UNIKL CTF 2020

FINAL REPORT

By **G0oD1Uck** (Adani & Najihah)

Table of Contents

This were a question that successfully answered by us.

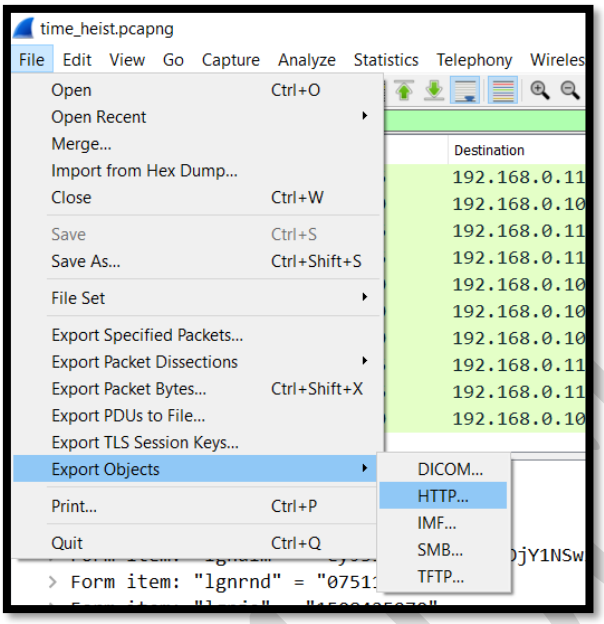
FORENSIC

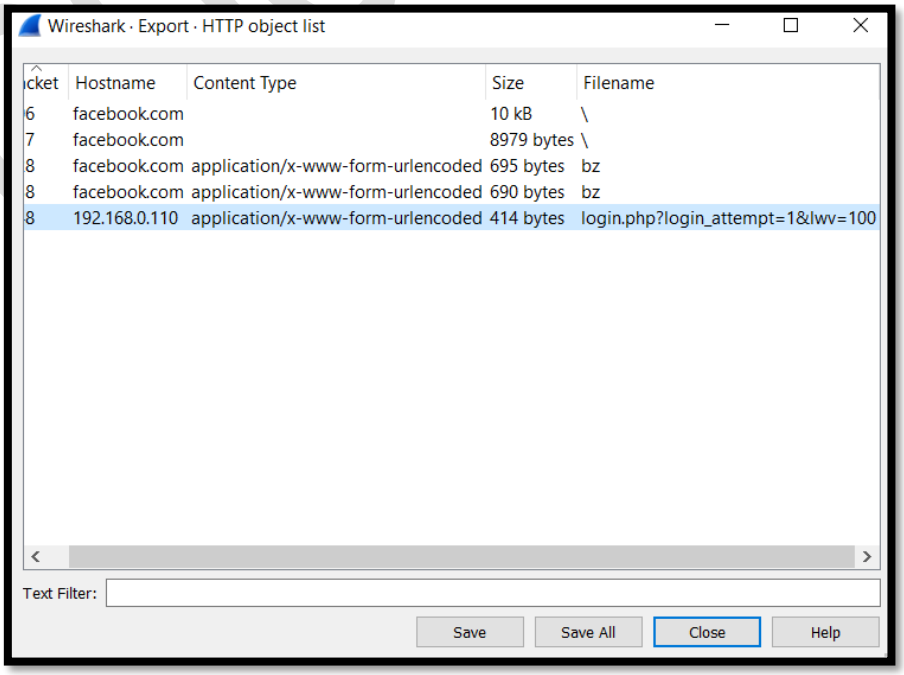
1. **Easxnalysis (20)**

Open the downloaded file which is easxnalysis,zip. Unzip the file, we get time_heist.pcapng and flag,zip.

Yeahh, the given flag.zip is password protected. So, I think we need to find the password to unzip from given pcap file.

