solve

March 1, 2025

[21]:
```python
import json
import zlib
from hashlib import sha256
```

[56]:
```python
with open("flag.txt.ozed","rb") as f:
    data = f.read()
```

[57]:
```python
OZED = data[:4]
data = data[4:]
OZED
```

[57]:
```
b'OZED'
```

[58]:
```python
paddedmetadata = data[:300]
data = data[300:]
metadata = json.loads(paddedmetadata[:136])
metadata,paddedmetadata[136:]
```

[58]:
```
({'user': 'zed',
  'password_hash':
'b3a97eb583db5a940c0705e6450b81f4d702a9122d7342a25768e3d75be739be',
  'filename': 'flag.txt',
  'size': 63},
 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00')
```

[71]:
```python
default_password = b"OpenZEDdefaultpasswordtochangebeforedeployinproduction"
password_hash = bytes.fromhex(metadata["password_hash"])
metadata["password_hash"],sha256(default_password).hexdigest()
```

[71]:
```
('b3a97eb583db5a940c0705e6450b81f4d702a9122d7342a25768e3d75be739be',
 'ab6e35c53f58dcbaade511a6aac33ee3d6df83c0a97b3c64a66da4939c9b8b1e')
```

They do not use the standard password here

[60]:
```
payload = zlib.decompress(data)
```

[61]:
```
payload
```

[61]:
```
b'zed\xfc\xca\x96[u`\xd9$\x83\xa4\xd6kL\x16\xe5\x02\x9e\x92Y\xe0?e\xcf;\xa3\xe1\
xc9GO\x10\x90m\t\xc8\xd6\xaeGa\xed\xec\x1eG\x88\x99\tV\xb2\xbc\x92!\xc2\t\xf3/\x
fb\x17P5N\x08\x8d\xa0\xedI\x92\xb6\xeao\xef\xec\xf1\x96%\xac\xdb\xbf'
```

[62]:
```
def derive_password(self):
    for i in range(100):
        self.key = sha256(self.password).digest()[:16]

def generate_iv(self):
    self.iv = (self.user+self.password)[:16]
```

[63]:
```
iv = payload[:16]
username = iv[:3]
password = iv[3:]
username,password
```

[63]:
```
(b'zed', b'\xfc\xca\x96[u`\xd9$\x83\xa4\xd6kL')
```

[64]:
```
f"Missing bytes: {16-len(password)}"
```

[64]:
```
'Missing bytes: 3'
```

1 We can just brute force those 3 bytes :)

[72]:
```
from Crypto.Util.number import long_to_bytes
```

[73]:
```
keys = []
for guess in range(256**3):
    test = password + long_to_bytes(guess,3)
    hash = sha256(test).digest()[:16]

    if password_hash.startswith(hash):
        print(password + long_to_bytes(guess,3))
        keys.append(password + long_to_bytes(guess,3))
```

```
b'\xfc\xca\x96[u`\xd9$\x83\xa4\xd6kL\x80\xb9t'
```

[74]:
```
from openzedlib import aes_cbc_zed
```

[75]:
```
instance = aes_cbc_zed.AES_CBC_ZED(user=username,password=keys[0])
print(instance.decrypt(payload))
```

```
b'PWNME{49e531f28d1cedef03103af6cec79669_th4t_v3Ct0r_k1nd4_l3aky}'
```