wdb2020 Go

题目描述

题目名称

wdb2002——Go

题目难度

★★

题目分值

200

考察知识点

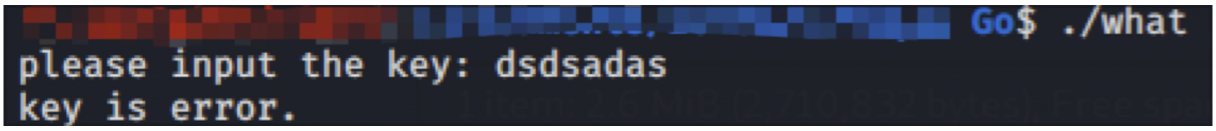
二进制分析 算法还原 换表base

解题步骤

第一步

题目信息：

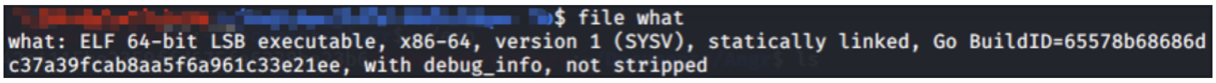
运行程序 观察基本特征：



```
[redacted] Go$ ./what
please input the key: dsdsadas
key is error.
```

输入一些值， 发现flag错误

使用file命令查看程序的基本信息

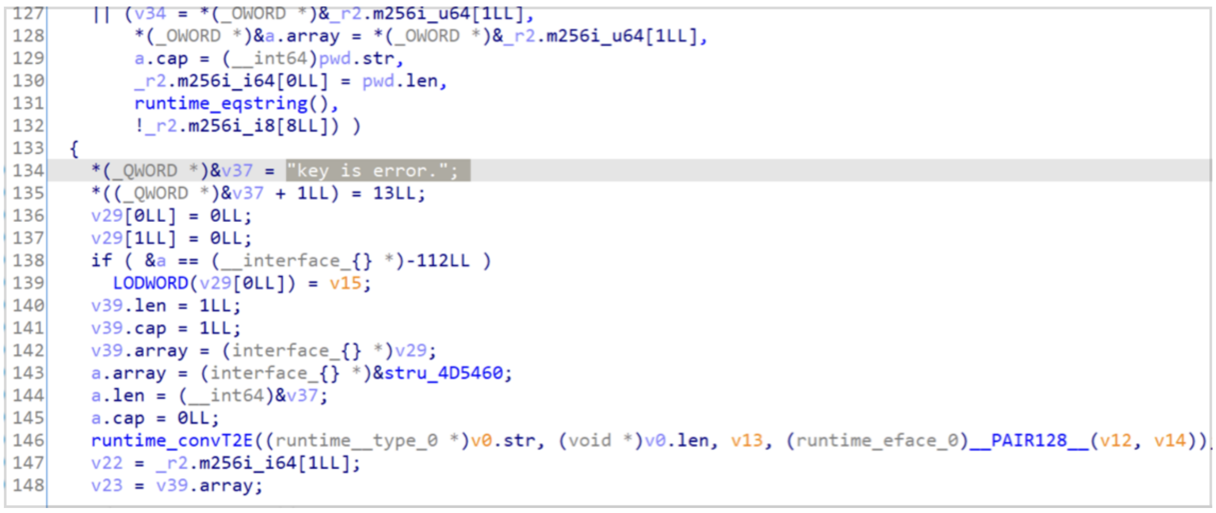


```
[redacted]$ file what
what: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, Go BuildID=65578b68686d
c37a39fcab8aa5f6a961c33e21ee, with debug_info, not stripped
```

