small number (e.g., three) of failed password entry attempts, also known as throttling.[4] :63B Sec 5.2.2 In the absence of other vulnerabilities, such systems can be effectively secure with relatively simple passwords if they have been well chosen and are not easily guessed.[24]

Many systems store a cryptographic hash of the password. If an attacker gets access to the file of hashed passwords guessing can be done offline, rapidly testing candidate passwords against the true password's hash value. In the example of a web-server, an online attacker can guess only at the rate at which the server will respond, while an off-line attacker (who gains access to the file) can guess at a rate limited only by the hardware on which the attack is running.

Passwords that are used to generate cryptographic keys (e.g., for disk encryption or Wi-Fi security) can also be subjected to high rate guessing. Lists of common passwords are widely available and can make password attacks very efficient. (See Password cracking.) Security in such situations depends on using passwords or passphrases of adequate complexity, making such an attack computationally infeasible for the attacker. Some systems, such as PGP and Wi-Fi WPA, apply a computation-intensive hash to the password to slow such attacks. See key stretching.

#### Limits on the number of password guesses

An alternative to limiting the rate at which an attacker can make guesses on a password is to limit the total number of guesses that can be made. The password can be disabled, requiring a reset, after a small number of

consecutive bad guesses (say 5); and the user may be required to change the password after a larger cumulative number of bad guesses (say 30), to prevent an attacker from making an arbitrarily large number of bad guesses by interspersing them between good guesses made by the legitimate password owner.[25] Attackers may conversely use knowledge of this mitigation to implement a denial of service attack against the user by intentionally locking the user out of their own device; this denial of service may open other avenues for the attacker to manipulate the situation to their advantage via social engineering.

#### Form of stored passwords

Some computer systems store user passwords as plaintext, against which to compare user logon attempts. If an attacker gains access to such an internal password store, all passwords—and so all user accounts—will be compromised. If some users employ the same password for accounts on different systems, those will be compromised as well.

More secure systems store each password in a cryptographically protected form, so access to the actual password will still be difficult for a snooper who gains internal access to the system, while validation of user access attempts remains possible. The most secure do not store passwords at all, but a one-way derivation, such as a polynomial, modulus, or an advanced hash function.[15] Roger Needham invented the now-common approach of storing only a "hashed" form of the plaintext password.[26][27] When a user types in a password on such a system, the password handling software runs through a cryptographic hash algorithm, and if the hash value generated from the user's entry matches the hash stored in the password database, the user is permitted access. The hash value is created by applying a cryptographic hash function to a string consisting of the submitted password and, in many implementations, another value known as a salt. A salt prevents attackers from easily building a list of hash values for common passwords and prevents password cracking efforts from scaling across all users.[28] MD5 and SHA1 are frequently used cryptographic hash functions, but they are not recommended for password hashing unless they are used as part of a larger construction such as in PBKDF2.[29]

The stored data—sometimes called the "password verifier" or the "password hash"—is often stored in Modular Crypt Format or RFC 2307 hash format, sometimes in the `/etc/passwd` file or the `/etc/shadow` file.[30]

The main storage methods for passwords are plain text, hashed, hashed and salted, and reversibly encrypted.[31] If an attacker gains access to the password file, then if it is stored as plain text, no cracking is necessary. If it is hashed but not salted then it is vulnerable to rainbow table attacks (which are more efficient than cracking). If it is reversibly encrypted then if the attacker gets the decryption key along with the file no cracking is necessary, while if he fails to get the key cracking is not possible. Thus, of the common storage formats for passwords only when passwords have been salted and hashed is cracking both necessary and possible.[31]

If a cryptographic hash function is well designed, it is computationally infeasible to reverse the function to recover a plaintext password. An attacker can, however, use widely available tools to attempt to guess the passwords. These tools work by hashing possible passwords and comparing the result of each guess to the actual password hashes. If the attacker finds a match, they know that their guess is the actual password for the associated user. Password cracking tools can operate by brute force (i.e. trying every possible combination of characters) or by hashing every word from a list; large lists of possible passwords in many languages are widely available on the Internet.[15] The existence of password cracking tools allows attackers to easily recover poorly chosen passwords. In particular, attackers can quickly recover passwords that are short, dictionary words, simple variations on dictionary words, or that use easily guessable patterns.[32] A modified version of the DES algorithm was used as the basis for the password hashing algorithm in early Unix systems.[33] The crypt algorithm used a 12-bit salt value so that each user's hash was unique and iterated the

