

## cloak\_ss\_file

# Cloak: Payload Encoder, Obfuscation, Detection, Reporting, and Decoding Framework

### 1. Help Options :

This screenshot shows the available command-line arguments for Cloak. It demonstrates how users can select payloads, encoders, obfuscators, and the decode flag.

```
[x]-[gr0ot@parrot]-(~/Desktop/cloak]
└─$ python3 main.py -h
usage: main.py [-h] --payload PAYLOAD
                [--encoder {base64,xor,rot13,hex,base32,base85}] [--key KEY]
                [--obfuscator {insert,split,escape,reverse}] [--decode]

Cloak: Payload Encoder/Obfuscation/Detection/Decoding Framework

options:
-h, --help            show this help message and exit
--payload PAYLOAD, -p PAYLOAD
                    Raw payload string (for encode/decode)
--encoder {base64,xor,rot13,hex,base32,base85}, -e {base64,xor,rot13,hex,base32,base85}
                    Encoding method
--key KEY, -k KEY    Key for XOR encoding/decoding
--obfuscator {insert,split,escape,reverse}, -o {insert,split,escape,reverse}
                    Optional obfuscation method
--decode, -d          Decode instead of encode/obfuscate
```

### 2. Normal String Encoding & Decoding with Obfuscation :

Here, a simple string is encoded using Base64 with an obfuscation method applied. The same string is then decoded back, showing that reversible transformations work correctly.

```
[gr0ot@parrot] -[~/Desktop/cloak]
└─ $python3 main.py --p "hello" -e base64 -o reverse
== Cloak Execution ==
[ENCODER] BASE64, result, detected_original, detected_transformed)
[OBFUSCATOR] reverse
[OUTPUT] =8GbsVGA
[DETECTION] Original: BYPASSED
[DETECTION] Transformed: BYPASSED else 'NONE' ")
[EFFECTIVENESS] NEUTRAL – Both original and transformed bypassed.

--- Report saved to results/report_20260204_143730.txt ---
[gr0ot@parrot] -[~/Desktop/cloak]
└─ $python3 main.py --p "=8GbsVGA" -e base64 -o reverse -d
== Cloak Decoding === name) ---")
[DECODER] BASE64
[DE-OBFUSCATOR] reverse
[OUTPUT] hello
```

---

### 3. Detectable String Encoding & Decoding with Obfuscation :

This example uses a payload that would normally be flagged by detection. After encoding and obfuscation, the transformed payload bypasses detection. The next run shows successful decoding back to the original string.

```
[gr0ot@parrot] -[~/Desktop/cloak]
└─ $python3 main.py -p "import os; os.system('/bin/sh')" -e rot13 -o split
== Cloak Execution ==
[ENCODER] ROT13 ))", payload)
[OBFUSCATOR] split_pairs)
[OUTPUT] vz+cb+eg+ b+f;+ b+f.+f1+fg+rz+( '+/o+va+/f+u'+)
[DETECTION] Original: DETECTED
[DETECTION] Transformed: BYPASSED
[EFFECTIVENESS] SUCCESS – Transformation bypassed detection.

imiter, "")
--- Report saved to results/report_20260204_145509.txt ---
[gr0ot@parrot] -[~/Desktop/cloak]
└─ $python3 main.py -p "vz+cb+eg+ b+f;+ b+f.+f1+fg+rz+( '+/o+va+/f+u'+)" -e rot13 -o split -d
== Cloak Decoding ===
[DECODER] ROT13 tadata,
[DE-OBFUSCATOR] split exactly which chars were added,
[OUTPUT] import os; os.system('/bin/sh')
```

---

### 4. Saved Report Output :

This screenshot highlights Cloak's reporting feature. Each run generates a detailed report file with detection results, verdicts, and hashes, saved automatically in the results/ directory.

The screenshot shows a terminal window titled "report\_20260204\_145509.txt (~/Desktop/cloak/results) - Pluma". The window contains the following text:

```
1 === Cloak Report ===
2 Timestamp: 2026-02-04T14:55:09.560528
3
4 Original Payload: import os; os.system('/bin/sh')
5 Original SHA256: 4b2e1e3ab250ccb7ac9ef58bba380eef04cf1b824a4242cfdddfed2975cfaea
6 Transformed Payload Preview: vz+cb+eg+ b+f;+ b+f.+fl+fg+rz+( '+/o+va+/f+u' +)
7 Transformed SHA256: 1cd7c9399b4a92798742530cd4be1abe82566589270bac27b3bb79b313ef9496
8
9 Original Detection: DETECTED
10 Transformed Detection: BYPASSED
11 Effectiveness: SUCCESS – Transformation bypassed detection.
```

## 5. Results Directory Overview :

This view shows the results/ folder where all generated reports are stored. It confirms that Cloak maintains organized output files for later review.