64位程序，使用IDA载入

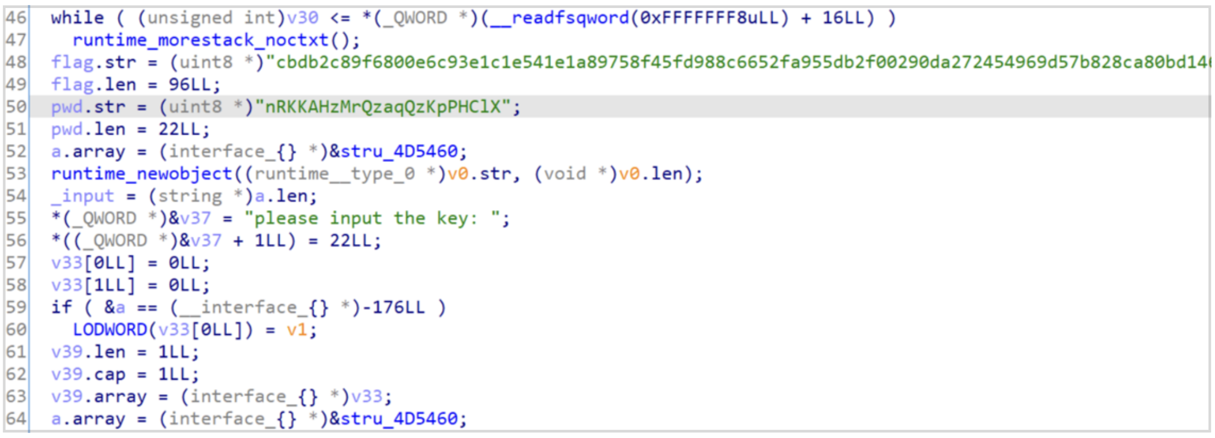
使用7.0反编译，无法获得伪C代码，尝试使用IDA 7.5版本载入

成功反编译



```
    || (v34 = *(_OWORD *)&_r2.m256i_u64[1LL],
        *(_OWORD *)&a.array = *(_OWORD *)&_r2.m256i_u64[1LL],
        a.cap = (__int64)pwd.str,
        _r2.m256i_i64[0LL] = pwd.len,
        runtime_eqstring(),
        !_r2.m256i_i8[8LL]) )
  {
    *(_QWORD *)&v37 = "key is error.";
    *((_QWORD *)&v37 + 1LL) = 13LL;
    v29[0LL] = 0LL;
    v29[1LL] = 0LL;
    if ( &a == (__interface_{} *)-112LL )
      LODWORD(v29[0LL]) = v15;
    v39.len = 1LL;
    v39.cap = 1LL;
    v39.array = (interface_{} *)v29;
    a.array = (interface_{} *)&stru_4D5460;
    a.len = (__int64)&v37;
    a.cap = 0LL;
    runtime_convT2E((runtime__type_0 *)v0.str, (void *)v0.len, v13, (runtime_eface_0)__PAIR128__(v12, v14))
    v22 = _r2.m256i_i64[1LL];
    v23 = v39.array;
```

key is error部分伪代码如下



```
while ( (unsigned int)v30 <= *(_QWORD *)(__readfsqword(0xFFFFFFF8uLL) + 16LL) )
  runtime_morestack_noctxt();
flag.str = (uint8 *)"cbdb2c89f6800e6c93e1c1e541e1a89758f45fd988c6652fa955db2f00290da272454969d57b828ca80bd14
flag.len = 96LL;
pwd.str = (uint8 *)"nRKKAHzMrQzaqQzKpPHClX";
pwd.len = 22LL;
a.array = (interface_{} *)&stru_4D5460;
runtime_newobject((runtime__type_0 *)v0.str, (void *)v0.len);
_input = (string *)a.len;
*(_QWORD *)&v37 = "please input the key: ";
*((_QWORD *)&v37 + 1LL) = 22LL;
v33[0LL] = 0LL;
v33[1LL] = 0LL;
if ( &a == (__interface_{} *)-176LL )
  LODWORD(v33[0LL]) = v1;
v39.len = 1LL;
v39.cap = 1LL;
v39.array = (interface_{} *)v33;
a.array = (interface_{} *)&stru_4D5460;
```

