# Password Policy

A recent analysis of password security found that in a collection of 15,212,645,925 leaked passwords. Only 2,217,015,490 were unique (Meyer, 2021).  Intuitively, the most commonly used ones being incredibility predictable. The top ten being some combination or variation on ascending numbers one through nine, ‘123456789’ or strings such as ‘qwerty’ and ‘password’. As such, in order to maintain a secure system. A set of rules or policy should be used to encourage users to utilize stronger passwords. Such a policy would typically set out.

Password length

A minimum length often around eight characters. Occasionally a maximum length. When working with legacy systems for example.

Password composition

Requiring the use of numbers, special characters as well as both upper- and lower-case letters. Prohibiting the use of certain keywords may also be helpful. Denying a user from setting their password as their own personal details, username, or ID number for example. This can be extended to blocking the use of the company name or any other identifiable keyword that can be linked to the user.

Password duration

Requiring a password be changed after an arbitrary number of days, usually around 90. Can be found in many policies.  The benefit of this practice however is debatable. A user may find it difficult to generate both secure and memorable passwords. This problem is further compounded if the user has to come up with completely original passwords each time. Such strict policies may lead to users writing passwords down in easily accessible places.

# Encryption

To conform to data protection rules an appropriate level of protections must be provided to secure personal data. Encryption is a low-cost solution with many varieties to suit many different situations and is a suggested method of security from the UK GDPR. Encryption takes data (plaintext) and converts it into an unintelligible form (ciphertext). Two types of encryption are most commonly used. Symmetric and asymmetric. Both types essentially apply an algorithm with a variable or key to data to make it unreadable.  Symmetric encryption involves the use of a single secret key that is used both to encrypt and decrypt data. The speed and efficiency of this method lends itself to larger amounts of data, such as databases. The use of a single key however means that symmetric encryption alone is not suitable for the public transmission of sensitive data. Asymmetric encryption makes use of a pair of keys. One private and the other public. Without both keys it is much harder for an unauthorised person to read the data. As symmetric encryption algorithms are faster at handling large amounts of data but are prone to a man-in-the-middle type of attack. A symmetric encryption key could be encrypted asymmetrically then sent more securely.

# Buffer Overflow

A buffer overflow is where there is an overflow of temporary data being held within memory that transfers to another location. How the buffer gets in an overflow state is when that memory capacity has gone past its limit due to this while the program is trying to write the data that is coming into the buffer memory location while this happens it will overwrite the memory location.

An example of this could be a pin code system where the amount of numbers the user is poses to be 4 bytes of data long but if you're able to put more than that the program could write that data past the buffer.

For someone who is trying to exploit a program in this way would be called an attacker due them exploiting the weakness of the program by overwriting memory. They would want to do this because it will allow the file path of the program that they are exploiting to change and when that happens it can change the programs data to give out pieces of data that it should not for instance passwords and usernames could be leaked. An attacker can also write their own instructions in this way allowing them to have full control of the program to their liking.

There are two different types of buffer overflow attacks one is stack based buffer overflows which utilize stack memory that is used within the execution timer of a certain function.

Heap based buffer overflows floods the programs memory so that the program so that it does runtime operations.

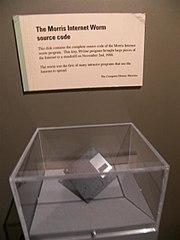
Preventing a buffer overflow attack can be done in many ways one of the easiest ways of doing so is using a language that does not really let them occur but this way is not the best practice for everyone. If you can work out if there is a flaw to a buffer overflow exploit within your code then that is a very important skill to have because you can prevent this attack happening and causing lots of damage.

C as a language can be exploited heavily for buffer overflow attacks mainly due to its ability to have direct access to memory. The best way to handle these problems are using secure practices that will handle the problem.

Some languages like Python and Java do not need to have to deal with problems like this.

Examples of real buffer overflow problems:

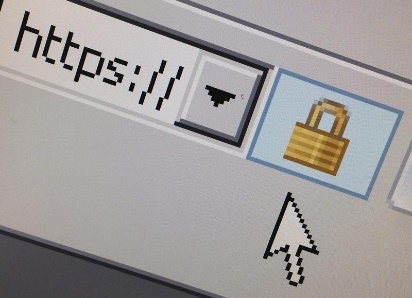
The Morris Worm is a computer worm that exploited vulnerabilities of the systems that had the worm. One of the vulnerabilities it exploits is a buffer overflow in a network service.



MissingNo is from the videogames pokemon red and blue. I believe that it is one of the easiest and a real scenario of a buffer overflow mainly due to how it performed within the games. This bug occurs when the player goes past the limit of 128 for items caused by events that can be triggered due this buffer overflow; it can corrupt your save file and make that file very unplayable.



