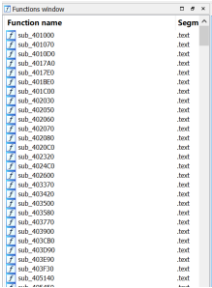
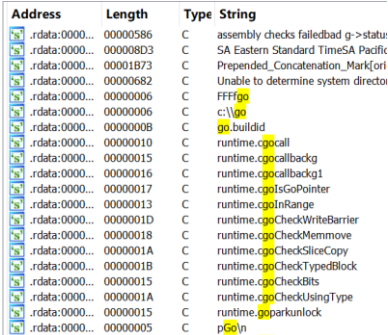


《rev》解题思路

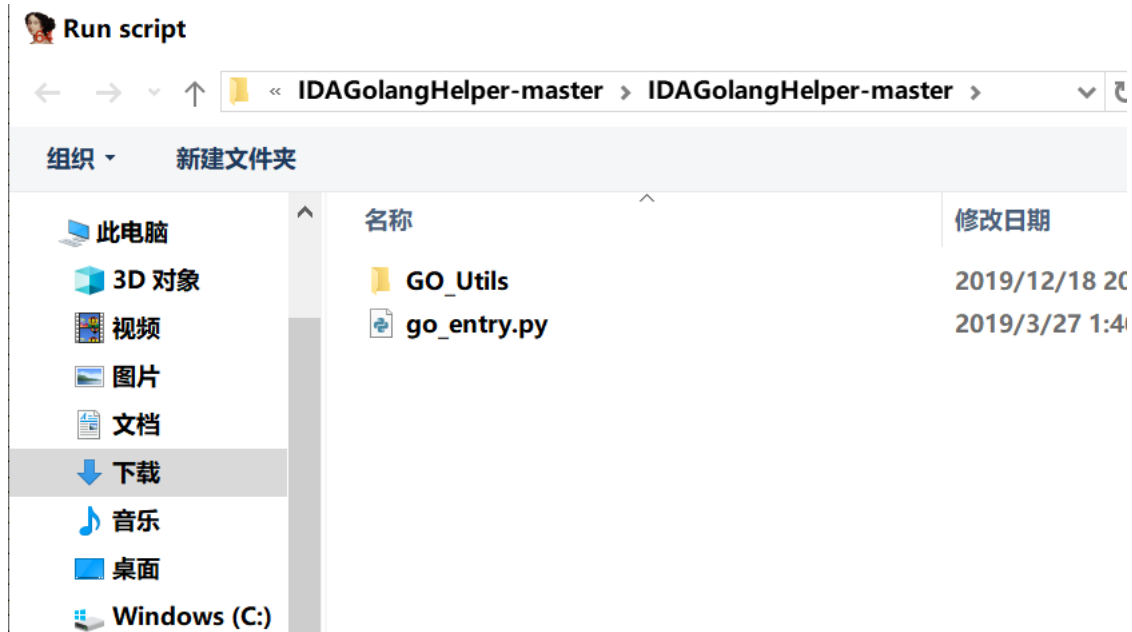
类型	REVERSE	
实验名称	rev	
实验目的与要求	1) 了解 go 逆向与符号恢复 2) 了解 base58 编码与逆向	
实验环境	实	无
	验	
	靶	

境	机	
	访 问 要 求	在线 Web 方式访问
	实 验 环 境	客户端 PC 机
	测 试 工	1) 分析器：IDA 7.0

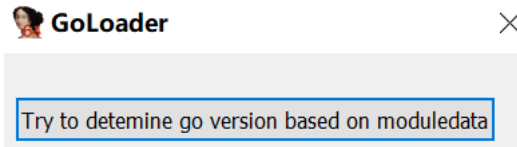
	具	
预备知识		<div>1) go 逆向符号恢复</div> <div>2) base58 编码与求逆</div>
实验步骤		<div>1) IDA 打开后发现被去除了符号</div> <div></div> <div>可以在字符串窗口中发现这是 go 编写的程序</div> <div></div>

2) 使用工具 IDAGolangHelper 来恢复符号, github 链接为: <https://github.com/sibears/IDAGolangHelper>

