SushiRoll Team

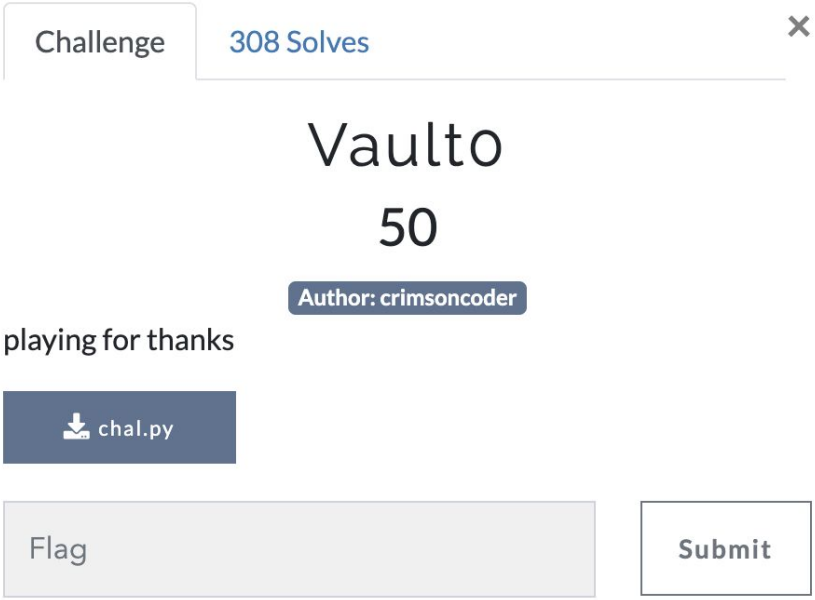
University of Puerto Rico at Mayaguez

Team Members:

- Luis Romero
- José Maldonado
- Onix Tarrats
- Roberto Guzmán

Challenges

Reversing
Vault0:



For this challenge, I simply copied over all of the hex values and converted them to Ascii with an online converter. The resulting hex was:

636173746f72734354467b72317854795f6d316e757433735f67745f73317874795f6d316e757433737d

Which translates in Ascii to:

castorsCTF{r1xTy_m1nut3s_gt_s1xty_m1nut3s}

Forensics:

Challenge 137 Solves

Leftovers

131

Author: icinta

We suspect a user has been typing faster than a computer. Our analyst don't know what to make of it, maybe you will be the one to shine light on the subject.

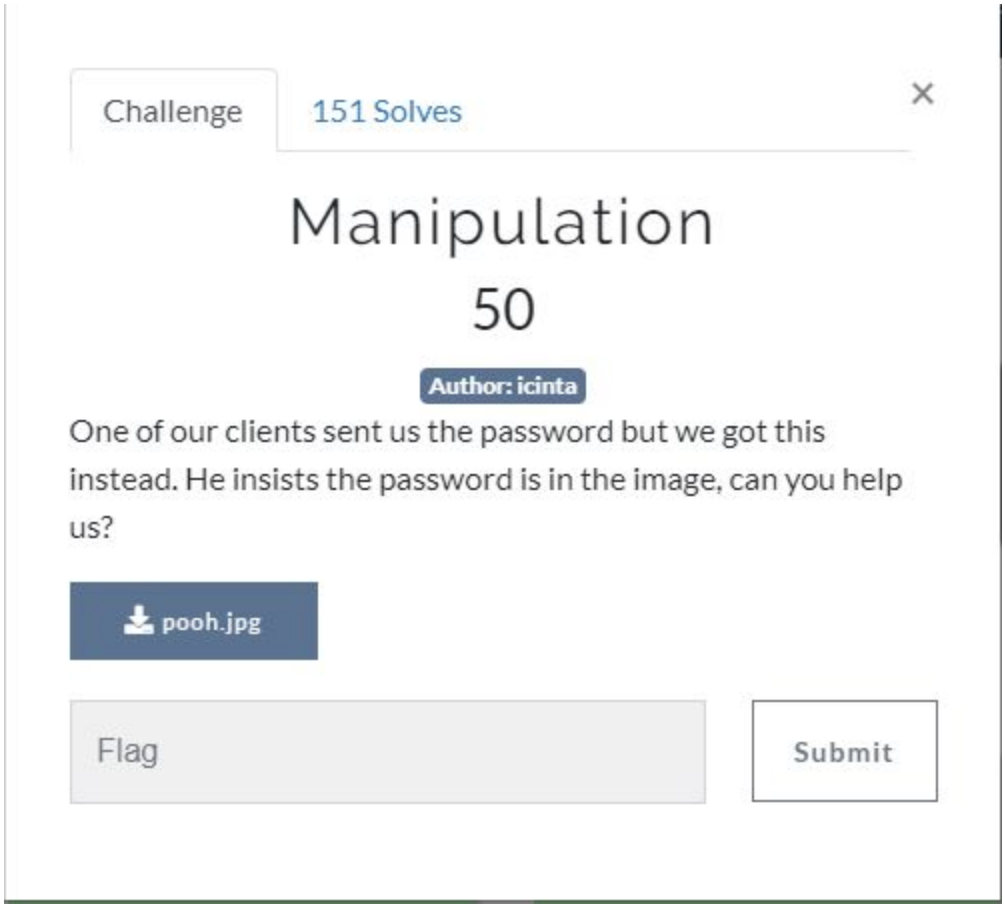
Author: icinta

interrupts.pc...