第二步

解题过程

程序相关的main函数完整代码如下：

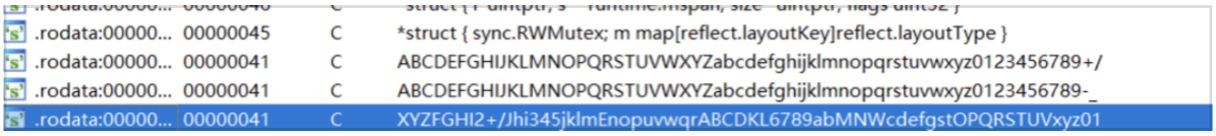
```
void __cdecl main_main()
{
  string v0; // rdi
  __int32 v1; // eax
  void *v2; // rdx
  unsigned int v3; // rcx
  unsigned int v4; // r8
  unsigned int v5; // rdx
  __int64 v6; // rax
  interface_{} *v7; // rbx
  interface_{} *v8; // rdx
  string v9; // rdx
  string v10; // rdx
  string v11; // r8
  unsigned int v12; // r8
  void *v13; // rdx
  __int64 v14; // rcx
  __int32 v15; // eax
  __int64 v16; // rdx
  __int32 v17; // eax
  void *v18; // rdx
  unsigned int v19; // rcx
  unsigned int v20; // r8
  unsigned int v21; // rdx
  __int64 v22; // rax
  interface_{} *v23; // rbx
  __interface_{} a; // [rsp+0h] [rbp-168h] BYREF
  __m256i _r2; // [rsp+18h] [rbp-150h]
  string *_input; // [rsp+48h] [rbp-120h]
  string pwd; // [rsp+50h] [rbp-118h]
  string flag; // [rsp+60h] [rbp-108h]
  int v29[2]; // [rsp+70h] [rbp-F8h] BYREF
  int v30[2]; // [rsp+80h] [rbp-E8h] BYREF
  __interface_{} err; // [rsp+90h] [rbp-D8h] BYREF
  string *v32; // [rsp+A8h] [rbp-C0h]
  int v33[2]; // [rsp+B0h] [rbp-B8h] BYREF
  __int128 v34; // [rsp+C0h] [rbp-A8h]
  int v35; // [rsp+D0h] [rbp-98h]
  int v36; // [rsp+D8h] [rbp-90h]
  __int128 v37; // [rsp+E0h] [rbp-88h] BYREF
  __uint8 dst; // [rsp+F0h] [rbp-78h]
  __interface_{} v39; // [rsp+108h] [rbp-60h]
  __uint8 v40; // [rsp+120h] [rbp-48h]
  main_gogo_0 coder; // [rsp+138h] [rbp-30h] BYREF

  while ( (unsigned int)v30 <= *(_QWORD *)(__readfsqword(0xFFFFFFF8uLL) +
16LL) )
    runtime_morestack_noctxt();
  flag.str = (uint8 *)”
cbdb2c89f6800e6c93e1c1e541e1a89758f45fd988c6652fa955db2f00290da272454969d57b828
ca80bd146ebe8c89d”;
  flag.len = 96LL;
  pwd.str = (uint8 *)”nRKKAHzMrQzaqQzKpPHClX”;
  pwd.len = 22LL;
  a.array = (interface_{} *)&stru_4D5460;
  runtime_newobject((runtime__type_0 *)v0.str, (void *)v0.len);
  _input = (string *)a.len;
  *(_QWORD *)&v37 = “please input the key: “;
  *((_QWORD *)&v37 + 1LL) = 22LL;
  v33[0LL] = 0LL;
  v33[1LL] = 0LL;
  if ( &a == (__interface_{} *)-176LL )
    LODWORD(v33[0LL]) = v1;
  v39.len = 1LL;
  v39.cap = 1LL;
  v39.array = (interface_{} *)v33;
  a.array = (interface_{} *)&stru_4D5460;
  a.len = (__int64)&v37;
  a.cap = 0LL;
  runtime_convT2E((runtime__type_0 *)v0.str, (void *)v0.len, v2,
(runtime_eface_0)__PAIR128__(v4, v3));
  v6 = _r2.m256i_i64[1LL];
  v7 = v39.array;
  err.cap = _r2.m256i_i64[0LL];
  v39.array->_type = (runtime__type_0 *)_r2.m256i_i64[0LL];
  v32 = (string *)v6;
  if ( runtime_writeBarrier.enabled )
  {
    a.array = (interface_{} *)&v7->data;
    a.len = v6;
    runtime_writebarrierptr((uintptr *)v0.str, v0.len);
  }
  else
  {
    v7->data = (void *)v6;
  }
  fmt_Print(v39, (__int64)v0.str, (error_0)__PAIR128__(v5, v0.len));
  err.array = 0LL;
  err.len = 0LL;
  v8 = (interface_{} *)&err;
  if ( &a == (__interface_{} *)-144LL )
    LODWORD(err.array) = (_DWORD)_input;
  v39.len = 1LL;
  v39.cap = 1LL;
  v39.array = (interface_{} *)&err;
  err.cap = (__int64)&unk_4CBD20;
  err.array = (interface_{} *)&unk_4CBD20;
  v32 = _input;
  if ( runtime_writeBarrier.enabled )
  {
    a.array = (interface_{} *)&err.len;
    a.len = (__int64)_input;
    runtime_writebarrierptr((uintptr *)v0.str, v0.len);
    v8 = v39.array;
  }
  else
  {
    err.len = (__int64)_input;
  }
  a.array = v8;
  a.len = v39.len;
  a.cap = v39.cap;
