HACKvent 2019 - Writeup

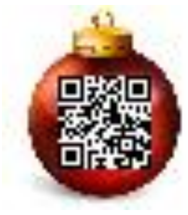
HV19.01 censored

I got this little image, but it looks like the best part got censored on the way. Even the tiny preview icon looks clearer than this! Maybe they missed something that would let you restore the original content?

Solution

Exiftool shows that the image has a large thumbnail. Let's see if this is also censored. We can use mig hty ImageMagick (https://imagemagick.org/) to extract the thumbnail:

```
convert challenge.jpg thumbnail:thumb.jpg
```



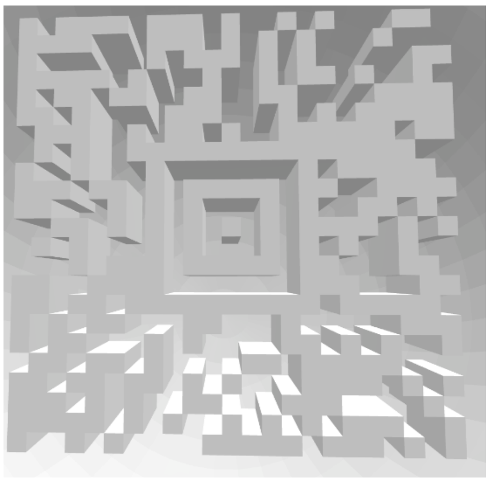
Flag: HV19{just-4-PREview!}

HV19.02 Triangulation

Today we give away decorations for your Christmas tree. But be careful and do not break it.

Solution

The zip contains a 3D plan (.stl file) of a christmas ball. Using any STL viewer (e.g. FreeCAD), we can open the model. Once, we zoom into the inner part of it, we can see a QR code:



Unfortunately, due to the poor contrast and the 3D-effects, my QR code scanner was not able to read the code, so I decided to quickly recreate the pattern in photoshop (I know, manual work sucks, but it works :)).

Flag: HV19{Cr4ck_Th3_B411!}

HV19.03 Hodor, Hodor, Hodor

```
$HODOR: hhodor. Hodor. Hodor!?  = `hodor?!? HODOR!? hodor? Hodor oHodor. hodor? ,
HODOR!?! ohodor!?  dhodor? hodor odhodor? d HodorHodor  Hodor!? HODOR HODOR?
hodor! hodor!? HODOR hodor! hodor? !

hodor?!? Hodor  Hodor Hodor? Hodor  HODOR  rhodor? HODOR Hodor!?  h4Hodor?!?
Hodor?!? 0r hhodor?  Hodor!? oHodor?! hodor? Hodor  Hodor! HODOR Hodor hodor? 64
HODOR Hodor  HODOR!? hodor? Hodor!? Hodor!? .

HODOR?!? hodor- hodorHoOodoOor Hodor?!? OHoOodoOorHooodorrHODOR hodor. oHODOR...
Dhodor- hodor?! HooodorrHODOR HoOodoOorHooodorrHODOR RoHODOR... HODOR!?! 1hodor?!
HODOR... DHODOR- HODOR!?! HooodorrHODOR Hodor- HODORHoOodoOor HODOR!?! HODOR...
DHODORHoOodoOor hodor. Hodor! HoOodoOorHodor HODORHoOodoOor 0Hooodorrhodor
HoOodoOorHooodorrHODOR 0=`;
hodor.hod(hhodor. Hodor. Hodor!? );
```

Solution

This challenge was labelled as fun-programming, so my first guess was that the obscure string could be a program in some sort of esoteric programming language. A quick google search reveals that there is a Hodor programming language. The website links to a GitHub-Repo, which shows that Hodor is just some JavaScript translation. Fortunately, there are already some online Tools, which allow you to run your Hodor code online. I pasted the code and got the following output:

```
Awesome, you decoded Hodors language!

As sis a real h4xx0r he loves base64 as well.

SFYxOXtoMDFkLXRoMy1kMDByLTQyMDQtbGQ0WX0=
```

So we only need to base64-decode the output to get the flag.

Flag: HV19{h01d-th3-d00r-4204-ld4Y}

Bonus

If we look at the code in the Hodor-Repository, we can see that during execution the whole program is translated back to JavaScript and executed using eval. This way, we can easily recover the original JS code:

```
var html = `Awesome, you decoded Hodors language!

As sis a real h4xx0r he loves base64 as well.

SFYxOXtoMDFkLXRoMy1kMDByLTQyMDQtbGQ0WX0=`;
console.log(html);
```

HV19.04 password policy circumvention

Santa released a new password policy (more than 40 characters, upper, lower, digit, special).

