CTF Report Forensics - DISKO 1

Platform: picoCTF

Challenge Name: DISKO 1

Category: Forensics

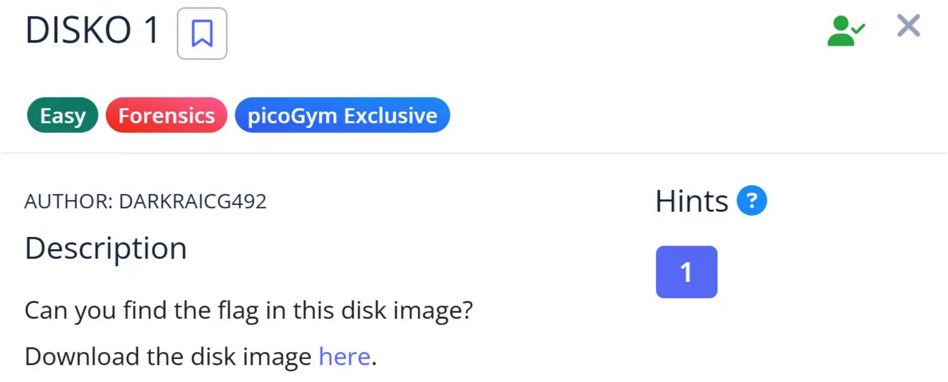
Difficulty: Easy

Submitted By: Gurleen Kaur Brar

Objective

The goal of the challenge was to analyze a disk image file and extract the embedded flag. This required basic forensic skills, such as viewing raw strings from files and inspecting the contents for patterns or indicators that resemble a CTF flag.

Challenge Description



DISKO 1

AUTHOR: DARKRAICG492

Description

Can you find the flag in this disk image?

Download the disk image here.

Files and Tools Used

- **File Provided:** `disko-1.dd.gz` (compressed disk image)
- **Tools Used:**
  - Kali Linux Terminal

   - `gunzip` – for decompressing `.gz` files
   - `strings` – for extracting readable strings from binary
   - `nano` – for viewing output

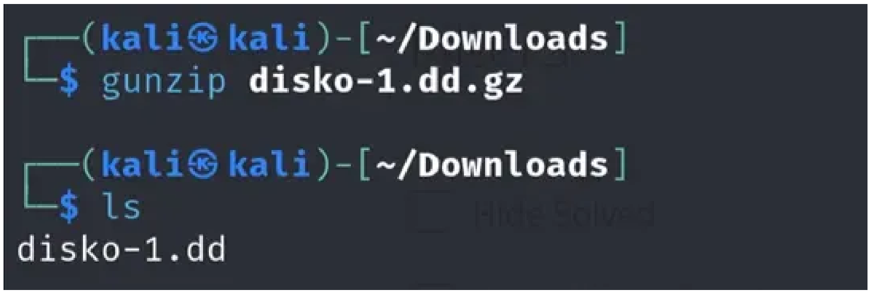
Step-by-Step Process

Step 1: Decompress the Disk Image

The challenge provided a Gzip-compressed disk image file. To access the contents, I decompressed it using the `gunzip` command:

```
gunzip disko-1.dd.gz
```

After running `ls`, I confirmed that `disko-1.dd` was extracted.



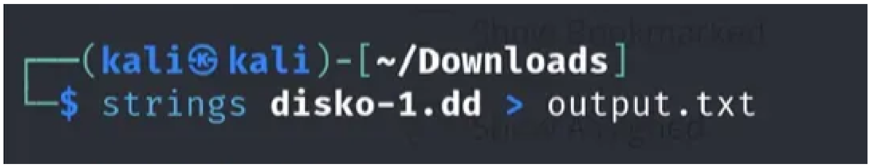
```
┌──(kali㉿kali)-[~/Downloads]
└─$ gunzip disko-1.dd.gz

┌──(kali㉿kali)-[~/Downloads]
└─$ ls
disko-1.dd
```

Step 2: Extract Strings from the Disk Image

I used the `strings` command to extract all printable characters from the raw disk image and redirected the output into a text file for easier searching.

```
strings disko-1.dd > output.txt
```

This step is crucial because disk images often contain plaintext fragments that could include flags, commands, or metadata.