DES algorithm 25 times in order to make the hash function slower, both measures intended to frustrate automated guessing attacks.[33] The user's password was used as a key to encrypt a fixed value. More recent Unix or Unix-like systems (e.g., Linux or the various BSD systems) use more secure password hashing algorithms such as PBKDF2, bcrypt, and scrypt, which have large salts and an adjustable cost or number of iterations.[34] A poorly designed hash function can make attacks feasible even if a strong password is chosen. See LM hash for a widely deployed and insecure example.[35]

#### Methods of verifying a password over a network

##### Simple transmission of the password

Passwords are vulnerable to interception (i.e., "snooping") while being transmitted to the authenticating machine or person. If the password is carried as electrical signals on unsecured physical wiring between the user access point and the central system controlling the password database, it is subject to snooping by wiretapping methods. If it is carried as packeted data over the Internet, anyone able to watch the packets containing the logon information can snoop with a very low probability of detection.

Email is sometimes used to distribute passwords but this is generally an insecure method. Since most email is sent as plaintext, a message containing a password is readable without effort during transport by any eavesdropper. Further, the message will be stored as plaintext on at least two computers: the sender's and the recipient's. If it passes through intermediate systems during its travels, it will probably be stored on there as well, at least for some time, and may be copied to backup, cache or history files on any of these systems.

Using client-side encryption will only protect transmission from the mail handling system server to the client machine. Previous or subsequent relays of the email will not be protected and the email will probably be stored on multiple computers, certainly on the originating and receiving computers, most often in clear text.

##### Transmission through encrypted channels

The risk of interception of passwords sent over the Internet can be reduced by, among other approaches, using cryptographic protection. The most widely used is the Transport Layer Security (TLS, previously called SSL) feature built into most current Internet browsers. Most browsers alert the user of a TLS/SSL-protected exchange with a server by displaying a closed lock icon, or some other sign, when TLS is in use. There are several other techniques in use; see cryptography.

##### Hash-based challenge–response methods

Unfortunately, there is a conflict between stored hashed-passwords and hash-based challenge–response authentication; the latter requires a client to prove to a server that they know what the shared secret (i.e., password) is, and to do this, the server must be able to obtain the shared secret from its stored form. On many systems (including Unix-type systems) doing remote authentication, the shared secret usually becomes the hashed form and has the serious limitation of exposing passwords to offline guessing attacks. In addition, when the hash is used as a shared secret, an attacker does not need the original password to authenticate remotely; they only need the hash.

##### Zero-knowledge password proofs

Rather than transmitting a password, or transmitting the hash of the password, password-authenticated key agreement systems can perform a zero-knowledge password proof, which proves knowledge of the password without exposing it.

Moving a step further, augmented systems for password-authenticated key agreement (e.g., AMP, B-SPEKE, PAK-Z, SRP-6) avoid both the conflict and limitation of hash-based methods. An augmented system allows a client to prove knowledge of the password to a server, where the server knows only a (not exactly) hashed

password, and where the un-hashed password is required to gain access.

#### Procedures for changing passwords

Usually, a system must provide a way to change a password, either because a user believes the current password has been (or might have been) compromised, or as a precautionary measure. If a new password is passed to the system in unencrypted form, security can be lost (e.g., via wiretapping) before the new password can even be installed in the password database and if the new password is given to a compromised employee, little is gained. Some websites include the user-selected password in an unencrypted confirmation e-mail message, with the obvious increased vulnerability.

Identity management systems are increasingly used to automate the issuance of replacements for lost passwords, a feature called self-service password reset. The user's identity is verified by asking questions and comparing the answers to ones previously stored (i.e., when the account was opened).