I looked through the packet capture file attached and saw that it was various keyboard interrupts. Through previous CTF experiences, I knew that packet captures could be used for USB data, and looked for a script to convert these interrupts into Ascii. Through this, I found a script that did the job, which gave me the flag:

```
myKeys = open('hexoutput.txt')
i = 1
for line in myKeys:
    bytesArray = bytearray.fromhex(line.strip())
#print "Line Number: " + str(i)
    for byte in bytesArray:
        if byte != 0:
            keyVal = int(byte)
            if keyVal in newmap:
                print newmap[keyVal]
            else:
                print "No map found for this value: " + str(keyVal)
        #print format(byte, '02X')
    i+=1
```

Manipulation:



The manipulation problem was approached using the HxD hexadecimal editor. Initially, the file was corrupted and it did not have the proper jpg signature bytes, but it had the Exif signature bytes so we were dealing with an image file. I scrolled to the end of the file end found there the first line. Also, the bytes were grouped in groups of two something I fixed with a python script.

```
file = open('pooh.jpg', 'r')
Lines = file.readlines()

result = open('resImg.jpg','w+')
b = open('bites.txt','w+')



# for line in Lines:
#     print(line)
print(Lines[0])
for i in range(len(Lines)-3):
    h = (Lines[i])[:10]
```

```
    m = (Lines[i])[10:49]
    t = (Lines[i])[49:]
    r = ""
    sl = str(m).split(" ")
    for s in sl:
        r += str(s[:2]) + " " + str(s[2:]) +" "
    print(h,"->",r[:-1], "->", t)
    result.write(h+r[:-1]+t)
    b.write(r[:-1])

h = (Lines[len(Lines)-3])[:10]
m = (Lines[len(Lines)-3])[10:24]
t = (Lines[len(Lines)-3])[24:]
r = ""
sl = str(m).split(" ")
for s in sl:
    r += str(s[:2]) + " " + str(s[2:]) + " "
result.write(h+r[:-1]+ t)
b.write(r[:-1])

b.close()
file.close()
result.close()
```

Coding

Arithmethics:

Arithmetics

50

Author: hasu

2 plus two = ?

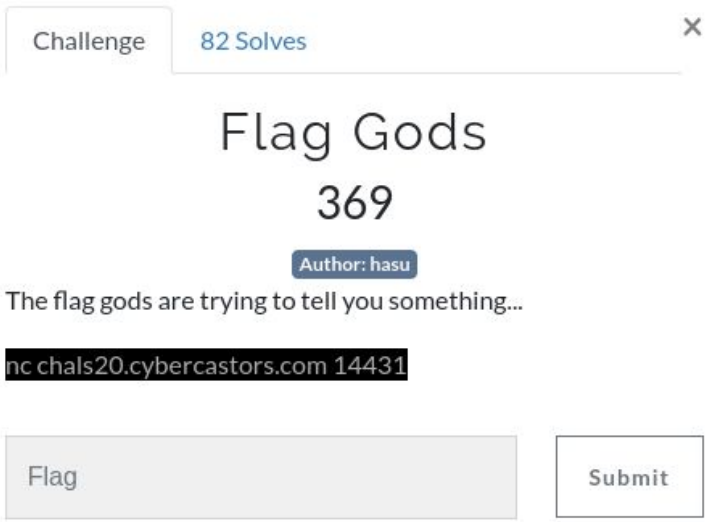
nc chals20.cybercastors.com 14429

Initially, I connected to the server with netcat and tried to do the arithmetic operations myself, but quickly realized the server was expecting answers too quickly for me to

answer them. Upon this realization, I started looking into connecting to netcat with python, and found that the built in socket library was able to do this. Through some trial and error, a script was devised to receive the arithmetic problems and send back the answer.

```python
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('chals20.cybercastors.com', 14429))
print(client.recv(1024))
client.send("\n".encode())
numbers = {'zero': 0,
    'one': 1,
    'two': 2,
    'three': 3,
    'four': 4,
    'five': 5,
    'six': 6,
    'seven': 7,
    'eight': 8,
    'nine': 9,
    'ten': 10}

operands = {
    'minus': '-',
    'plus': '+',
    'multiplied-by': '*',
    'divided-by' : '//'
}

def doit(received):
    print(received)
    if "castors" in received:
        print("U A")
        print(received)
    elif "answer" in received:
        doit(str(client.recv(1024)))
    else:
        cut = received.split("is", 1)[1][:-5]
        for key in numbers:
            if key in cut:
                cut = cut.replace(key, str(numbers[key]))

        for key in operands:
            if key in cut:
                cut = cut.replace(key, operands[key])
        cut_bytes = str(eval(cut)).encode()
        print(cut_bytes)
        client.send(cut_bytes)
        doit(str(client.recv(1024)))

doit(str(client.recv(1024)))
```