So, let's open up the time_heist.pcapng file. The first thing to try when dealing with pcap file is, You go to: **FILE → Export Objects → HTTP**

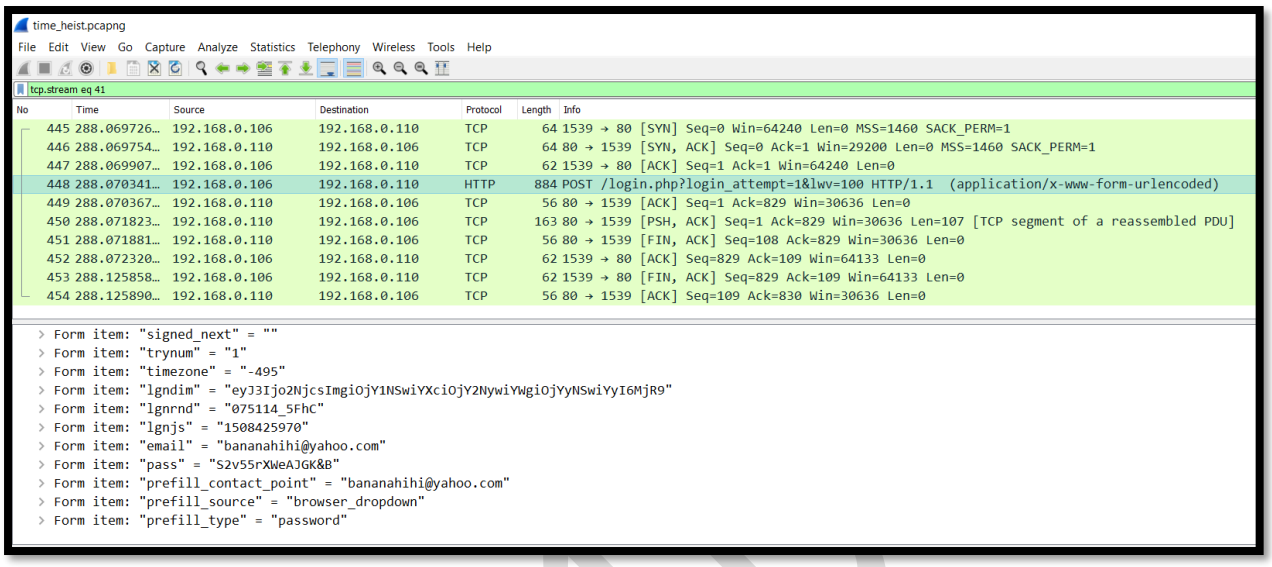




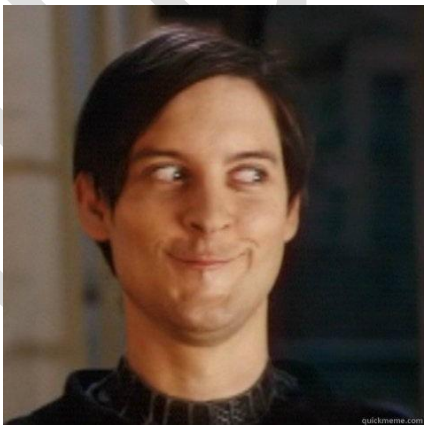
You may export/save to your computer from here. Some CTF, you may find flag here.

Ok, so back to our challenge. The last row that I have highlight seems interesting. It is a login page. We may get username or password here.

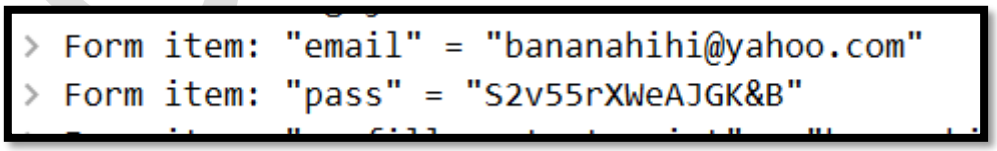
So, click that line will bring you to the picture below. So, now, just simply analyse.



And …



We found the password. Password: S2v55rXWeAJGK&B

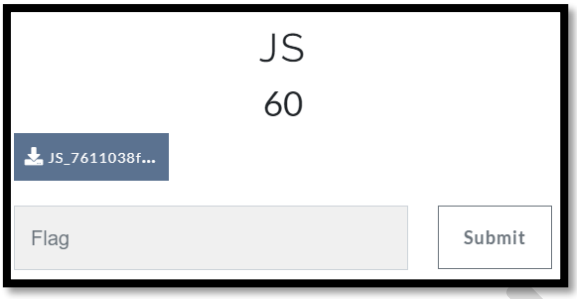


Try unzip flag.zip with this password successfully give us flag.txt.