  fmt_Scanln(a, (__int64)v0.str, (error_0)__PAIR128__((unsigned int)v8,
v0.len));
  v0.len = (__int64)_input;
  a.array = (interface_{} *)_input->str;
  a.len = _input->len;
  v9.len = a.len;
  main_encode(v0, v9);
  v10.len = a.cap;
  v35 = a.cap;
  a.array = (interface_{} *)a.cap;
  v36 = _r2.m256i_i64[0LL];
  a.len = _r2.m256i_i64[0LL];
  a.cap = (__int64)"==";
  _r2.m256i_i64[0LL] = 2LL;
  strings_TrimRight(v0, v10, v11);
  v13 = (void *)pwd.len;
  v14 = _r2.m256i_i64[1LL];
  v15 = _r2.m256i_i32[4LL];
  if ( _r2.m256i_i64[2LL] != pwd.len
    || (v34 = *(_OWORD *)&_r2.m256i_u64[1LL],
        *(_OWORD *)&a.array = *(_OWORD *)&_r2.m256i_u64[1LL],
        a.cap = (__int64)pwd.str,
        _r2.m256i_i64[0LL] = pwd.len,
        runtime_eqstring(),
        !_r2.m256i_i8[8LL]) )
  {
    *(_QWORD *)&v37 = "key is error.";
    *((_QWORD *)&v37 + 1LL) = 13LL;
    v29[0LL] = 0LL;
    v29[1LL] = 0LL;
    if ( &a == (__interface_{} *)-112LL )
      LODWORD(v29[0LL]) = v15;
    v39.len = 1LL;
    v39.cap = 1LL;
    v39.array = (interface_{} *)v29;
    a.array = (interface_{} *)&stru_4D5460;
    a.len = (__int64)&v37;
    a.cap = 0LL;
    runtime_convT2E((runtime__type_0 *)v0.str, (void *)v0.len, v13,
(runtime_eface_0)__PAIR128__(v12, v14));
    v22 = _r2.m256i_i64[1LL];
    v23 = v39.array;
    err.cap = _r2.m256i_i64[0LL];
    v39.array->_type = (runtime__type_0 *)_r2.m256i_i64[0LL];
    v32 = (string *)v22;
    if ( !runtime_writeBarrier.enabled )
    {
      v23->data = (void *)v22;
      goto LABEL_16;
    }
    goto LABEL_17;
  }
  coder.Mode.str = 0LL;
  coder.Mode.len = 0LL;
  coder.Key.str = 0LL;
  coder.Key.len = 0LL;
  coder.IV.str = 0LL;
  coder.IV.len = 0LL;
  v0.len = (__int64)_input;
  coder.Key = *_input;
  a.array = 0LL;
  *(string *)&a.len = flag;
  runtime_stringtoslicebyte(
    (uint8 (*)[32])v0.str,
    (string)__PAIR128__((unsigned int)v13, (unsigned int)_input),
    (__uint8)a);
  v16 = _r2.m256i_i64[0LL];
  a.array = (interface_{} *)&coder;
  v40 = *(__uint8 *)_r2.m256i_i8;
  *(_OWORD *)&a.len = *(_OWORD *)_r2.m256i_i8;
  _r2.m256i_i64[0LL] = _r2.m256i_i64[2LL];
  main___gogo__Decode((main_gogo_0 *)v0.str, (__uint8)a, *(__uint8
*)_r2.m256i_i8, (error_0)__PAIR128__(v16, v0.len));
  a.array = 0LL;
  dst = *(__uint8 *)&_r2.m256i_u64[1LL];
  *(_OWORD *)&a.len = *(_OWORD *)&_r2.m256i_u64[1LL];
  _r2.m256i_i64[0LL] = _r2.m256i_i64[3LL];
  runtime_slicebytetostring((uint8 (*)[32])v0.str, (__uint8)a,
(string)__PAIR128__(_r2.m256i_u64[1LL], v0.len));
  v37 = *(_OWORD *)&_r2.m256i_u64[1LL];
  v30[0LL] = 0LL;
  v30[1LL] = 0LL;
  if ( &a == (__interface_{} *)-128LL )
    LODWORD(v30[0LL]) = v17;
  v39.len = 1LL;
  v39.cap = 1LL;
  v39.array = (interface_{} *)v30;
  a.array = (interface_{} *)&stru_4D5460;
  a.len = (__int64)&v37;
  a.cap = 0LL;
  runtime_convT2E((runtime__type_0 *)v0.str, (void *)v0.len, v18,
(runtime_eface_0)__PAIR128__(v20, v19));
  v22 = _r2.m256i_i64[1LL];
  v23 = v39.array;
  err.cap = _r2.m256i_i64[0LL];
  v39.array->_type = (runtime__type_0 *)_r2.m256i_i64[0LL];
  v32 = (string *)v22;
  if ( runtime_writeBarrier.enabled )
  {
LABEL_17:
    a.array = (interface_{} *)&v23->data;
    a.len = v22;
    runtime_writebarrierptr((uintptr *)v0.str, v0.len);
    goto LABEL_16;
  }
  v23->data = (void *)v22;
LABEL_16:
  fmt_Println(v39, (__int64)v0.str, (error_0)__PAIR128__(v21, v0.len));
```

```
}
```