Flag Gods



Following the approach used to solve the arithmetics problem, we used python in order to connect to the flag god server. After that it was pretty straight forward, we built a script that obtained the transmitted and received msgs, transformed them into binary in order to calculate the hamming distance and then send that to the server. We did that in an infinite look until the radio was calibrated and we obtained the flag.

```
import binascii
import socket

binary = lambda x: "".join(reversed( [i+j for i,j in zip( *[
["{0:04b}".format(int(c,16)) for c in reversed("0"+x)][n::2] for n in
[1,0] ] ) ] ))

def find_str_between_strs( s, first, last ):
  try:
      start = s.index( first ) + len( first )
      end = s.index( last, start )
      return s[start:end]
  except ValueError:
      return ""

def hamming_distance(chaine1, chaine2):
  return sum(c1 != c2 for c1, c2 in zip(chaine1, chaine2))
```

```
def formatandCalculate(m1,m2):
      m1e = m1.encode('utf-8')
      chaine1 = binascii.hexlify(m1e).decode('utf-8')
      chaine2 = m2
      chaine1bin = binary(chaine1)
      chaine2bin = binary(chaine2)
      return hamming_distance(chaine1bin,chaine2bin)

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('chals20.cybercastors.com', 14431))
client.recv(1024)
client.send("\n".encode())


while True:
  data = client.recv(1024).decode("utf-8")
  if "Enter hamming distance:" in data:
      # print(client.recv(1024))
      transmited_message = find_str_between_strs(data,"Transmitted
message: ",  "\nReceived message: ")
      recieved_message = find_str_between_strs(data,"Received
message: ",  "\nEnter hamming distance:")
      print('TRANSMITED_MESSAGE: ',transmited_message)
      print('REICEIVED_MESSAGE: ',recieved_message)
      hammingDistance = str(formatandCalculate(transmited_message,
recieved_message)).encode('utf-8')
      print('HAMMING_DISTANCE: ', hammingDistance)
      client.send(hammingDistance)
      print(client.recv(1024))
      print()
  elif "The machine is currently" in data:
      print("Well FUCK")
      transmited_message = find_str_between_strs(data,"Transmitted
message: ",  "\nReceived message: ")
      recieved_message = find_str_between_strs(data,"Received
message: ",  "\nEnter hamming distance:")
      print('TRANSMITED_MESSAGE: ',transmited_message)
```

```
    print('REICEIVED_MESSAGE: ',recieved_message)
    hammingDistance = str(formatandCalculate(transmited_message,
recieved_message)).encode('utf-8')
    print('HAMMING_DISTANCE: ', hammingDistance)
    client.send(hammingDistance)
    print(client.recv(1024))
    print()
```

flagGodzzz : castorsCTF{c0mmun1ng_w17h_7h3_f14g_g0d5}

Glitchity glitch

Through my previous experience connecting to servers with netcat through python, I was able to whip up a solution quite quickly to this problem. However, I soon realized that using the built in socket library for these problems was not working, since it would keep resetting my connection after a certain amount of tries. With this realization, I switched over to using the pwntools library, which proved to be a much smoother process.

At first, I fiddled around with the server and found that if you bought a VPN and sold it back, the system would not deduct the VPN item from the user but still increase the user's money as if he or she had sold it. With this in mind, I devised a simple script that would do this until getting to the required $6000 needed to buy the flag.

```python
from pwn import *

conn = remote('chals20.cybercastors.com', 14432)
print(conn.recvline())
conn.recvuntil("Choice: ")
conn.sendline("6 \n")
conn.recvuntil("Choice: ")

for i in range(301):
    conn.sendline("0 \n")
    conn.recvuntil("Choice: ")
    conn.sendline("1 \n")
    try:
        print(conn.recvuntil("Choice: "))
    except:
        continue

conn.sendline("5 \n")
print(conn.recvline())
print(conn.recvline())
print(conn.recvline())
```

General:

Readme:

Readme

50

Author: hasu

I noticed something strange while reading the rules... Must be my imagination.

For this one, I simply looked through the source code of the Readme page, looking for "castorsCTF{". Through this, I found the flag:

```
<ul>
    <li>First  place: $256 <span style="color: white">castorsCTF{0u7_0f_5173_0u7_0f_m1nd}</span></li>
    <li>Second place: $128</li>
    <li>Third  place: $64</li>
</ul>
```

Welcome:

Welcome!

50

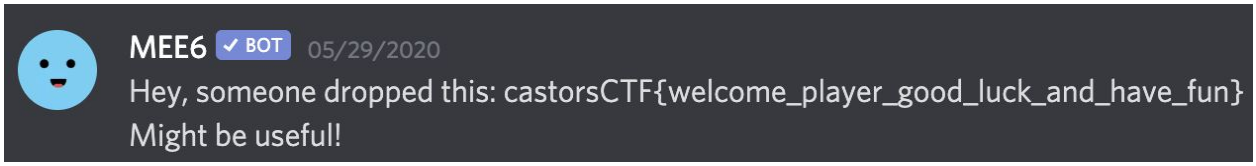
Author: hasu

Oh jeez! With all the rush I must've dropped the welcome !flag somewhere in the server. If only we had a bot we could command to pick it up.

Flag

Submit

Initially, I checked the site's robots.txt file to see if the flag was hidden there. When I saw that the side didn't have one, I decided to look at the discord and saw many people sending "!flag" through the off-topic channel. This command gave me the flag.



