Matreshka

Download the file and extract it. We got: code2.class and data.bin

Try to decompile it using this web: www.javadecompilers.com

We will get .java so we can open it and read the code

```
public static void main(String[] paramArrayOfString) throws Exception {
  String str = "matreha!";
  byte[] arrayOfByte1 = encode(System.getProperty("user.name").getBytes(), str);
  byte[] arrayOfByte2 = { 76, -99, 37, 75, -68, 10, -52, 10, -5, 9, 92, 1, 99, -94, 105, -18 };
  for (int i = 0; i < arrayOfByte2.length; i++) {
    if (arrayOfByte2[i] != arrayOfByte1[i]) {
      System.out.println("No");
      return;
    }
  }
```

The Program will use encode function to encode the login name. then compare with arrayOfByte2. if you use right login name, it pass and decode the data.bin file for us. If wrong, it will exit.

The encode using DES for encrypt so we just decode it and got the right username: “ lettreha”. The program decode the file for us, we got stage2.bin is an elf file so I run it on my VM.

I run it, it only return the string “Fail”. But I cant see it in string list.

So I do some reverse in main function and find out some function name rc4. Maybe the program use rc4 to decrypt the “Fail” and print it out later.

```
.text:0000000000475FCF                mov     [rsp+0C0h+var_8], rbp
.text:0000000000475FD7                lea     rbp, [rsp+0C0h+var_8]
.text:0000000000475FDF                mov     rax, cs:main_statictmp_0
.text:0000000000475FE6                mov     [rsp+0C0h+var_51], rax
.text:0000000000475FEB                mov     rax, cs:main_statictmp_1
.text:0000000000475FF2                mov     qword ptr [rsp+0C0h+var_49], rax
.text:0000000000475FF7                movups  xmm0, xmmword ptr cs:main_statictmp_1+1
.text:0000000000475FFE                movups  [rsp+0C0h+var_49+1], xmm0
.text:0000000000476003                lea     rax, [rsp+0C0h+var_51]
.text:0000000000476008                mov     [rsp+0C0h+var_C0], rax
.text:000000000047600C                mov     [rsp+0C0h+var_B8], 8
.text:0000000000476015                mov     [rsp+0C0h+var_B0], 8
.text:000000000047601E                call    crypto_rc4_NewCipher
.text:0000000000476023                mov     rax, [rsp+0C0h+var_A8]
.text:0000000000476028                mov     [rsp+0C0h+var_18], rax
.text:0000000000476030                mov     rcx, cs:os_executablePath
.text:0000000000476037                mov     rdx, cs:qword_52DE48
.text:000000000047603E                mov     [rsp+0C0h+var_C0], rcx
.text:0000000000476042                mov     [rsp+0C0h+var_B8], rdx
.text:0000000000476047                call    path_filepath_Dir
.text:000000000047604C                mov     rax, [rsp+0C0h+var_A8]
.text:0000000000476051                mov     rcx, [rsp+0C0h+var_B0]
.text:0000000000476056                mov     [rsp+0C0h+var_C0], rcx
.text:000000000047605A                mov     [rsp+0C0h+var_B8], rax
```

Ok let do some debug

```
.text:0000000000475FC2 jbe     loc_47631D
```

Break this, modify CF to 0. Then continue

```
.text:00000000004762EA call    runtime_printlock
.text:00000000004762EF lea     rax, aFailGreekkhmer          ; "Fail\nGreekKhmerLatinLimbuNushuOgh
.text:00000000004762F6 mov     [rsp+0C0h+var_C0], rax
.text:00000000004762FA mov     [rsp+0C0h+var_B8], 5
.text:0000000000476303 call    runtime_printstring
.text:0000000000476308 call    runtime_printunlock
```

Ok it will print out “Fail” so we need to avoid it, jump back

```
.text:0000000000476126 jnz     loc_4762EA
```

Break this, restart debug and modify ZF at last breakpoint.

Next we see one more check, it will jump to

```
.text:0000000000476181 call    runtime_printlock
.text:0000000000476186 lea     rax, aFailGreekkhmer+2B2Ch      ; "OK, decoding payload..
```

Seem good, we restart and modify it.

We have 1 new file: result.pyc

```
result.pyc  stage2.bin
```

Thanks batman for his awesome tip, we decompile the pyc file to get source

```
def decode(data, key):
    idx = 0
    res = []
    for c in data:
        res.append(chr(c ^ ord(key[idx])))
        idx = (idx + 1) % len(key)

    return res


flag = [
 40, 11, 82, 58, 93, 82, 64, 76, 6, 70, 100, 26, 7, 4, 123, 124, 127, 45, 1, 125, 107, 115, 0, 2, 31, 15]
print('Enter key to get flag:')
key = input()
if len(key) != 8:
    print('Invalid len')
    quit()
res = decode(flag, key)
print(''.join(res))
```

Last stage is easy

```
def decode(data, key):
    idx = 0
    res = []
    for c in data:
        res.append(chr(c ^ ord(key[idx])))
        idx = (idx + 1) % len(key)

    return res

def make_key(array):
    key = ""
    prefix = "cybrics{"
    for i in range(0,len(prefix)):
        key += chr(ord(prefix[i]) ^ array[i])
    return key

flag = [40, 11, 82, 58, 93, 82, 64, 76, 6, 70, 100, 26, 7, 4, 123, 124, 127, 45, 1, 125, 107, 115, 0, 2, 31, 15]
key = make_key(flag)
print "key: %s" % key
if len(key) != 8:
    print('Invalid len')
    quit()
res = decode(flag, key)
print(''.join(res))
```