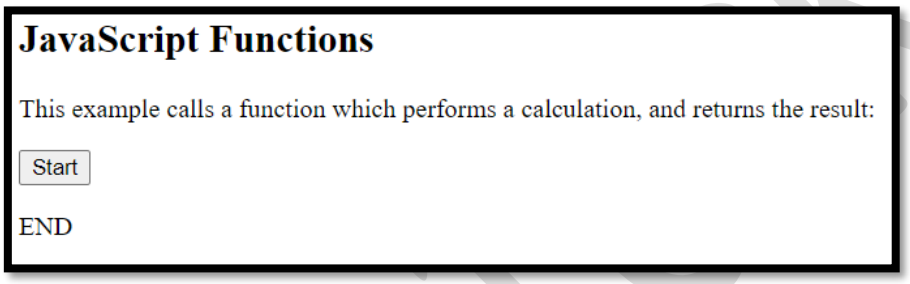
Flag: unikl ctf20{7h15_15_7o0_345y_f0r_y0u}

WEB

1. JS (60)



Download file JS_7611038f48adf601c6d5799a2a61eec6.html. I click START button.



It shocked me. Upon click START button, it is print the flag in order but one by one letter very-very fast. In a glance. So, for this challenge I found two ways to solve it.

FIRST

I change the time. So that it be very slow. I change to 2000. So, we can jot down the flag.

```
else  {
//frame = atob(array[c]);
document.getElementById("demo").innerHTML = framejs[c];
c = c + 1;
t = setTimeout(function1, 2000);
  }

            }
```

SECOND

Open the downloaded html with any code viewer (I use Sublime Text). Upon opening it, you will see something fishy and it caught my interest.

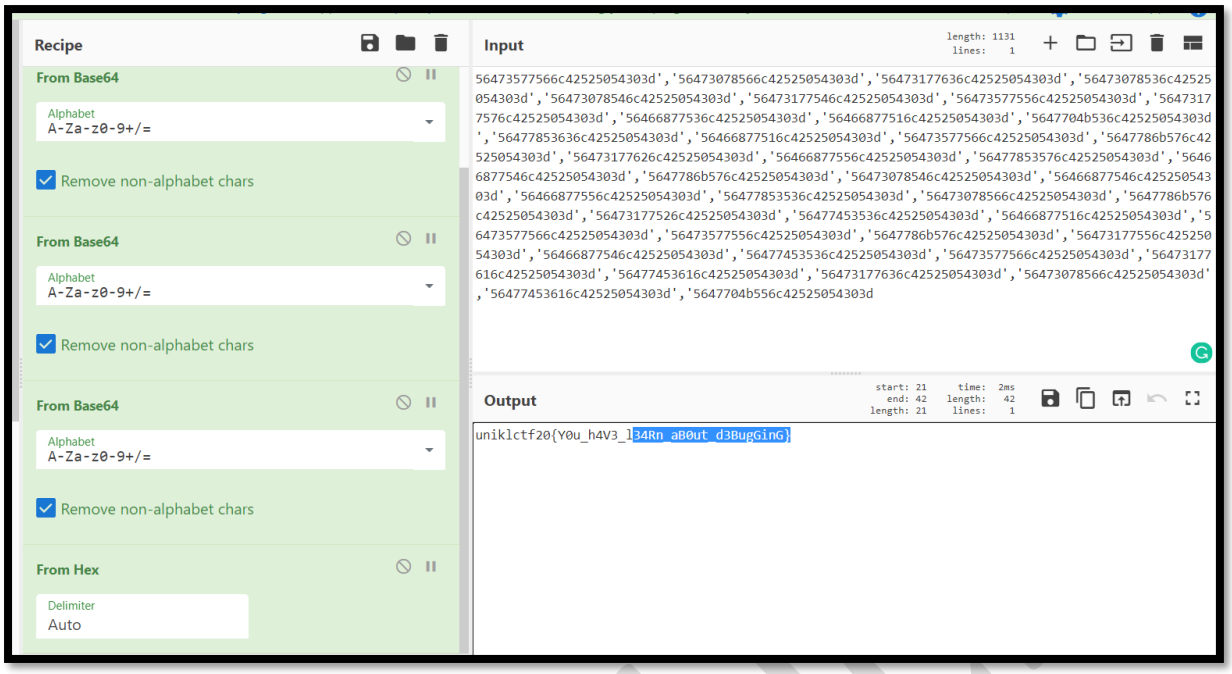
```
JS_7611038f48adf601c6d5799a2a61eec6.html
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Functions</h2>
<p>This example calls a function which performs a calculation, and returns the result:</p>
<button onclick="function2()">Start</button>
<p id="demo"></p>
<script>
array = ['56473577566c42525054303d','56473078566c42525054303d','56473177636c42525054303d','56473078536c42525054303d','56473078546c42525054303d','
    56473177546c42525054303d','56473577556c42525054303d','56473177576c42525054303d','56466877536c42525054303d','56466877516c42525054303d','
    5647704b536c42525054303d','56477853636c42525054303d','56466877516c42525054303d','56473577566c42525054303d','5647786b576c42525054303d','
    56473177626c42525054303d','56466877556c42525054303d','56477853576c42525054303d','56466877546c42525054303d','5647786b576c42525054303d','
    56473078546c42525054303d','56466877546c42525054303d','56466877556c42525054303d','56477853536c42525054303d','56473078566c42525054303d','
    5647786b576c42525054303d','56473177526c42525054303d','56477453536c42525054303d','56466877516c42525054303d','56473577566c42525054303d','
    56473577556c42525054303d','5647786b576c42525054303d','56473177556c42525054303d','56466877546c42525054303d','56477453536c42525054303d','
    56473577566c42525054303d','56473177616c42525054303d','56477453616c42525054303d','56473177636c42525054303d','56473078566c42525054303d','
    56477453616c42525054303d','5647704b556c42525054303d'];

var c = 0;
var t;
var timer_is_on = 0;
var frame;
var framejs = [];
```