Web:

Bane Art:

Challenge 101 Solves ×

Bane Art

300

Author: icinta

http://web1.cybercastors.com:14438

Flag

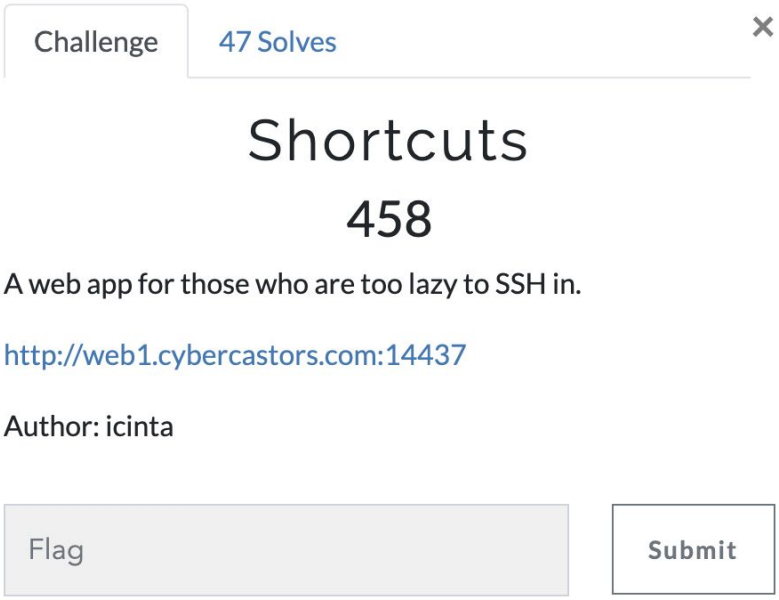
Submit

The approach to this problem was fairly straightforward. When opening the page, I noticed that the page was vulnerable to an LFI attack through the topic parameter, as so:
http://web1.cybercastors.com:14438/app.php?topic=games.php

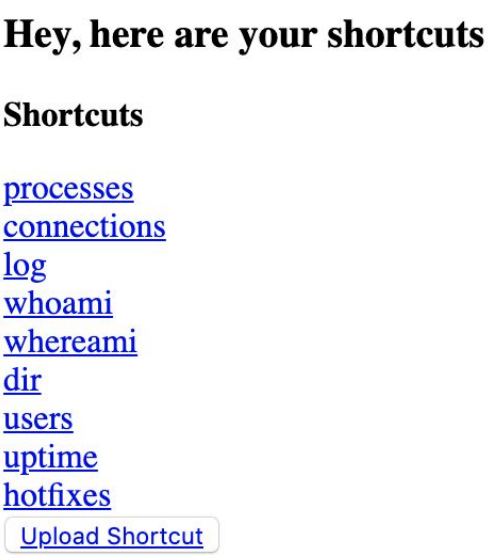
Through changing the topic, I was able to access various files on the system. Through this I got access to the Apache 2 log, as so:
http://web1.cybercastors.com:14438/app.php?topic=/var/log/apache2/access.log

It is there where we saw a get request for the flag location, which was:
http://web1.cybercastors.com:14438/app.php?topic=/home/falg/flag/test/why/the/hassle/right/flag.txt

Shortcuts:



Upon entering the webpage, the first thing I thought of doing was checking the links on the page. Upon closer inspection, I saw a link that leads to a list of shortcuts that seem to execute system commands when accessed:



The upload shortcuts button was of particular interest to me because it seemed to be accepting files uploaded by users. At first, I tried doing some simple commands injection through the filenames to no avail. Upon closer inspection, I noticed that the server would try to execute Go code with the name of the life we uploaded by adding the extension “.go” to the end. Through trial and error, I devised a small script that would run “ls” on the server and uploaded it twice (one as filename, and one as filename.go. This second one would be run when the first filename was pressed). After running ls through the root directory, then home, and then tom, I was able to device the final script, which would get the flag file through a running the cat command in Go.

```go
package main

import (
    "bytes"
    "fmt"
    "os/exec"
)

func main() {
    cmd := exec.Command("cat", "/home/tom/flag.txt")
    var out bytes.Buffer
    var stderr bytes.Buffer
    cmd.Stdout = &out
    cmd.Stderr = &stderr
    err := cmd.Run()
    if err != nil {
        fmt.Println(fmt.Sprint(err) + ": " + stderr.String())
        return
}
    fmt.Printf("%q\n", out.String())
}

```

Mixed Feelings

Challenge 26 Solves

Mixed Feelings

488

Author: icinta

We tried to tell Jeff that one doesn't go with the other but he didn't listen. Can you please pwn him and reveal his dirty secrets? Also for some reason they told us he likes XXXTentacion.