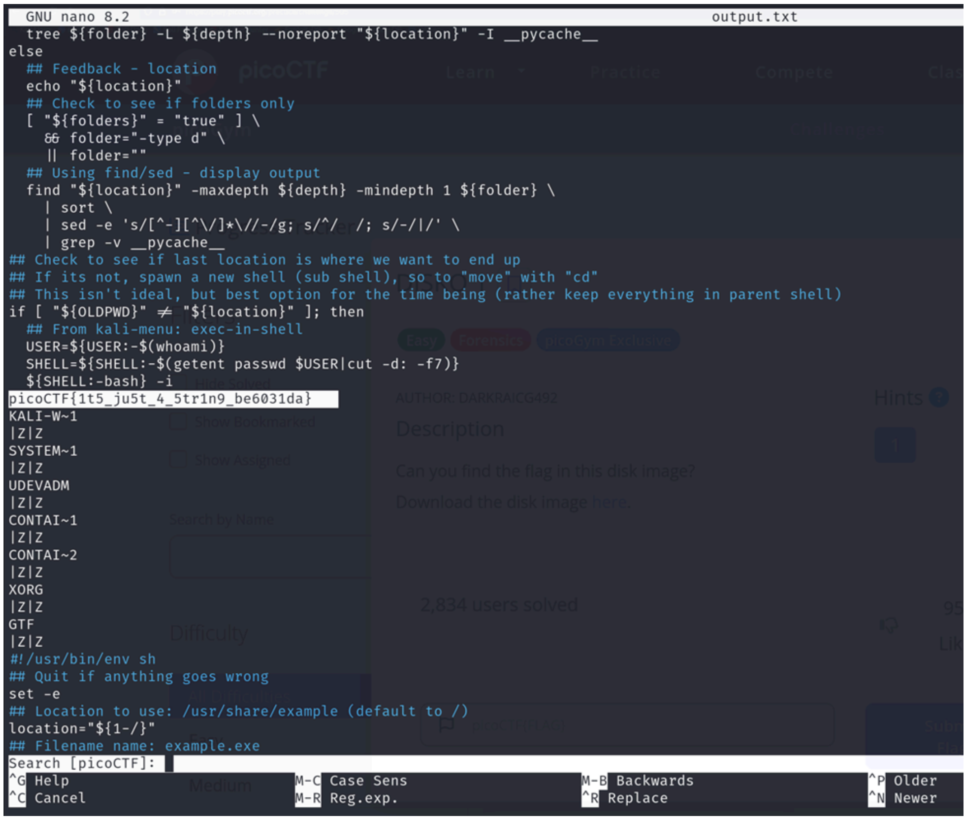
```
┌──(kali㉿kali)-[~/Downloads]
└─$ strings disko-1.dd > output.txt
```

Step 3: Search for the Flag

I opened the `output.txt` file using `nano`:

```
nano output.txt
```

I manually scrolled through the file, looking for anything that resembled a flag. Near the bottom of the file and found a valid-looking flag in the format expected by picoCTF:



```
  GNU nano 8.2                                                    output.txt
  tree ${folder} -L ${depth} --noreport "${location}" -I __pycache__
else
  ## Feedback - location
  echo "${location}"
  ## Check to see if folders only
  [ "${folders}" = "true" ] \
    && folder="-type d" \
    || folder=""
  ## Using find/sed - display output
  find "${location}" -maxdepth ${depth} -mindepth 1 ${folder} \
    | sort \
    | sed -e 's/[^-][^\/]*\//-/g; s/^/  /; s/-/|/' \
    | grep -v __pycache__
## Check to see if last location is where we want to end up
## If its not, spawn a new shell (sub shell), so to "move" with "cd"
## This isn't ideal, but best option for the time being (rather keep everything in parent shell)
if [ "${OLDPWD}" ≠ "${location}" ]; then
  ## From kali-menu: exec-in-shell
  USER=${USER:-$(whoami)}
  SHELL=${SHELL:-$(getent passwd $USER|cut -d: -f7)}
  ${SHELL:-bash} -i
picoCTF{1t5_ju5t_4_5tr1n9_be6031da}
KALI-W~1
|Z|Z
SYSTEM~1
|Z|Z
UDEVADM
|Z|Z
CONTAI~1
|Z|Z
CONTAI~2
|Z|Z
XORG
|Z|Z
GTF
|Z|Z
#!/usr/bin/env sh
## Quit if anything goes wrong
set -e
## Location to use: /usr/share/example (default to /)
location="${1-/}"
## Filename name: example.exe
Search [picoCTF]:
^G Help          M-C Case Sens        M-B Backwards        ^P Older
^C Cancel        M-R Reg.exp.         ^R Replace           ^N Newer
```

Flag Submitted

The extracted flag was:

```
picoCTF{it5_ju5t_4_Str1n9_be6031da}
```

I submitted this flag in the challenge and confirmed it as correct.

Forensics

Easy

DISKO 1

2,835 solves

95%