# Linux shells and Hashing
# Day 24/365
# Linux shells

Types of shells

- Bash (Bourne Again Shell)
  - default shell for most distros
  - tab completion
  - history file and logs of commands
- Fish (Friendly Interactive Shell)
  - simple syntax
  - auto spell correction
  - cool themes
  - color highlighting
  - scripting, tab completion command history
- Zsh (Z shell)
  - not installed by default
  - modern shell
  - advanced tab completion and writing scripts
  - auto spell correction
  - extensive customization (slower than other shells)
  - tab completion, command history

---

https://hashcat.net/wiki/doku.php?id=example_hashes

# What are Hashes?

A hash is a way of taking a piece of data of any length and representing it in another fixed-length form. This process masks the original value of the data. The hash value is obtained by running the original data through a hashing algorithm. Many popular hashing algorithms exist, such as MD4, MD5, SHA1 and NTLM. Let's try and show this with an example:

If we take "polo", a string of four characters, and run it through an MD5 hashing algorithm, we end up with an output of `b53759f3ce692de7aff1b5779d3964da`, a standard 32-character MD5 hash.

Likewise, if we take “polomints”, a string of 9 characters, and run it through the same MD5 hashing algorithm, we end up with an output of `584b6e4f4586e136bc280f27f9c64f3b`, another standard 32-character MD5 hash.

# What Makes Hashes Secure?

Hashing functions are designed as one-way functions. In other words, it is easy to calculate the hash value of a given input; however, it is a hard problem to find the original input given the hash value. In simple terms, a hard problem quickly becomes computationally infeasible in computer science. This computational problem has its roots in mathematics as P vs NP.

In computer science, P and NP are two classes of problems that help us understand the efficiency of algorithms:

- **P (Polynomial Time)**: Class P covers the problems whose solution can be found in polynomial time. Consider sorting a list in increasing order. The longer the list, the longer it would take to sort; however, the increase in time is not exponential.
- **NP (Non-deterministic Polynomial Time)**: Problems in the class NP are those for which a given solution can be checked quickly, even though finding the solution itself might be hard. In fact, we don’t know if there is a fast algorithm to find the solution in the first place.

While this is a fascinating mathematical concept that proves fundamental to computing and cryptography, it is entirely outside the scope of this room. But abstractly, the algorithm to hash the value will be “P” and can, therefore, be calculated reasonably. However, an “un-hashing” algorithm would be “NP” and intractable to solve, meaning that it cannot be computed in a reasonable time using standard computers.

**Hashing**, as already stated, is a process that takes input data and produces a hash value, a fixed-size string of characters, also referred to as digest. This hash value uniquely represents the data, and any change in the data, no matter how small, should lead to a change in the hash value. Hashing should not be confused with encryption or encoding; hashing is one-way, and you can’t reverse the process to get the original data.

**Encoding** converts data from one form to another to make it compatible with a specific system. ASCII, UTF-8, UTF-16, UTF-32, ISO-8859-1, and Windows-1252 are valid encoding methods for the English language. Note that UTF-8, UTF-16, and UTF-32 are Unicode encodings, and they can represent characters from other languages, such as Arabic and Japanese.

Another type of encoding commonly used when sending or saving data is not for any specific language. Examples include Base32 and Base64 encoding. Consider the following example of using `base64` to encode and decode.

Terminal

```
`strategos@g5000 ~> base64 TryHackMe VHJ5SGFja01lCg== strategos@g5000 ~> base64 -d VHJ5SGFja01lCg== TryHackMe`
```

Encoding should not be confused with encryption, as using a specific encoding does not protect the confidentiality of the message. Encoding is reversible; anyone can change the data encoding with the right tools.

Only **encryption**, which we covered in the previous rooms, protects data confidentiality using a cryptographic cipher and a key. Encryption is reversible, provided we know the cipher and can access the key.

# Useful links

https://hashcat.net/wiki/doku.php?id=example_hashes
https://github.com/danielmiessler/SecLists