http://web1.cybercastors.com:14439/

Flag Submit

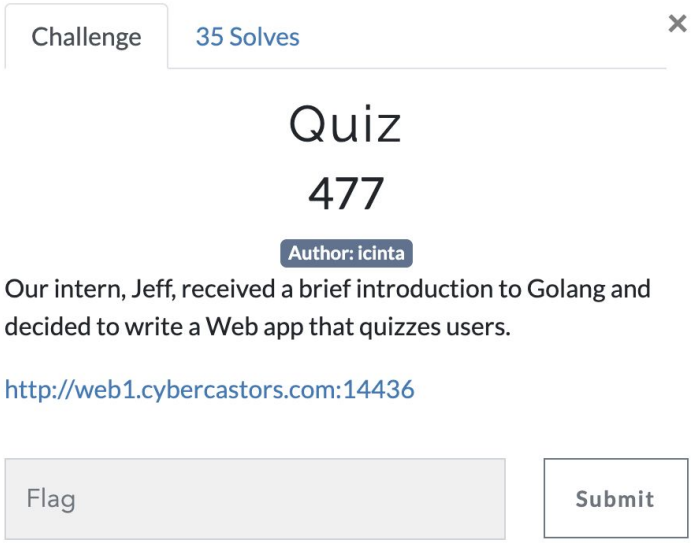
At first glance, I looked at this problem and was stumped on how to solve it. To try to take a stab at it, I decided to view the source of the page to see if any interesting comments were left behind. When I did this, I found a comment mentioning a directory that was not linked to anywhere.

```
<?php
if(isset($file)) {
    if ($user == falling_down_a_rabit_hole) {
        exit()?
    }
    else {
        go to .flagkindsir
    }
    ?>
```

When I accessed the .flagkindsir directory of the domain, I was greeted with 2 buttons that would each send a POST request to the server. I tried fiddling around with the content of the parameters for a while and noticed that if I sent anything other than the default parameters that were set on the page, I would get a blank response. On a hunch, I messaged one of the admins which told me to try sending a parameter with “something I wanted”. This led me to send the following POST request, which returned a page with the flag.

```
curl -X POST -d "cookies=flag" web1.cybercastors.com:14439/.flagkindsir
```

Quiz



Initially, I tried fiddling around with the quiz portion of the site the challenge linked to, thinking there was a way to execute my own Go code on the server. However, after a few attempts, I realized this seemed to be the wrong approach. On a hunch, I looked at the discord, in which one of the admins suggested that people try “dirb” with this challenge. When I did, I found a hidden file:

http://web1.cybercastors.com:14436/backup/

Upon closer inspection, this file seemed to be the source code of the code that was analyzing the responses to the quiz. It also seemed to contain an interesting bit of code that would allow one to access local directories on the server:

```
func super(w http.ResponseWriter, req *http.Request, ps httprouter.Params) {
        fmt.Println(ps.ByName("whynot"))
        var file string = "/" + ps.ByName("directory") + "/" + ps.ByName("theme") + "/" + ps.ByName("whynot")
        test, err := os.Open(file)
        handleError(w, err)
        defer test.Close()

        scanner := bufio.NewScanner(test)
        var content string
        for scanner.Scan() {
                content = scanner.Text()
        }

        fmt.Fprintf(w, "Directories: %s/%s\n", ps.ByName("directory"), ps.ByName("theme"))
        fmt.Fprintf(w, "File: %s\n", ps.ByName("whynot"))
        fmt.Fprintf(w, "Contents: %s\n", content)
}
```

Through trial and error, reading through the Go documentation, and looking through various system files, I was able to find the flag by following the challenge description,

which mentioned the user “Jeff” as being the one who created the program. Through this, I was able to access the flag:

http://web1.cybercastors.com:14436/test/home/jeff/flag.txt

Pitfall

Pitfall

456

Author: hasu

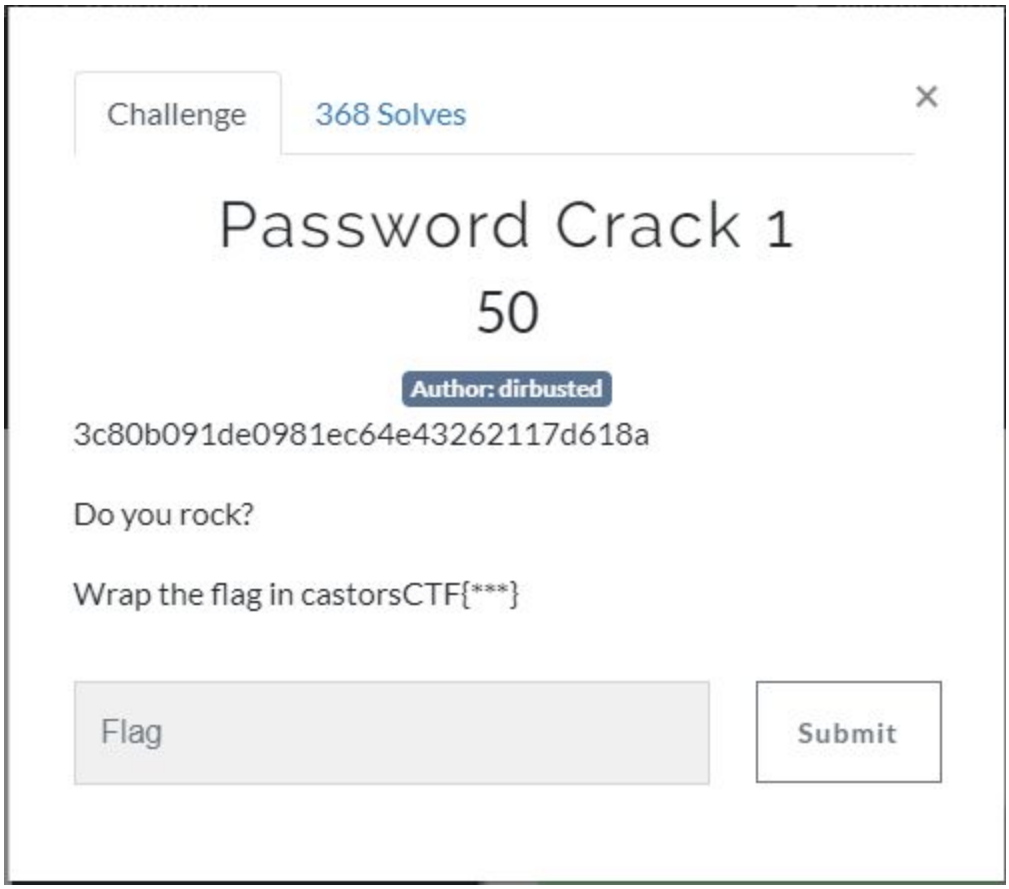
sylv3on_ was visiting cybercastors island and thought it'd be funny to bury the flag.txt. Can you help us DIG it out?

