ByteCTF2021 reverse languagebinding writeup

原创

1mmorta1 于 2021-10-18 00:53:44 发布 141 收藏 1

分类专栏： reverse 文章标签： lua windows go语言

本文链接：https://blog.csdn.net/qq_41866334/article/details/120818939

Language Binding

> AAA : immortal

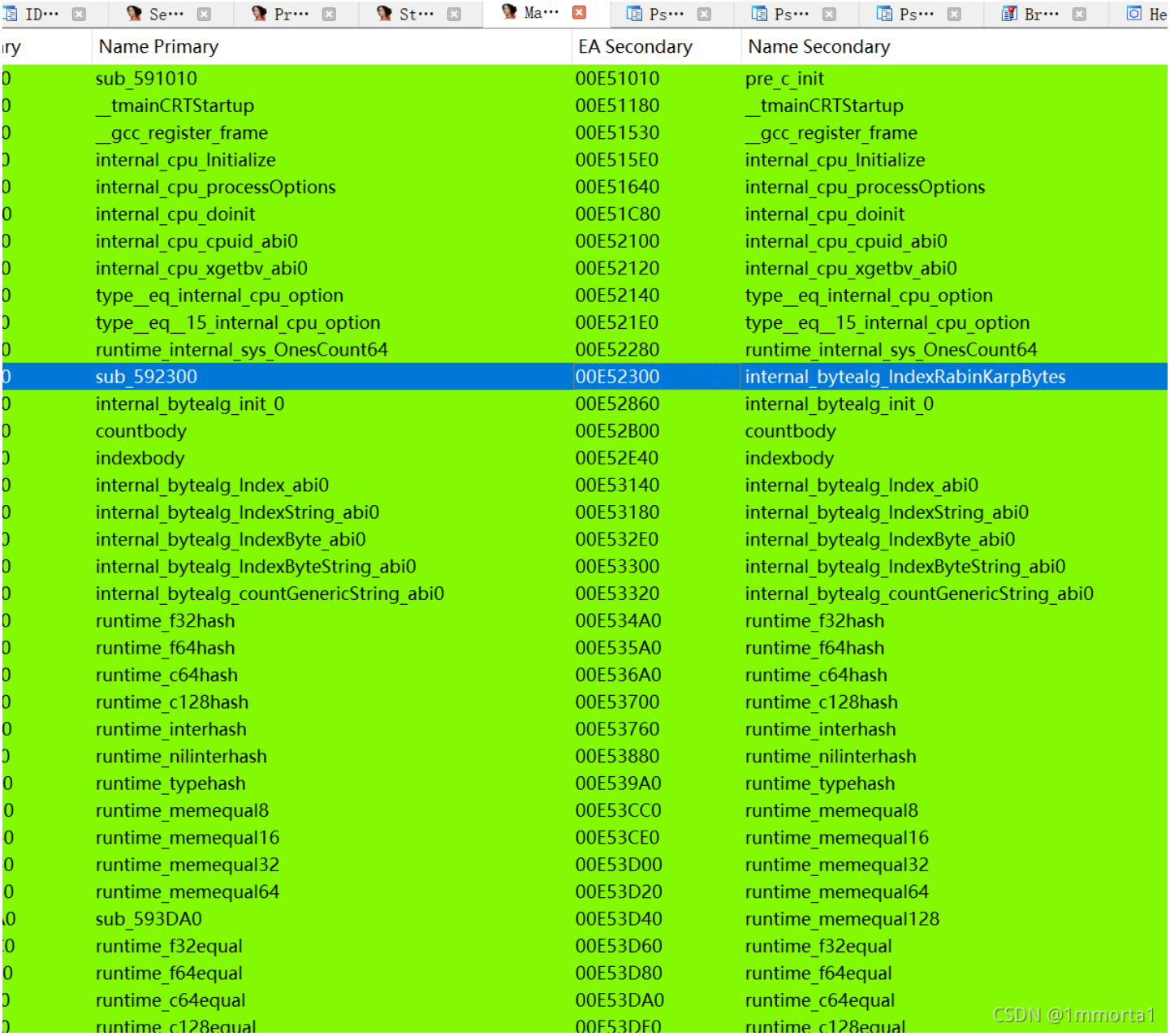
拿到题目ida打开发现是go语言逆向，而且函数名都被混淆掉了，动态调了一下整个人都傻了。