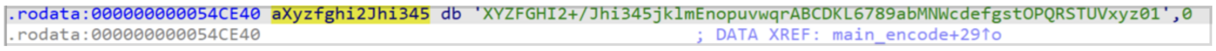
发现程序是一个类似于base64的加密

将题目中的字符串进行伪base64解密：

在数据段中也找到了关键字符串



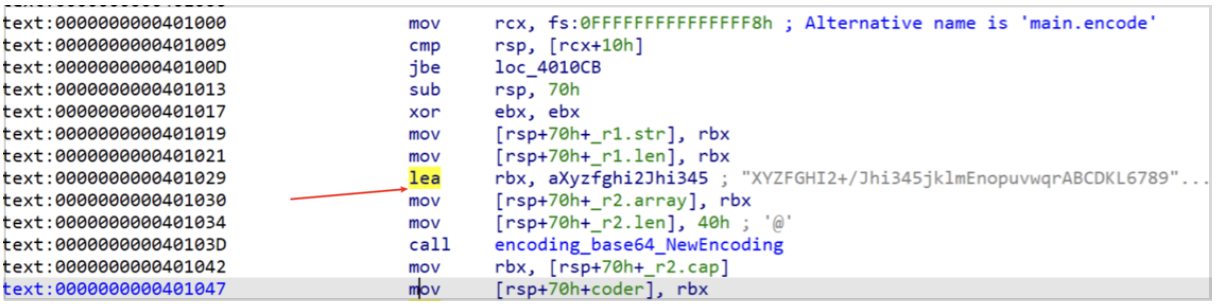
```
.rodata:00000... 00000045  C  *struct { sync.RWMutex; m map[reflect.layoutKey]reflect.layoutType }
.rodata:00000... 00000041  C  ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
.rodata:00000... 00000041  C  ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-_
.rodata:00000... 00000041  C  XYZFGHI2+/Jhi345jklmEnopuvwqrABCDKL6789abMNWcdefgstOPQRSTUVxyz01
```



```
.rodata:000000000054CE40 aXyzfghi2Jhi345 db 'XYZFGHI2+/Jhi345jklmEnopuvwqrABCDKL6789abMNWcdefgstOPQRSTUVxyz01',0
.rodata:000000000054CE40                                         ; DATA XREF: main_encode+29↑o
```

