Tasteless CTF 2019 - RGB (steg)

The CTF in retrospective

I've checked out some of the challenges, but in the end I've sticked to the RGB Challenge, because I did already quite some steganography challenges. I really liked that they had a good variaty of challenges and some of them were quite unique.

I would rank this CTF in the skill requirement to the level **high**.
It is also reflected in the scoreboard on CTF Time, where were not a lot of teams that managed to get past 639 Points.

Scoreboard Tasteless CTF 2019 on CTFtime

In the End I also managed to grab one of the 3 Flags.

Tasteless released the Files of this CTF on their Github repo:
Tasteless CTF 2019 Repo

Overview - RGB

We got presented with 3 Challenges called `R`, `G` and `B`. All of them linked to the same pcapng file. So in one file were 3 Flags hidden. This was a steganography challenge. We needed to find the hidden flags in the network dump.

You can find the pcapng file in the following github link: chall.pcapng

R challenge

Investigating the PCAP

The PCAP file exists of a single stream of a HTTP Communication:

The endpoint the page `ctf.tasteless.eu/stegano` is requested by the caller.

```
GET /stegano HTTP/1.1
Host: ctf.tasteless.eu
User-Agent: curl/7.55.1
Accept: */*
```

and get responded by a PNG File

```
HTTP/1.1 200 OK
Content-Type: image/png
Accept-Ranges: bytes
Transfer-Encoding: chunked
Date: Sat, 19 Oct 2019 13:37:00 GMT
Server: lighttpd

<PNG binary Data>
```

After extracting the picture, I've checked it out with stegsolve. Stegsolve is a cool Steganography tool, written in Java by Caesum.
Caesum made a great Handbook about Steganography where he also links to his selfwritten tools Stegsolve

In the picture we found some hints for all 3 Challenges in the corresponding Plane

- Red plane 0:

```
2616
Category: Standards Track
== HTTP1.1
```

- Green plane 0:

```
2083
Category: Informational
== PNG
```

- Blue plane 0:

```
1951
Category: Informational
== DEFLATE
```

Finding the flag

While checking the PCAP File, I saw that the responses are chunked. When I investigated the TCP Stream I checked the chunked responses, that were introduced with the string `1000;` and I also saw that they have a single character afterwards attached in the line. While stepping through I realized that it was the flag.

Extracting the flag

I didn't take the time to write a python script with `scapy` to extract the packets, instead I extracted the flag by hand with help of almighty Wireshark.

I've opened the `Follow -> TCP Stream` Option and put into the search Field the value `1000;`. Then I clicked through and assembled the Flag by hand.

Flag: `tctf{NoB0dy_3xPec7s_chUnK_ex7En5iOnz}`

G challenge

This Challenge was not solved by me but by the colleague `@dachleitner` in Mattermost.

It was hidden in the CRC fields of the IDAT chunks.
CRC Fields are also sometimes a usable sidechannel to hide information :-)

Flag: `tctf{Wh0_Do3s_n0t_l1k3_fL4gz_1n_cRc}`

B challenge

Finding the flag

We didn't managed to find it during the CTF.

Challenge Aftermath

After the CTF I asked in the IRC channel what was the intended solution for the B challenge. I got the following anser:
`hetti91: extract bits from byte padding in front of stored blocks`

It was linked to following Function of zlib:
zlib BYTEBITS()

Lessons learned

- Dig further
- When doing CTFs, write down the values of the flag format characters in hex, so you can find them in a hexdump or stream of data.