So, I copy all the number and fire up my favourite “oven”.

I am using Cyber Chef (https://gchq.github.io/CyberChef/).

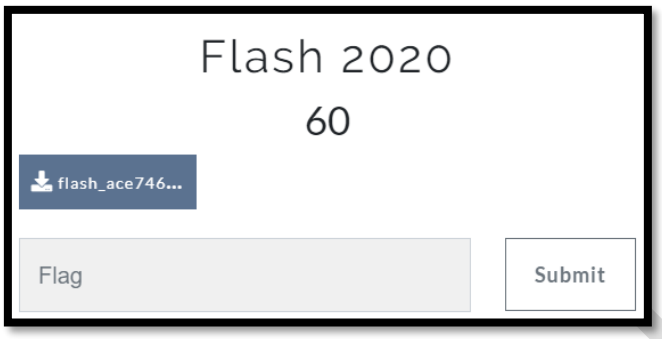
Hex → Base64 → Bas64 → Base64 → Hex



Flag: uniklctf20{Y0u_h4V3_l34Rn_aB0ut_d3BugGinG}

MOBILE

1. Flash (60)

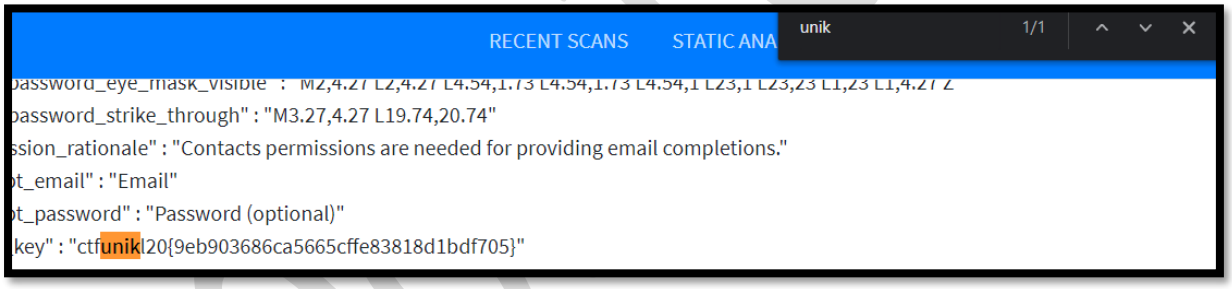


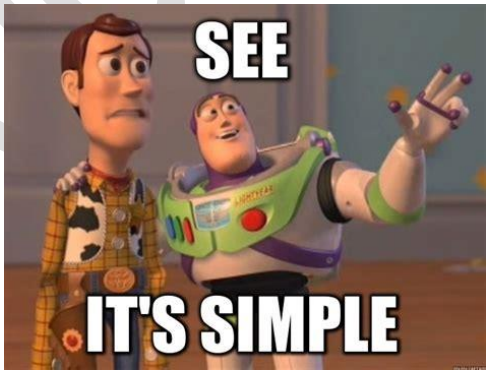
Download the flash.apk.

This challenge I will use MobSF for reverse engineer the apk.

Then, you run scan on the apk.

After result is out, just **CTRL + F** and find our flag keyword "unikl"





Flag: ctfunikl20{9eb903686ca5665cffe83818d1bdf705}

2. Superman (40)

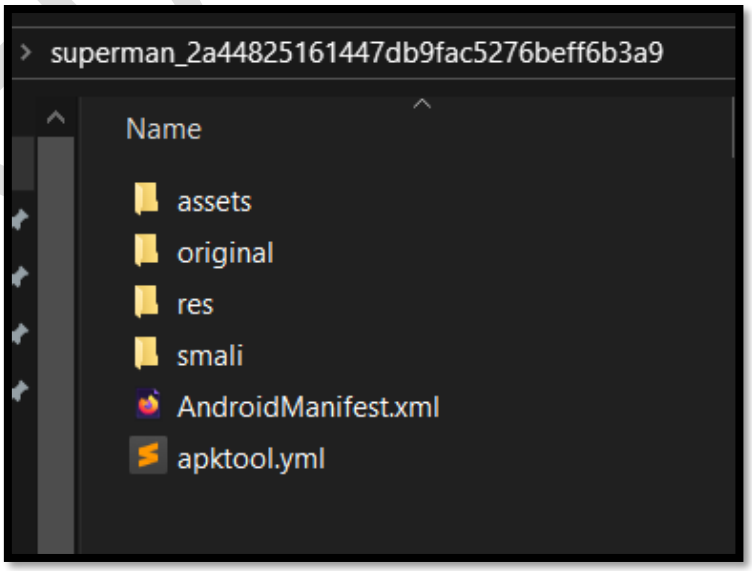
Download the superman.apk

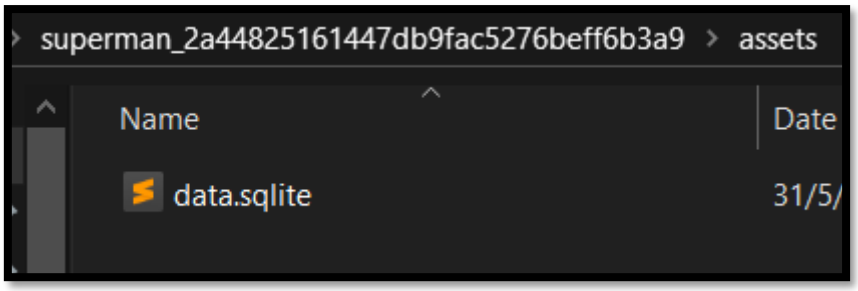
First, I use MobSF static analyser. But, it lead me nowhere. So, I decide to use apktool.