XYZFGHI2+/Jhi345jklmEnopuvwqrABCDKL6789abMNWcdefgstOPQRSTUVxyz01

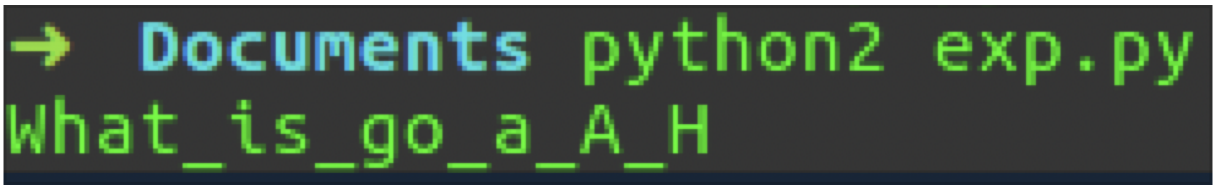
向上交叉引用溯源，得到



```
text:0000000000401000    mov     rcx, fs:0FFFFFFFFFFFFFFF8h ; Alternative name is 'main.encode'
text:0000000000401009    cmp     rsp, [rcx+10h]
text:000000000040100D    jbe     loc_4010CB
text:0000000000401013    sub     rsp, 70h
text:0000000000401017    xor     ebx, ebx
text:0000000000401019    mov     [rsp+70h+_r1.str], rbx
text:0000000000401021    mov     [rsp+70h+_r1.len], rbx
text:0000000000401029    lea     rbx, aXyzfghi2Jhi345 ; "XYZFGHI2+/Jhi345jklmEnopuvwqrABCDKL6789"...
text:0000000000401030    mov     [rsp+70h+_r2.array], rbx
text:0000000000401034    mov     [rsp+70h+_r2.len], 40h ; '@'
text:000000000040103D    call    encoding_base64_NewEncoding
text:0000000000401042    mov     rbx, [rsp+70h+_r2.cap]
text:0000000000401047    mov     [rsp+70h+coder], rbx
```

逆推算法，写出对应解题脚本为：

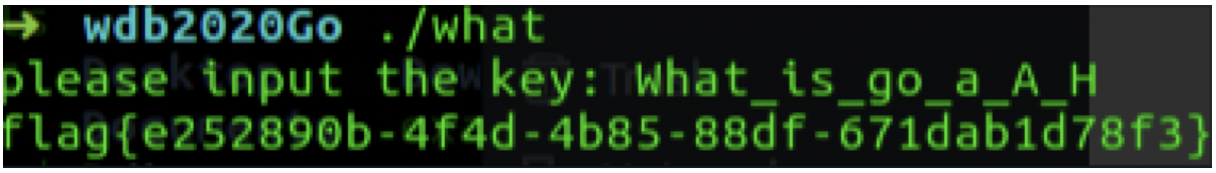
```python
import string

import base64

flag = 'nRKKAHzMrQzaqQzKpPHClX'

std_table = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'

my_table = 'XYZFGHI2+/Jhi345jklmEnopuvwqrABCDKL6789abMNWcdefgstOPQRSTUVxyz01'

flag = flag.translate(string.maketrans(my_table, std_table))

flag += "=="

print base64.b64decode(flag)
```



```
→  Documents python2 exp.py
What_is_go_a_A_H
```

运行程序后得到的结果为 WhatisgoaA_H

输入得到的内容 得到flag



```
→  wdb2020Go ./what
please input the key:What_is_go_a_A_H
flag{e252890b-4f4d-4b85-88df-671dab1d78f3}
```

第三步

获得flag

Flag{e252890b-4f4d-4b85-88df-671dab1d78f3}

Flag

flag{e252890b-4f4d-4b85-88df-671dab1d78f3}