Some password reset questions ask for personal information that could be found on social media, such as mother's maiden name. As a result, some security experts recommend either making up one's own questions or giving false answers.[36]

#### Password longevity

"Password aging" is a feature of some operating systems which forces users to change passwords frequently (e.g., quarterly, monthly or even more often). Such policies usually provoke user protest and foot-dragging at best and hostility at worst. There is often an increase in the number of people who note down the password and leave it where it can easily be found, as well as help desk calls to reset a forgotten password. Users may use simpler passwords or develop variation patterns on a consistent theme to keep their passwords memorable.[37] Because of these issues, there is some debate as to whether password aging is effective.[38] Changing a password will not prevent abuse in most cases, since the abuse would often be immediately noticeable. However, if someone may have had access to the password through some means, such as sharing a computer or breaching a different site, changing the password limits the window for abuse.[39]

#### Number of users per password

Allotting separate passwords to each user of a system is preferable to having a single password shared by legitimate users of the system, certainly from a security viewpoint. This is partly because users are more willing to tell another person (who may not be authorized) a shared password than one exclusively for their use. Single passwords are also much less convenient to change because many people need to be told at the same time, and they make removal of a particular user's access more difficult, as for instance on graduation or resignation. Separate logins are also often used for accountability, for example to know who changed a piece of data.

#### Password security architecture

Common techniques used to improve the security of computer systems protected by a password include:

Not displaying the password on the display screen as it is being entered or obscuring it as it is typed by using asterisks (\*) or bullets (•).

Allowing passwords of adequate length. (Some legacy operating systems, including early versions[which?] of Unix and Windows, limited passwords to an 8 character maximum,[40][41][42] reducing security.)

Requiring users to re-enter their password after a period of inactivity (a semi log-off policy).

Enforcing a password policy to increase password strength and security.

Assigning randomly chosen passwords.

Requiring minimum password lengths.[29]

Some systems require characters from various character classes in a password—for example, "must have at least one uppercase and at least one lowercase letter". However, all-lowercase passwords are more secure per keystroke than mixed capitalization passwords.[43]

Employ a password blacklist to block the use of weak, easily guessed passwords

Providing an alternative to keyboard entry (e.g., spoken passwords, or biometric identifiers).

Requiring more than one authentication system, such as two-factor authentication (something a user has and something the user knows).

Using encrypted tunnels or password-authenticated key agreement to prevent access to transmitted passwords via network attacks

Limiting the number of allowed failures within a given time period (to prevent repeated password guessing).

After the limit is reached, further attempts will fail (including correct password attempts) until the beginning of the next time period. However, this is vulnerable to a form of denial of service attack.

Introducing a delay between password submission attempts to slow down automated password guessing programs.

Some of the more stringent policy enforcement measures can pose a risk of alienating users, possibly decreasing security as a result.

### Password reuse

It is common practice amongst computer users to reuse the same password on multiple sites. This presents a substantial security risk, because an attacker needs to only compromise a single site in order to gain access to other sites the victim uses. This problem is exacerbated by also reusing usernames, and by websites requiring email logins, as it makes it easier for an attacker to track a single user across multiple sites. Password reuse can be avoided or minimized by using mnemonic techniques, writing passwords down on paper, or using a password manager.[44]

It has been argued by Redmond researchers Dinei Florencio and Cormac Herley, together with Paul C. van Oorschot of Carleton University, Canada, that password reuse is inevitable, and that users should reuse passwords for low-security websites (which contain little personal data and no financial information, for example) and instead focus their efforts on remembering long, complex passwords for a few important accounts, such as bank accounts.[45] Similar arguments were made by Forbes in not change passwords as often as many "experts" advise, due to the same limitations in human memory.[37]

### Writing down passwords on paper

Historically, many security experts asked people to memorize their passwords: "Never write down a password". More recently, many security experts such as Bruce Schneier recommend that people use passwords that are too complicated to memorize, write them down on paper, and keep them in a wallet.[46][47][48][49][50][51][52]

Password manager software can also store passwords relatively safely, in an encrypted file sealed with a single master password.

### After death

According to a survey by the University of London, one in ten people are now leaving their passwords in their wills to pass on this important information when they die. One-third of people, according to the poll, agree that their password-protected data is important enough to pass on in their will.[53]

### Multi-factor authentication

Main article: Multi-factor authentication

Multi-factor authentication schemes combine passwords (as "knowledge factors") with one or more other



means of authentication, to make authentication more secure and less vulnerable to compromised passwords. For example, a simple two-factor login might send a text message, e-mail, automated phone call, or similar alert whenever a login attempt is made, possibly supplying a code that must be entered in addition to a password.[54] More sophisticated factors include such things as hardware tokens and biometric security.