Heartbleed is a security bug within a OpenSSL cryptography library that had the bug within the Transport Layer Security protocol. This bug allowed stealing of private keys, usernames, passwords and session cookies on the websites that were using OpenSSL. This bug was later patched within six days of discovering it. How heartbleed worked was by sending data to the server that went past a limit of that element of a website. This a buffer overflow attack that was very dangerous to the internet.



# SQL Injection

SQL injection is a common form of attack that can be made against any application that queries an SQL database in some way. If your application uses an SQL database, you need to be aware of SQL injection and protect against it, or all your usernames, passwords, and other data could be leaked or destroyed.

To illustrate how this type of attack works, here is an example:

A website selling products has a search feature. An end user can type into a textbox and click a search button to show all products matching the string they write in the box.

To achieve this, the website makes a POST request. It takes the string the user writes into the box and constructs an SQL query out of it. That query is then used to retrieve some data from the database, and that data is then sent back and displayed on the website.

Suppose the query is constructed in PHP like this:

SELECT \* FROM products WHERE name = user\_input;

If the user types in something innocuous, like “Soap”, we get a query like this:

SELECT \* FROM products WHERE name = ‘Soap’;

However, since the string is not processed in any way before the SQL query is constructed, this means that the user can add their own SQL inside of this constructed query. For example, the user could type:

Soap’ OR ‘1’=‘1

Which would result in the following valid SQL statement being formed:

SELECT \* FROM products WHERE name = ‘Soap’ OR ‘1’=‘1’;

This would result in the entire products table being sent back to the user, not just the requested row.

That type of SQL injection is not a huge problem. Apart from taking a long time to load all the products on the entire website, this would not do any damage. However, you could do something more malicious, for example typing in this string:

Soap’; DROP TABLE products; --

Would result in the following sequence of SQL statements being formed:

SELECT \* FROM products WHERE name = ‘Soap’;

DROP TABLE products;

The -- at the end of the string indicates a comment. This means that the rest of the query constructed by the server’s host language (such as PHP) will be interpreted as a comment and ignored.

This process of ending a statement and inserting another straight after does not work for all database management software. SQL is a language shared by several platforms and implemented slightly differently on each. Some of these platforms do allow multiple SQL statements in one line, and so long as none are malformed, they will execute them in sequence. On a platform that supports it, this set of SQL statements will delete the entire products table from the server.

This type of attack can be about guesswork, as the attacker must sometimes guess what the identifiers are for the SQL server including names of tables and columns. However sometimes people get lucky, or sometimes they are clever when they construct their queries.

If the attacker knows what type of database management software is in use, they can target their attacks much more directly. For example, if the attacker knows the server is using PostgreSQL, they can use a command like

SELECT table\_name FROM information\_schema.tables WHERE table\_schema='public';

This will list all tables in the database. From there they can construct queries to directly attack tables that contain sensitive data, such as usernames and passwords.

### **How to protect against SQL Injection**

SQL injection is easy to protect against. All you have to do is sanitise user input. In other words, all the text that a user enters must be treated as text and not as SQL.

One straightforward way to do this is to escape all characters that have some special meaning in SQL. For example, each single quote in the input string should be replaced by two single quotes in order to form an SQL string literal. There is even a built in PHP function that can do this for you: mysqli::real\_escape\_string()

# References

Meyer, B. (2021, Feb 09). *Most common passwords: latest 2021 statistics*. Retrieved from cybernews: https://cybernews.com/best-password-managers/most-common-passwords/

2021. [online] Available at: <https://www.youtube.com/watch?v=rE5dW3BTpn4&ab\_channel=TomScott> [Accessed 25 February 2021].

2021. [online] Available at: <https://www.youtube.com/watch?v=1S0aBV-Waeo&ab\_channel=Computerphile> [Accessed 25 February 2021].

Learning Center. 2021. *What is a Buffer Overflow | Attack Types and Prevention Methods | Imperva*. [online] Available at: <https://www.imperva.com/learn/application-security/buffer-overflow/> [Accessed 25 February 2021].

En.wikipedia.org. 2021. *MissingNo.*. [online] Available at: <https://en.wikipedia.org/wiki/MissingNo.> [Accessed 25 February 2021].

En.wikipedia.org. 2021. *Heartbleed*. [online] Available at: <https://en.wikipedia.org/wiki/Heartbleed>

[Accessed 25 February 2021].

En.wikipedia.org. 2021. *Morris worm*. [online] Available at: <https://en.wikipedia.org/wiki/Morris\_worm> [Accessed 25 February 2021].

The PHP Group (2006) PHP:mysqli::real\_escape\_string. Available at: <https://www.php.net/manual/en/mysqli.real-escape-string.php>

Microsoft (2013) SQL Injection [online] Available at: https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms161953(v=sql.105)