Initially, I thought the problem would include the dig command, but couldn’t figure out what the included image had to do with that command. Since I could not find the relation, I tried running various steganography tools on the image, which proved to be unfruitful since the image was in png format and a lot of the tools available required the image to be in another format (notably, steghide). After realizing this, I tried running the dig command on the cybercastors server, which proved to be correct when I submitted the dig txt command and received the flag.

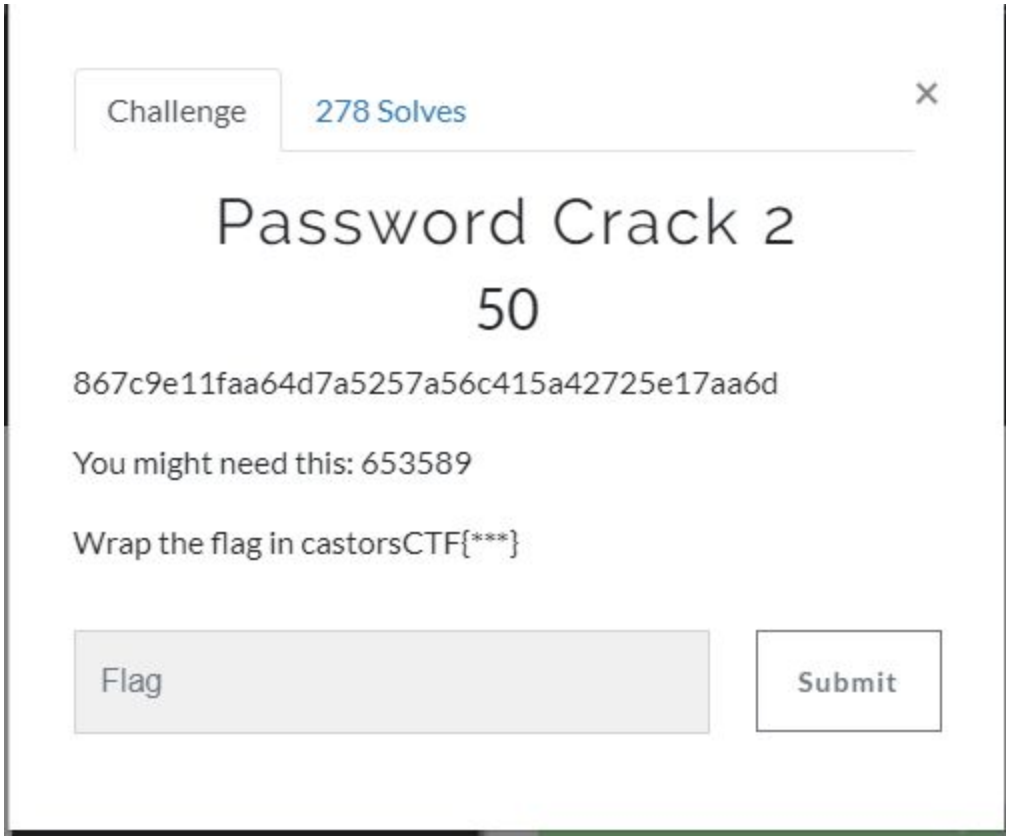
```
; <<>> DiG 9.10.6 <<>> cybercastors.com TXT
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37374
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1452
;; QUESTION SECTION:
;cybercastors.com.              IN      TXT

;; ANSWER SECTION:
cybercastors.com.       300     IN      TXT     "v=spf1 include:_spf.google.com ~all"
cybercastors.com.       300     IN      TXT     "flag=castorsCTF{L00K_1_DuG_uP_4_fL4g_464C4147}"

;; Query time: 94 msec
;; SERVER: 10.0.0.1#53(10.0.0.1)
;; WHEN: Sun May 31 21:55:32 AST 2020
;; MSG SIZE  rcvd: 168
```