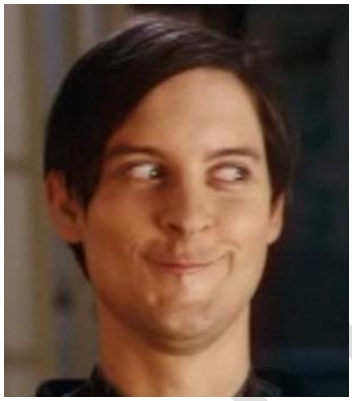
Command: apktool superman_2a44825161447db9fac5276beff6b3a9.apk

```
C:\Users\HP\Desktop>apktool superman_2a44825161447db9fac5276beff6b3a9.apk
I: Using Apktool 2.4.1 on superman_2a44825161447db9fac5276beff6b3a9.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\HP\AppData\Local\apktool\framework\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

After finished, I open the folder. And this were inside it. Assets folder caught my eye.

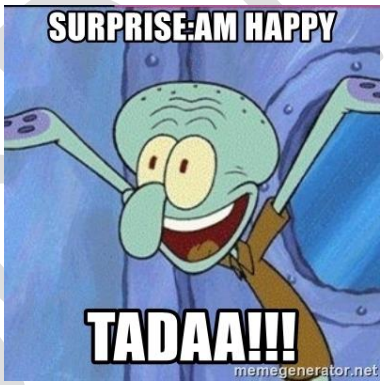






Ouh, there is a sqlite file. So, quick search said, we can view it use online tools.

https://inloop.github.io/sqlite-viewer/



KEY (1 rows)

```
SELECT * FROM 'KEY' LIMIT 0,30
```

ID	PTLABKEY
1	uniklctf20{b1fcaf6a4f7063b4d86e36c18b972c65}

Flag: uniklctf20{b1fcaf6a4f7063b4d86e36c18b972c65}

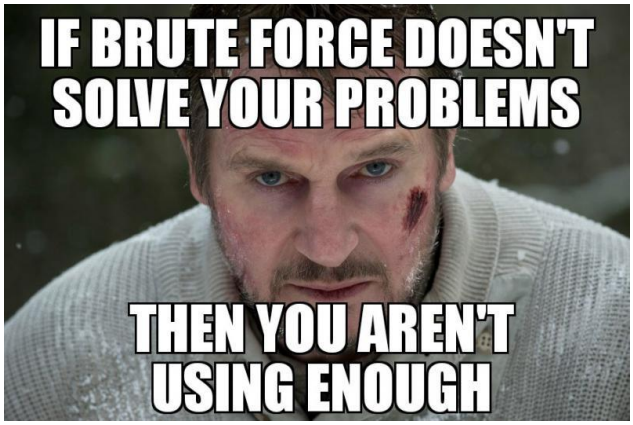
Cryptography

1. XOR not Hard (20)

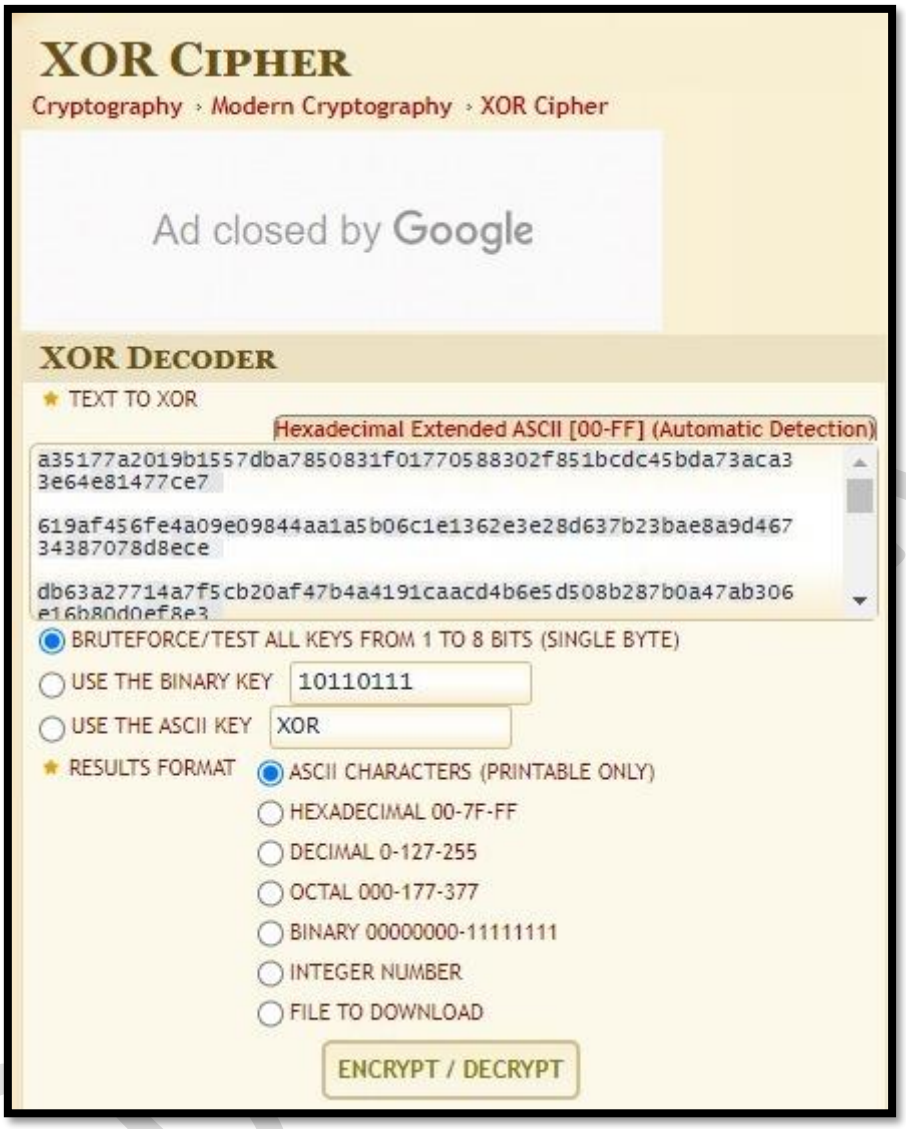
Download the file give us this:


```
a35177a2019b1557dba7850831f01770588302f851bcdc45bda73aca33e64e81477ce7
619af456fe4a09e09844aa1a5b06c1e1362e3e28d637b23bae8a9d46734387078d8ece
db63a27714a7f5cb20af47b4a4191caacd4b6e5d508b287b0a47ab306e16b80d0ef8e3
b1a63eb6efe821d3b17df6e2e05a4cc0e76cc7d83d873a2acb3e15ee681c34815741f7
bfeb509fad978044f4a7a379d09a95007a143fcba8f8065b2d0f3eabe6e6dd059a6918
f6189dbd1fb58763cb9489943b9bbb2535c545ff11128f439d928aff0c2a575027c983
a40db4b6f1872e8e01decb96db03947f529e72420d04d92082ed9e4a389520e96c3bb4
afb5250db2deacecb9887a5eb154158c47eead0e72d08bd3801a26fc225d76d5990c64
5b98479da694dc077c62e1fc1f82042bed78e56249eaf14047a7ab44cdb5b9ba098d9e
db3d474a6b5b2f3c57279237d7da6cf13ea962b6fa607604b3ea03572f6190f5e62cfc
0d8690f15e0d629565e84239300c4cd53bd6dec6d18680cde540f53fb1b550e7d777bd
6a37a582f3961c637d4d0f1daa345450f94ffb46df775398fbf075af2af5e843acdd50
5ba9c01e7f362bc8d85d34e034e003414448eaf0e8ab4b0398d5a7c94937d4311ca4fb
```

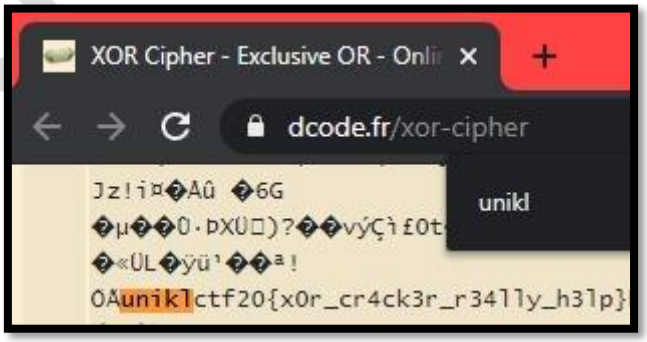
So, the challenge said XOR. But… We do not have the key. It's Okay.



We decide to use this online tool. https://www.dcode.fr/xor-cipher. Let's do XOR Brute force.



CTRL +F our keyword "unikl"



Flag: uniklctf20{x0r_cr4ck3r_r34lly_h3lp}