Entropy analysis.

Entropy analysis is particularly helpful when we are examining smaller and more uniform datasets. Things that will contrast against baseline. It is also very useful to unusually high entropy in protocols that are typically clear text. For example, DNS.

```
saf-lx@Saf-Ubuntu:~/Desktop/SOC-201/0cdf57a5$ file hello.exe
hello.exe: PE32+ executable (console) x86-64, for MS Windows, 19 sections
saf-lx@Saf-Ubuntu:~/Desktop/SOC-201/0cdf57a5$ hexdump -C -n 10 hello.exe
00000000  4d 5a 90 00 03 00 00 00  04 00                    |MZ........|
0000000a
saf-lx@Saf-Ubuntu:~/Desktop/SOC-201/0cdf57a5$
```

Here we have this file, hello.exe which is PE32 executable file with the magic bits MZ. Standard windows executable file.

```
hello.exe  hello.gz  precooked
saf-lx@Saf-Ubuntu:~/Desktop/SOC-201/0cdf57a5$
```

Simple compression using compression algorithm.

```
hello.exe  hello.gz  precooked
saf-lx@Saf-Ubuntu:~/Desktop/SOC-201/0cdf57a5$
```

Let's encrypt the hello.exe file using AES-256 algorithm.

```
saf-lx@Saf-Ubuntu:~/Desktop/SOC-201/0cdf57a5$ openssl enc -aes-256-cbc -in hello.exe -out hello.bin -nosalt
enter AES-256-CBC encryption password:
Verifying - enter AES-256-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

We will now use the entropy command or ent. Simple and powerful randomness test utility that we can use to analyze the entropy and sort of statistical characteristics of a file's contents. Usually, we are going to be dealing with the binary files like executables or different payloads or strings or data blobs.

```
saf-lx@Saf-Ubuntu:~/Desktop/SOC-201/0cdf57a5$ ent hello.exe
Entropy = 5.898227 bits per byte.

Optimum compression would reduce the size
of this 249544 byte file by 26 percent.

Chi square distribution for 249544 samples is 4960751.10, and randomly
would exceed this value less than 0.01 percent of the times.

Arithmetic mean value of data bytes is 53.1615 (127.5 = random).
Monte Carlo value for Pi is 3.882087040 (error 23.57 percent).
Serial correlation coefficient is 0.266194 (totally uncorrelated = 0.0).
saf-lx@Saf-Ubuntu:~/Desktop/SOC-201/0cdf57a5$
```

Our entropy here is: Entropy = 5.898227 bits per byte.

If we do with .gz file then,

```
saf-lx@Saf-Ubuntu:~/Desktop/SOC-201/0cdf57a5$ ent hello.gz
Entropy = 7.996619 bits per byte.

Optimum compression would reduce the size
of this 99963 byte file by 0 percent.

Chi square distribution for 99963 samples is 471.31, and randomly
would exceed this value less than 0.01 percent of the times.

Arithmetic mean value of data bytes is 128.0779 (127.5 = random).
Monte Carlo value for Pi is 3.114765906 (error 0.85 percent).
Serial correlation coefficient is -0.017301 (totally uncorrelated = 0.0).
saf-lx@Saf-Ubuntu:~/Desktop/SOC-201/0cdf57a5$
```

Entropy = 7.996619 bits per byte. Gzip file has considerably high entropy and no compressibility, making it almost look random.

Next with hello.bin

```
saf-lx@Saf-Ubuntu:~/Desktop/SOC-201/0cdf57a5$ ent hello.bin
Entropy = 7.999195 bits per byte.

Optimum compression would reduce the size
of this 249552 byte file by 0 percent.

Chi square distribution for 249552 samples is 277.91, and randomly
would exceed this value 15.50 percent of the times.

Arithmetic mean value of data bytes is 127.3758 (127.5 = random).
Monte Carlo value for Pi is 3.141950375 (error 0.01 percent).
Serial correlation coefficient is -0.001347 (totally uncorrelated = 0.0).
saf-lx@Saf-Ubuntu:~/Desktop/SOC-201/0cdf57a5$
```

Among all, the entropy file has higher entropy, tiny bit higher than gzip file.

Encryption aims to produce near-perfect randomness, while compression produces a very high but not always perfect, entropy.

If the file has high entropy, it defines structure and format anomaly for us to look into. Main takeaway is attacker can hide or obfuscate the malicious file with compressions and encryptions and we can identify it.