### **Password rotation**

Password rotation is a policy that is commonly implemented with the goal of enhancing computer security. In 2019, Microsoft stated that the practice is "ancient and obsolete".[55][56]

### **Password rules**

Further information: Password policy

Most organizations specify a password policy that sets requirements for the composition and usage of passwords, typically dictating minimum length, required categories (e.g., upper and lower case, numbers, and special characters), prohibited elements (e.g., use of one's own name, date of birth, address, telephone number). Some governments have national authentication frameworks[57] that define requirements for user authentication to government services, including requirements for passwords.

Many websites enforce standard rules such as minimum and maximum length, but also frequently include composition rules such as featuring at least one capital letter and at least one number/symbol. These latter, more specific rules were largely based on a 2003 report by the National Institute of Standards and Technology (NIST), authored by Bill Burr.[58] It originally proposed the practice of using numbers, obscure characters and capital letters and updating regularly. In a 2017 article in The Wall Street Journal, Burr reported he regrets these proposals and made a mistake when he recommended them.[59]

According to a 2017 rewrite of this NIST report, many websites have rules that actually have the opposite effect on the security of their users. This includes complex composition rules as well as forced password changes after certain periods of time. While these rules have long been widespread, they have also long been seen as annoying and ineffective by both users and cyber-security experts.[60] The NIST recommends people use longer phrases as passwords (and advises websites to raise the maximum password length) instead of hard-to-remember passwords with "illusory complexity" such as "pA55w+rd".[61] A user prevented from using the password "password" may simply choose "Password1" if required to include a number and uppercase letter. Combined with forced periodic password changes, this can lead to passwords that are difficult to remember but easy to crack.[58]

Paul Grassi, one of the 2017 NIST report's authors, further elaborated: "Everyone knows that an exclamation point is a 1, or an I, or the last character of a password. \$ is an S or a 5. If we use these well-known tricks, we aren't fooling any adversary. We are simply fooling the database that stores passwords into thinking the user did something good."[60]

Pieris Tsokkis and Eliana Stavrou were able to identify some bad password construction strategies through their research and development of a password generator tool. They came up with eight categories of password construction strategies based on exposed password lists, password cracking tools, and online reports citing the most used passwords. These categories include user-related information, keyboard combinations and patterns, placement strategy, word processing, substitution, capitalization, append dates, and a combination of the previous categories[62]

### **Password cracking**

Main article: Password cracking

Attempting to crack passwords by trying as many possibilities as time and money permit is a brute force

attack. A related method, rather more efficient in most cases, is a dictionary attack. In a dictionary attack, all words in one or more dictionaries are tested. Lists of common passwords are also typically tested.

Password strength is the likelihood that a password cannot be guessed or discovered, and varies with the attack algorithm used. Cryptologists and computer scientists often refer to the strength or 'hardness' in terms of entropy.[15]

Passwords easily discovered are termed weak or vulnerable; passwords very difficult or impossible to discover are considered strong. There are several programs available for password attack (or even auditing and recovery by systems personnel) such as L0phtCrack, John the Ripper, and Cain; some of which use password design vulnerabilities (as found in the Microsoft LANManager system) to increase efficiency. These programs are sometimes used by system administrators to detect weak passwords proposed by users.

Studies of production computer systems have consistently shown that a large fraction of all user-chosen passwords are readily guessed automatically. For example, Columbia University found 22% of user passwords could be recovered with little effort.[63] According to Bruce Schneier, examining data from a 2006 phishing attack, 55% of MySpace passwords would be crackable in 8 hours using a commercially available Password Recovery Toolkit capable of testing 200,000 passwords per second in 2006.[64] He also reported that the single most common password was password1, confirming yet again the general lack of informed care in choosing passwords among users. (He nevertheless maintained, based on these data, that the general quality of passwords has improved over the years—for example, average length was up to eight characters from under seven in previous surveys, and less than 4% were dictionary words.[65])

