CSAW'25 Finals

CSAW 2025 finals, hosted online by NYU (https://www.csaw.io/).

Challenges available here: https://github.com/osirislab/CSAW-CTF-2025-Finals-Public

- hehexd

hehexd

Written by `@palee.`. Also hosted on Github.

hehexd was a misc challenge during the CSAW '25 finals.

When i first installed the files and unzipped it I was initially shown this monstrocity of a file hierarchy.

```
└── a
    ├── a
    │   ├── a
    │   ├── b
    │   ├── c
    │   └── d
    └── b
        ├── a
        │   ├── a
        │   ├── b
        │   ├── c
        │   ├── d
        │   └── e
        ├── b
        │   ├── a
        │   └── b
        └── c
            ├── a
            ├── b
            ├── c
            ├── d
            ├── e
            ├── f
            ├── g
            ├── h
            ├── i
            ├── j
            ├── k
            ├── l
```

```
        ├── m
        ├── n
        ├── o
        ├── p
        ├── q
        ├── r
        ├── s
        ├── t
        ├── u
        ├── v
        ├── w
        ├── x
        ├── y
        ├── z
        ├── za
        ├── zb
        ├── zc
        ├── zd
        ├── ze
        ├── zf
        ├── zg
        ├── zh
        ├── zi
        ├── zj
        ├── zk
        ├── zl
        ├── zm
        ├── zn
        ├── zo
        ├── zp
        └── zq
```

After seeing this my first though was to try `grep` or `find` to find any files, but unfortunately there was nothing resembling a flag. This then gave me the great idea of searching through every single directory to try and find a file.

- In retrespect there is probably a command to do this but I didn't think of that at the time.

After getting through all of `a/a`, `a/b/a` and `a/b/b` there was nothing, but `a/b/c` piqued my interest since there was a lot of directories, and it looked like there could be a pattern there hiding the flag.

The format of each directory was of the form:

```
.
├── a
│   ├── a
│   ├── b
│   ├── c
│   └── d
└── b
    ├── a
    ├── b
    ├── c
    └── d
```

which is interesting since 8 bits could very much represent a byte for a flag.

In each of these directories there would either be a folder or there wouldn't, which is likely binary which could represent an ascii value.

For example, in the `a` directory it looked like

```
.
├── a
│   ├── a
│   ├── b
│   │   └── a
│   ├── c
│   │   └── a
│   └── d
└── b
    ├── a
    ├── b
    ├── c
    │   └── a
    └── d
        └── a
```

which would map to a binary output of 01100011 going top to bottom, which if you translate to an ascii value gives you 'c', which is the first character of the flag. Very promising.

To verify this I also checked the 5th directory, which is `e` and it had the same format which tracks with the flag format being 'csawctf{}'.

After this I was confident that this was the flag and started to write a python script to automate translating each directory into a binary value ready for it to be converted into an ascii value later.

```
import os
import binascii

items = os.listdir(".")
items.remove("solve.py")
items.remove(".DS_Store")
sorted_items = sorted(items)

bin = ""

for i in range(len(sorted_items)):
    value = ""
    value = value + str(len(os.listdir(sorted_items[i] + "/a/a")))
    value = value + str(len(os.listdir(sorted_items[i] + "/a/b")))
    value = value + str(len(os.listdir(sorted_items[i] + "/a/c")))
    value = value + str(len(os.listdir(sorted_items[i] + "/a/d")))
    value = value + str(len(os.listdir(sorted_items[i] + "/b/a")))
    value = value + str(len(os.listdir(sorted_items[i] + "/b/b")))
    value = value + str(len(os.listdir(sorted_items[i] + "/b/c")))
    value = value + str(len(os.listdir(sorted_items[i] + "/b/d")))
    value = value.replace("2", "1")
    bin = bin + value



n = int(bin, 2)
print(binascii.unhexlify('%x' % n))
```

This then outputs the flag `csawctf{4n07H3r_350L4n6_F0r_7H3_c0lL3c710n}`