The elves can't remember such long passwords, so they found a way to continue to use their old (bad) password:

```
merry christmas geeks
```

Solution

The zip file contains an *.ahk file. A quick google search reveals, that this file-extension is widely used for AutoHotkey, which offers a simple Hotkey-based scripting language for Windows. If we take a look at the code, we see that the phrases of the additional sentence in the challenge description is defined as hotkeys. Therefore, I quickly installed the software, loaded the script and typed these words into a text editor. AutoHotkey applied the logic from the script, and the sentence transformed into the flag (the password that matches the policy).

Flag: HV19{R3memb3r, rem3mber - the 24th 0f December}

HV19.05 Santa Parcel Tracking

To handle the huge load of parcels Santa introduced this year a parcel tracking system. He didn't like the black and white barcode, so he invented a more solemn barcode. Unfortunately the common barcode readers can't read it anymore, it only works with the pimped models santa owns. Can you read the barcode

SP-Tracking: 1337-9999-4555-9

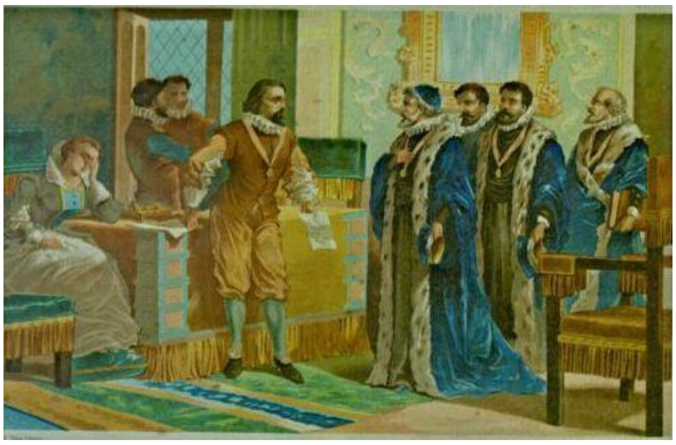
Solution

Scanning the barcode results in "Not the solution", so I assume the flag is stored into the colors. Manual analysis did not help, so I wrote a small C# program, which extracts all the colors from the image, piped them into a file (separated per channel) and took a closer look. As I know that the flag format for HACKvent is HV19{}, I started to look for the related ASCII codes (72, 86, 49, 57). Apparently, this sequence is present in the blue-channel of the barcode (from byte 13 - 48). Finally, I adapted my program to extract the flag:

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;

namespace HV1905
{
    class Program
    {
        static void Main(string[] args)
        {
            var colors = new List<Color>();
            var image = new Bitmap(@"./157de28f-2190-4c6d-a1dc-02ce9e385b5c.png");

            for (var y = 0; y < image.Height; y++)
            {
                for (var x = 0; x < image.Width; x++)
                {
                    var color = image.GetPixel(x, y);
                    if (!colors.Contains(color))
                    {
                        colors.Add(color);
                    }
                }
            }

            var lines = colors.Select(c => {
                return (char)c.B;
            }).ToArray();

            var flag = new string(lines).Substring(13, 35);

            Console.WriteLine(flag);