## **Incidents**

On July 16, 1998, CERT reported an incident where an attacker had found 186,126 encrypted passwords. At the time the attacker was discovered, 47,642 passwords had already been cracked.[66]

In September 2001, after the deaths of 658 of their 960 New York employees in the September 11 attacks, financial services firm Cantor Fitzgerald through Microsoft broke the passwords of deceased employees to gain access to files needed for servicing client accounts.[67] Technicians used brute-force attacks, and interviewers contacted families to gather personalized information that might reduce the search time for weaker passwords.[67]

In December 2009, a major password breach of the Rockyou.com website occurred that led to the release of 32 million passwords. The hacker then leaked the full list of the 32 million passwords (with no other identifiable information) to the Internet. Passwords were stored in cleartext in the database and were extracted through a SQL injection vulnerability. The Imperva Application Defense Center (ADC) did an analysis on the strength of the passwords.[68]

In June 2011, NATO (North Atlantic Treaty Organization) experienced a security breach that led to the public release of first and last names, usernames, and passwords for more than 11,000 registered users of their e-bookshop. The data was leaked as part of Operation AntiSec, a movement that includes Anonymous, LulzSec, as well as other hacking groups and individuals. The aim of AntiSec is to expose personal, sensitive, and restricted information to the world, using any means necessary.[69]

On July 11, 2011, Booz Allen Hamilton, a consulting firm that does work for the Pentagon, had their servers hacked by Anonymous and leaked the same day. "The leak, dubbed 'Military Meltdown Monday,' includes 90,000 logins of military personnel—including personnel from USCENTCOM, SOCOM, the Marine corps, various Air Force facilities, Homeland Security, State Department staff, and what looks like private sector contractors." [70] These leaked passwords wound up being hashed in SHA1, and were later decrypted and analyzed by the ADC team at Imperva, revealing that even military personnel look for shortcuts and ways around the password requirements.[71]

Alternatives to passwords for authentication

The numerous ways in which permanent or semi-permanent passwords can be compromised has prompted the development of other techniques. Some are inadequate in practice, and in any case few have become universally available for users seeking a more secure alternative.[72] A 2012 paper[73] examines why passwords have proved so hard to supplant (despite numerous predictions that they would soon be a thing of the past[74]); in examining thirty representative proposed replacements with respect to security, usability and deployability they conclude "none even retains the full set of benefits that legacy passwords already provide."

**Single-use passwords.** Having passwords that are only valid once makes many potential attacks ineffective. Most users find single-use passwords extremely inconvenient. They have, however, been widely implemented in personal online banking, where they are known as Transaction Authentication Numbers (TANs). As most home users only perform a small number of transactions each week, the single-use issue has not led to intolerable customer dissatisfaction in this case.

**Time-synchronized one-time passwords** are similar in some ways to single-use passwords, but the value to be entered is displayed on a small (generally pocketable) item and changes every minute or so.

**PassWindow one-time passwords** are used as single-use passwords, but the dynamic characters to be entered are visible only when a user superimposes a unique printed visual key over a server-generated challenge image shown on the user's screen.

**Access controls based on public-key cryptography** e.g. ssh. The necessary keys are usually too large to memorize (but see proposal Passmaze)[75] and must be stored on a local computer, security token or portable memory device, such as a USB flash drive or even floppy disk. The private key may be stored on a cloud service provider, and activated by the use of a password or two-factor authentication.

**Biometric methods** promise authentication based on unalterable personal characteristics, but currently (2008) have high error rates and require additional hardware to scan,[needs update] for example, fingerprints, irises, etc. They have proven easy to spoof in some famous incidents testing commercially available systems, for example, the gummie fingerprint spoof demonstration,[76] and, because these characteristics are unalterable, they cannot be changed if compromised; this is a highly important consideration in access control as a compromised access token is necessarily insecure.

**Single sign-on technology** is claimed to eliminate the need for having multiple passwords. Such schemes do not relieve users and administrators from choosing reasonable single passwords, nor system designers or administrators from ensuring that private access control information passed among systems enabling single sign-on is secure against attack. As yet, no satisfactory standard has been developed.

**Envaulting technology** is a password-free way to secure data on removable storage devices such as USB flash drives. Instead of user passwords, access control is based on the user's access to a network resource.