Misc

Password Crack 1



Password Crack 2

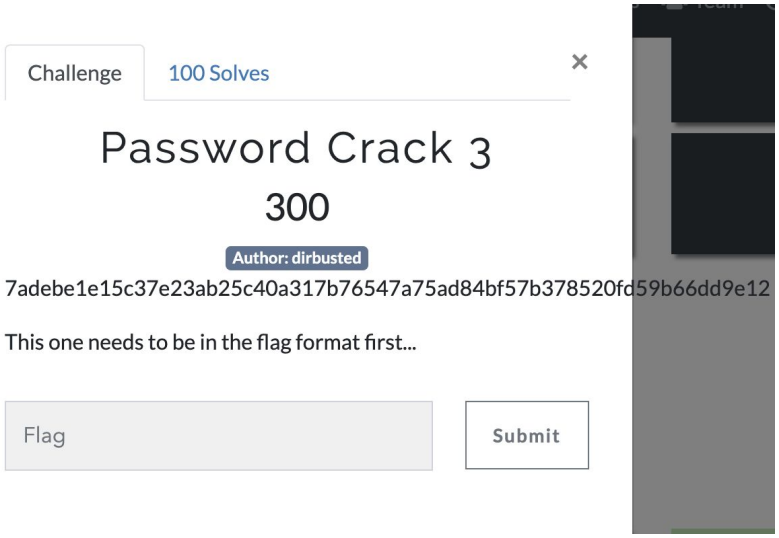


For both the Password Cracking 1 and Password Cracking 2 initially we ran a analysis on the hashes and resulted in an MD5 hash and then did a reverse lookup using the HashToolkit

Password Cracking 1: `irocktoo`

Password Cracking 2: `pi3141592653589`

Password crack 3:



This challenge was tricky. I knew the hash was SHA-256, but did not understand what it meant when it mentioned having to be in flag format. Through some digging, I was able to devise a rule for hashcat to be able to append and prepend characters to strings from a wordlist.

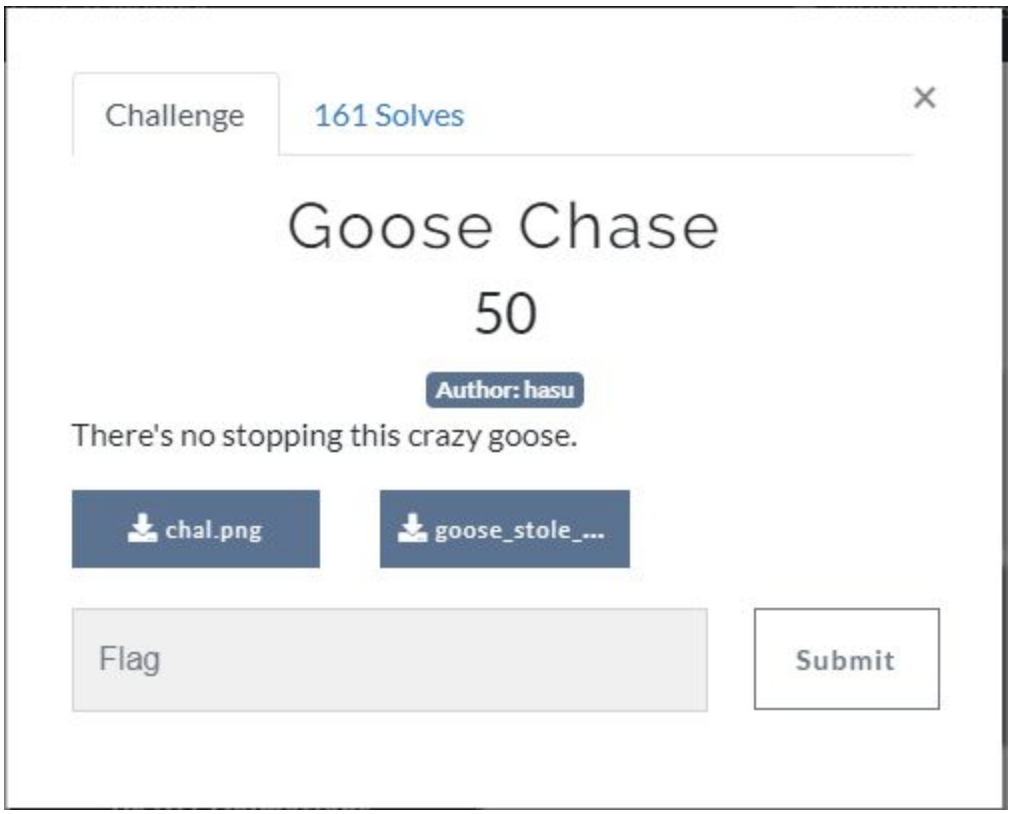
^{^F^T^C^s^r^o^t^s^a^c $}

With this rule set, I ran it with the rockyou.txt wordlist, which give me the flag:

castorsCTF{theformat!}

Crypto

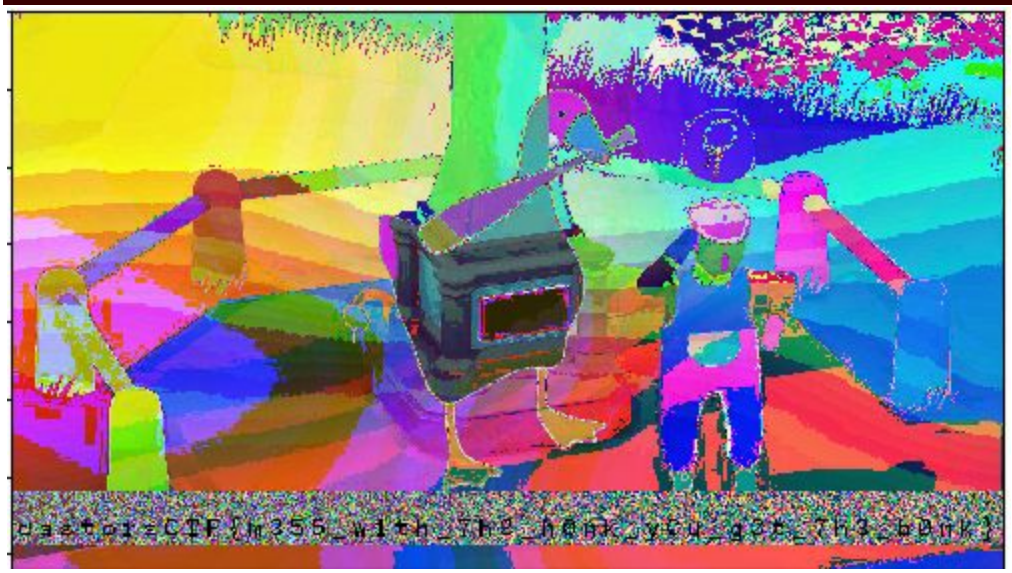
Goose Chase



Initially, when I saw this problem I noticed that one of the images had negative colors so immediately I thought it was a pixel addition or subtraction problem so I made a python script to add each pixel of the images and resulted on the key

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```
goo = cv2.imread("chal.png")
key = cv2.imread("goose_stole_the_key.png")

res = key + goo
imgplot = plt.imshow(res)
plt.show()
```



Then I took a picture of the screen with my cellphone to reduce the noise and got the flag

[Magic School Bus]

Challenge 83 Solves

Magic School Bus

366

Author: hasu

nc chals20.cybercastors.com 14421

Flag is slightly out of format. Add underscores and submit uppercase

Flag

Submit

When looking at this problem we immediately noticed that every time we input a string, the system scrambled it and returned it. But we could not make out a clear pattern. After some fooling around we noticed there was a pattern, every string of a set length was scrambled the same way, the positions that switched were always the same. So we thought if we could figure out the positions switched on a string with the same length as the string for the school bus, we would be able to decrypt it. So we went waaaay outside the box on this one, all the way to excel. In order to test the approach we created this script in js:

```
// Testing character displacements

// Original text
let og =
["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t
","u","v","w","x","y","z","1","2","3","4","5","6","7","8","9","0","á","é","í","
ó","ú","ä","ë","ï","ö","ü","ć","ý"]

// Scrambled text returned
let scramble =
["G","O","W","5","Í","Ć","D","L","T","2","0","Ï","A","I","Q","Y","7","Ú","C","K
","S","1","9","Ë","B","J","R","Z","8","Ä","H","P","X","6","Ó","Ý","E","M","U","
3","Á","Ö","F","N","V","4","É","Ü"]

//Turn OG into upper case
og = og.map(item=>{
   return item.toUpperCase()
 })

// Findout how the positions switched in the array
result = og.map(item => {
   return scramble.findIndex(position => position==item)
 })

// Format result for excel
res1 = res1.map(item=>{
   return `=B${item}`
 })
```

Given an array of 48 items (same as the string form the challenge) and the returned encrypted string, we determined the position switch and formatted them into something we could use and we got the following result:

```
res1= [
```

```
  '=B13', '=B25', '=B19', '=B7',  '=B37',
  '=B43', '=B1',  '=B31', '=B14', '=B26',
  '=B20', '=B8',  '=B38', '=B44', '=B2',
  '=B32', '=B15', '=B27', '=B21', '=B9',
  '=B39', '=B45', '=B3',  '=B33', '=B16',
  '=B28', '=B22', '=B10', '=B40', '=B46',
  '=B4',  '=B34', '=B17', '=B29', '=B23',
  '=B11', '=B41', '=B47', '=B5',  '=B35',
  '=B18', '=B30', '=B24', '=B12', '=B42',
  '=B48', '=B6',  '=B36'
]
```

With this we created a CSV containing the values in res1 and the values of "S,C,N,T,G,E,T,0,S,K,V,3,C,T,N,E,S,Y,S,2,I,S,L,7,A,F,4,I,0,S,C,0,C,O,M,5,O,R,S,3,1,R,R,3,A,Y,N,1", and this yielded the result:

C	S
A	C
S	N
T	T
O	G
R	E
S	T
C	0
T	S
F	K
2	V
0	3
R	C
3	T
C	N
0	E
N	S
4	Y
I	S
S	2
S	I
A	S

N	L
C	7
E	A
I	F
S	4
K	I
3	0
Y	S
T	C
O	0
S	C
0	O
L	M
V	5
1	O
N	R
G	S
M	3
Y	1
S	R
7	R
3	3
R	A
1	Y
E	N
5	1

Thus, getting the result:

CASTORSCTF20R3C0N4ISSANCEISK3YTOS0LV1NGMYS73R1E5