            Console.ReadLine();
        }

    }
}
```

Flag: HV19{D1fficult_to_g3t_a_SPT_R3ader}

HV19.06 bacon and eggs



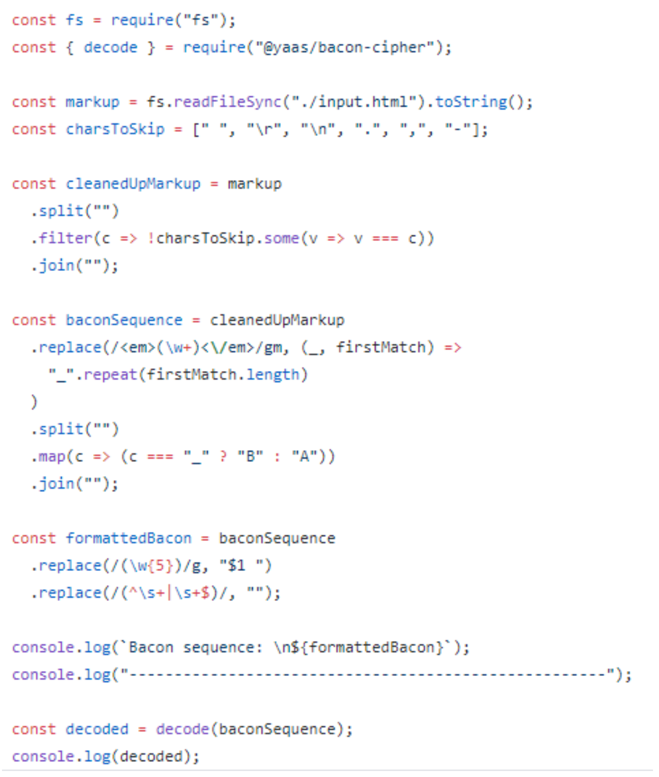
Francis Bacon was an English philosopher and statesman who served as Attorney General and as Lord Chancellor of England. His works are credited with developing the scientific method and remained influential through the scientific revolution. Bacon has been called the father of empiricism. His works argued for the possibility of scientific knowledge based only upon inductive reasoning and careful observation of events in nature. Most importantly, he argued science could be achieved by use of a sceptical and methodical approach whereby scientists aim to avoid misleading themselves. Although his practical ideas about such a method, the Baconian method, did not have a long-lasting influence, the general idea of the importance and possibility of a sceptical methodology makes Bacon the father of the scientific method. This method was a new rhetorical and theoretical framework for science, the practical details of which are still central in debates about science and methodology.

Bacon was the first recipient of the Queen's counsel designation, which was conferred in 1597 when Elizabeth I of England reserved Bacon as her legal advisor. After the accession of James VI and I in 1603, Bacon was knighted. He was later created Baron Verulam in 1618 and Viscount St. Alban in 1621. Because he had no heirs, both titles became extinct upon his death in 1626, at 65 years. Bacon died of pneumonia, with one account by John Aubrey stating that he had contracted the condition while studying the effects of freezing on the preservation of meat. He is buried at St Michael's Church, St Albans, Hertfordshire.

```
Born: January 22
Died: April 9
Mother: Lady Anne
Father: Sir Nicholas
Secrets: unknown
```

Solution

The text is a typeface-based Bacon Cipher. The trick is to convert the individual characters to A's and B's, depending on their typeface. I wrote a small JavaScript program to process the text and decode the Bacon Cipher



```javascript
const fs = require("fs");
const { decode } = require("@yaas/bacon-cipher");

const markup = fs.readFileSync("./input.html").toString();
const charsToSkip = [" ", "\r", "\n", ".", ",", "-"];

const cleanedUpMarkup = markup
  .split("")
  .filter(c => !charsToSkip.some(v => v === c))
  .join("");

const baconSequence = cleanedUpMarkup
  .replace(/<em>(\w+)<\/em>/gm, (_, firstMatch) =>
    "_".repeat(firstMatch.length)
  )
  .split("")
  .map(c => (c === "_" ? "B" : "A"))
  .join("");

const formattedBacon = baconSequence
  .replace(/(\w{5})/g, "$1 ")
  .replace(/(^\s+|\s+$)/, "");

console.log(`Bacon sequence: \n${formattedBacon}`);
console.log("----------------------------------------------------");

const decoded = decode(baconSequence);
console.log(decoded);
```

The output of my program looks like this:

```
Santalikeshisbaconbutalsothisbaconthepasswordishvxbaconcipherissimplebutcoolxrepla
cexwithbracketsanduseuppercaseforallcharacteraaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
a
```

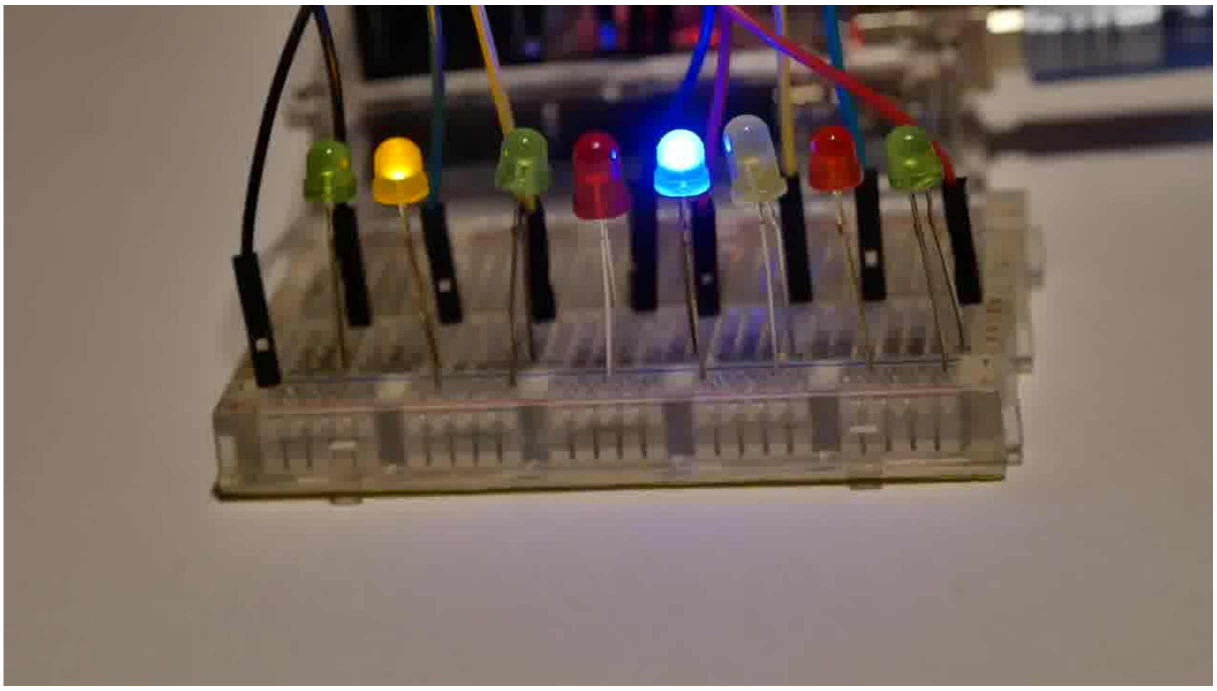
Flag: HV19{BACONCIPHERISSIMPLEBUTCOOL}

HV19.07 Santa Rider

Santa is prototyping a new gadget for his sledge. Unfortunately it still has some glitches, but look for yourself.

Solution

When looking at the middle of the video, we can see that the LEDs start to wildly blink instead of continuing the pattern from the beginning/end. Also, there are exactly 8 LEDs, which could represent eight bits. I stopped the video at the beginning (see photo) and realized that those are indeed the bits of an H.



Using the following `ffmpeg' command, I extracted all video frames and manually converted the blinking LEDs to bits:

```
ffmpeg -i .\3DULK2N7DcpXFg8qGo9Z9qEQqvaEDpUCBB1v.mp4 image-%d.jpeg
```

After a lot of swearing I ended up with the following bit sequence, which can be converted to the flag:

```
01001000 01010110 00110001 00111001 01111011 00110001 01101101 01011111 01100001
01101100 01110011 00110000 01011111 01110111 00110000 01110010 01101011 00110001
01101110 01100111 01011111 00110000 01101110 01011111 01100001 01011111 01110010
00110011 01101101 00110000 01110100 00110011 01011111 01100011 00110000 01101110
01110100 01110010 00110000 01101100 01111101
```

Flag: HV19{1m_als0_w0rk1ng_0n_a_r3m0t3_c0ntr0l}

HV19.08 SmileNcryptor 4.0

You hacked into the system of very-secure-shopping.com and you found a SQL-Dump with $$-creditcards numbers. As a good hacker you inform the company from which you got the dump. The managers tell you that they don't worry, because the data is encrypted.

Goal

Analyze the "Encryption"-method and try to decrypt the flag.

Solution

When looking at the encrypted credit card numbers, we can see that the characters are close to each other. As also the input characters (0-9) are close, we suspect that this might be a transposition cipher. However, the longer the number gets, the bigger is the offset between the expected source range (0-9) and the target range (Q-b). Therefore, my guess was that the offset increases depending on the position.

I created a small python script to brute force the possible offset values accross all credit card numbers, assuming that there must be one offset that leads to valid credit card numbers. For each position in the encrypted data, I increased the offset by one. Running my program revealed that the only possible offset in this scenario is 30. When using this value as an initial offset, I was able to decrypt the content of the flag-table:

```python
#!/usr/bin/python3
import sys

encrypted_flag = r"SlQRUPXWVo\Vuv_n_\ajjce"
encrypted_ccards = [r"QVXSZUVY\ZYYZ[a", r"QOUW[VT^VY]bZ", r"SPPVSSYVV\YY_\\]",
r"RPQRSTUVWXYZ[\]^", r"QTVWRSVUXW[_Z`\b]"]

def decrypt(input, initialShift):
    shift = initialShift
    output = ""

    for c in input:
        target = ord(c) - shift
        try:
            output += chr(target)
        except:
            break

        shift += 1

    return output

for encrypted_card in encrypted_ccards:
    decrypted = decrypt(encrypted_card, 30)
    if len(decrypted) == len(encrypted_card):
        print(decrypted)

print("Flag: HV19{%s}" % (decrypt(encrypted_flag, 30)))
```

Flag: HV19{5M113-420H4-KK3A1-19801}