**Non-text-based passwords**, such as graphical passwords or mouse-movement based passwords.[77] Graphical passwords are an alternative means of authentication for log-in intended to be used in place of conventional password; they use images, graphics or colours instead of letters, digits or special characters. One system requires users to select a series of faces as a password, utilizing the human brain's ability to recall faces easily.[78] In some implementations the user is required to pick from a series of images in the correct sequence in order to gain access.[79] Another graphical password solution creates a one-time password using a randomly generated grid of images. Each time the user is required to authenticate, they look for the images that fit their pre-chosen categories and enter the randomly generated alphanumeric character that appears in the image to form the one-time password.[80][81] So far, graphical passwords are promising, but are not widely used. Studies on this subject have been made to determine its usability in the real world. While some believe that graphical passwords would be harder to crack, others suggest that people will be just as likely to pick common images or sequences as they are to pick common passwords.[citation needed]

**2D Key (2-Dimensional Key)**[82] is a 2D matrix-like key input method having the key styles of multiline passphrase, crossword, ASCII/Unicode art, with optional textual semantic noises, to create big password/key beyond 128 bits to realize the MePKC (Memorizable Public-Key Cryptography)[83] using fully memorizable private key upon the current private key management technologies like encrypted private key, split private key,

and roaming private key.

Cognitive passwords use question and answer cue/response pairs to verify identity.

"The password is dead"

"The password is dead" is a recurring idea in computer security. The reasons given often include reference to the usability as well as security problems of passwords. It often accompanies arguments that the replacement of passwords by a more secure means of authentication is both necessary and imminent. This claim has been made by numerous people at least since 2004.[74][84][85][86][87][88][89][90]

Alternatives to passwords include biometrics, two-factor authentication or single sign-on, Microsoft's Cardspace, the Higgins project, the Liberty Alliance, NSTIC, the FIDO Alliance and various Identity 2.0 proposals.[91][92]

However, in spite of these predictions and efforts to replace them passwords are still the dominant form of authentication on the web. In "The Persistence of Passwords", Cormac Herley and Paul van Oorschot suggest that every effort should be made to end the "spectacularly incorrect assumption" that passwords are dead.[93] They argue that "no other single technology matches their combination of cost, immediacy and convenience" and that "passwords are themselves the best fit for many of the scenarios in which they are currently used."

Following this, Bonneau et al. systematically compared web passwords to 35 competing authentication schemes in terms of their usability, deployability, and security.[94][95] Their analysis shows that most schemes do better than passwords on security, some schemes do better and some worse with respect to usability, while every scheme does worse than passwords on deployability. The authors conclude with the following observation: "Marginal gains are often not sufficient to reach the activation energy necessary to overcome significant transition costs, which may provide the best explanation of why we are likely to live considerably longer before seeing the funeral procession for passwords arrive at the cemetery."

## CONCLUSION

A password generator is a valuable tool for enhancing online security. In conclusion, here are some key points to consider:

**Randomness:** A good password generator should create passwords that are truly random and not based on easily guessable patterns or common words.

**Length:** Longer passwords are generally more secure. Aim for at least 12 characters, but longer is better.

**Complexity:** Include a mix of uppercase letters, lowercase letters, numbers, and special characters to increase password complexity.

**Uniqueness:** Never reuse passwords across multiple accounts. A password manager can help you keep track of unique passwords.

**Regular Updates:** Change your passwords regularly, especially for critical accounts like email and banking.

**Security Awareness:** Be cautious about where and how you store generated passwords. Avoid writing them down in easily accessible places.

**Two-Factor Authentication (2FA):** Whenever possible, enable 2FA on your accounts for an additional layer of security.

**Password Manager:** Consider using a reputable password manager to generate, store, and automatically fill in your passwords. They can help you manage complex and unique passwords for all your accounts.

**Passphrases:** For some situations, using a memorable passphrase can be just as secure as a complex password. These are often longer and made up of random words or phrases.

**Stay Informed:** Keep up-to-date with the latest security best practices and be aware of common threats like phishing.

In summary, using a strong and unique password generated by a reliable password generator is essential for protecting your online accounts and personal information. However, remember that security is an ongoing process, and staying informed and vigilant is just as important as having a strong password.