在 IDA 中选择 File-Script File, 导入 py 文件

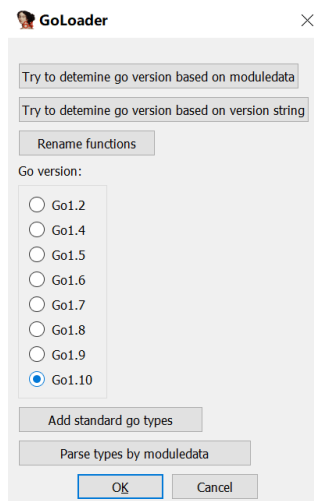


### 3) 选择



输出结果为 According to moduleData struct is should be go1.8 or go1.9 or go1.10

不妨就选择 go1.10, 没有太大影响, 选择 rename function



4) 这时符号就已经恢复好了，在函数窗口找到 main\_main，即主逻辑

在 main\_main 函数中，程序读取一个字符串，并追加字符串`go`，进入 Encode 函数，连续 3 次

```
fmt_Fscanf((__int64)&v34, (__int64)&off_51C1A0, v2, v3, (__int64)&v34, 1i64); // 程序读入一个字符串
v4 = v31[1];
runtime_concatstring2((unsigned __int64)&str_go, *v31, v5, v6); // 对字符串追加"go"
v7 = 1i64;
v8 = v31;
v31[1] = 1i64;
if ( dword_5DB090 )
    runtime_gcWriteBarrier();
else
    *v8 = v27;
runtime_stringtoslicebyte(v7, (__int64)&v29);
main_Encode(2i64);
runtime_concatstring2(v12, v9, v10, v11);
runtime_stringtoslicebyte(v27, (__int64)&v28);
main_Encode(2i64);
runtime_concatstring2(2, v13, v14, v15);
return v27;
```

5) 跟进 Encode 分析，可以看到关键的一个数组

.rdata:0000000000504FEB base58_table	db '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B'
.rdata:0000000000504FEB	; DATA XREF: main_Encode+1D6↑o
.rdata:0000000000504FEB	db 'C', 'D', 'E', 'F', 'G', 'H', 'J', 'K', 'L', 'M', 'N'
.rdata:0000000000504FEB	db 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'
.rdata:0000000000504FEB	db 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k'
.rdata:0000000000504FEB	db 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w'
.rdata:0000000000504FEB	db 'x', 'y', 'z'

```

62| while ( 1 )
63| {
64|     v37 = v7;
65|     v36 = v8;
66|     v38 = v6;
67|     math_big_ptr_Int_Cmp();
68|     if ( v11 <= 0 )
69|         break;
70|     v39 = 0;
71|     v40 = 0i64;
72|     v41 = 0i64;
73|     math_big_ptr_Int_DivMod((__int64)&v39, v9);
74|     if ( (_QWORD)v41 )
75|         v15 = *v40;
76|     else
77|         v15 = 0i64;
78|     if ( v39 )
79|         v15 = -(signed __int64)v15;
80|     if ( v15 >= 0x3A )
81|         goto LABEL_28;
82|     v16 = v36;
83|     v8 = v36 + 1;
84|     v17 = base58_table[v15];
85|     v18 = v37;

```

很明显是 base58 的码表，在这里命名为 base58\_table (84 行)，码表即为原生的 base58 码表，没有改变。

最后 3 轮加密后，和字符串

9JanaG7xcRhMTqkZWMSyrp9mjMXvopfVt9dBkL6tHPD4H6gjPAvag8ftNP4DMMUq6Y6go

进行比较，相等则输出 You win 等信息

6) 求逆, 3 轮: 每轮先减去减去最后的两个字符 go, 再 base58Decode 即可, exp 如下:

```
func main() {  
    sss := "9JanaG7xcRhMTqkZWMSyrp9mjMXvopfVt9dBkL6tHPD4H6gjPAvag8ftNP4DMMUq6Y6go"  
    sss = sss[0 : len(sss)-2]  
  
    ss := Decode(string(sss))  
    ss = ss[0 : len(ss)-2]  
  
    s := Decode(string(ss))  
    s = s[0 : len(s)-2]  
  
    fmt.Println(string(s))  
}
```

最终运行:

 选择Windows PowerShell

```
PS D:\> go run .\exp.go  
e110a6740c8801a8b8eec4929e0e44b0
```

```
PS D:\> .\rev.exe  
e110a6740c8801a8b8eec4929e0e44b0  
You win!  
flag{e110a6740c8801a8b8eec4929e0e44b0}
```



原理 知识	1) go 生成的 exe 如果被去除了符号，可以用 IDAGolangHelper 来恢复符号，对应主逻辑为函数 main_main
防护 方法	1) 使用更复杂的算法，对字符串进行更复杂的变换
分析 与思 考	1) IDAGolangHelper 的原理？