之后尝试直接打开了new_lang_script.out，看了一下有大量的0x55所以估计就是异或了0x55，解密出来发现是一个luac文件，修改文件头以后直接用luac -l执行，发现会报错。和我自己写的lua脚本比对会发现题目luac中的opcode全都是打乱的。

```
┌──(immortal㉿kali)-[~/ctf/reverse/ByteCTF/languagebind]
└─$ ./lua-5.3.6/src/luac -l  test.lua

main <test.lua:0,0> (15 instructions at 0x55f43690ec60)
0+ params, 2 slots, 1 upvalue, 0 locals, 7 constants, 0 functions
        1       [1]     GETTABUP        0 0 -1  ; _ENV "print"
        2       [1]     LOADK           1 -2    ; "input flag:\n"
        3       [1]     CALL            0 2 1
        4       [2]     GETTABUP        0 0 -4  ; _ENV "io"
        5       [2]     GETTABLE        0 0 -5  ; "read"
        6       [2]     CALL            0 1 2
        7       [2]     SETTABUP        0 -3 0  ; _ENV "flag"
        8       [3]     GETTABUP        0 0 -3  ; _ENV "flag"
        9       [3]     LEN             0 0
        10      [3]     EQ              1 0 -6  ; - 29
        11      [3]     JMP             0 3     ; to 15
        12      [5]     GETTABUP        0 0 -1  ; _ENV "print"
        13      [5]     LOADK           1 -7    ; "flag is wrong\n"
        14      [5]     CALL            0 2 1
        15      [6]     RETURN          0 1

┌──(immortal㉿kali)-[~/ctf/reverse/ByteCTF/languagebind]
└─$ ./lua-5.3.6/src/luac -l xor_new_lang_script.out

main <flag.lua:0,0> (69 instructions at 0x5583b4b49c40)
0+ params, 5 slots, 1 upvalue, 3 locals, 22 constants, 4 functions
        1       [2]     ADD             0 0 -1  ; - "print"
        2       [2]     VARARG          1 0
        3       [2]     SHR             0 2 1
        4       [3]     ADD             1 0 -4  ; - "io"
        5       [3]     LE              0 1 -5  ; - "read"
        6       [3]     SHR             0 1 2
        7       [3]     LEN             0 -3
        8       [5]     ADD             2 0 -3  ; - "flag"
        9       [5]     SUB             1 2 0
        10      [5]     CONCAT          0 1 -6
        11      [5]     EQ              0 -1 0  ; "print" -
        12      [5]     MOD             0 0 1
        13      [5]     MOD             0 1 0
        14      [5]     BOR             0 0 0
        15      [5]     EQ              0 -1 3  ; "print" -
        16      [6]     ADD             0 0 -1  ; - "print"
        17      [6]     VARARG          1 0
        18      [6]     SHR             0 2 1
        19      [7]     TFORLOOP        0 -130559       ; to -130539
        20      [10]    GETTABUP        0 3 0   ; E
        21      [10]    VARARG          1 0
        22      [10]    VARARG          2 0
        23      [10]    VARARG          3 0
        24      [10]    LOADK           0 -1538 ; nil
```


此时猜测这个程序就是用go语言写的一个执行luac脚本的编译器，因此真正解释执行的opcode顺序一定藏在exe文件之中。但是这里就有个问题，对于没有符号的go语言逆向真的痛苦。

最后队内大哥通过google找到了这个程序的源代码地址：github . 所以就自己装了一个go解释器然后在windows下编译该文件，利用bindiff进行符号表的还原。



ry	Name Primary	EA Secondary	Name Secondary
0	sub_591010	00E51010	pre_c_init
0	__tmainCRTStartup	00E51180	__tmainCRTStartup
0	__gcc_register_frame	00E51530	__gcc_register_frame
0	internal_cpu_Initialize	00E515E0	internal_cpu_Initialize
0	internal_cpu_processOptions	00E51640	internal_cpu_processOptions
0	internal_cpu_doinit	00E51C80	internal_cpu_doinit
0	internal_cpu_cpuid_abi0	00E52100	internal_cpu_cpuid_abi0
0	internal_cpu_xgetbv_abi0	00E52120	internal_cpu_xgetbv_abi0
0	type__eq_internal_cpu_option	00E52140	type__eq_internal_cpu_option
0	type__eq__15_internal_cpu_option	00E521E0	type__eq__15_internal_cpu_option
0	runtime_internal_sys_OnesCount64	00E52280	runtime_internal_sys_OnesCount64
0	sub_592300	00E52300	internal_bytealg_IndexRabinKarpBytes
0	internal_bytealg_init_0	00E52860	internal_bytealg_init_0
0	countbody	00E52B00	countbody
0	indexbody	00E52E40	indexbody
0	internal_bytealg_Index_abi0	00E53140	internal_bytealg_Index_abi0
0	internal_bytealg_IndexString_abi0	00E53180	internal_bytealg_IndexString_abi0
0	internal_bytealg_IndexByte_abi0	00E532E0	internal_bytealg_IndexByte_abi0
0	internal_bytealg_IndexByteString_abi0	00E53300	internal_bytealg_IndexByteString_abi0
0	internal_bytealg_countGenericString_abi0	00E53320	internal_bytealg_countGenericString_abi0
0	runtime_f32hash	00E534A0	runtime_f32hash
0	runtime_f64hash	00E535A0	runtime_f64hash
0	runtime_c64hash	00E536A0	runtime_c64hash
0	runtime_c128hash	00E53700	runtime_c128hash
0	runtime_interhash	00E53760	runtime_interhash
0	runtime_nilinterhash	00E53880	runtime_nilinterhash
0	runtime_typehash	00E539A0	runtime_typehash
0	runtime_memequal8	00E53CC0	runtime_memequal8
0	runtime_memequal16	00E53CE0	runtime_memequal16
0	runtime_memequal32	00E53D00	runtime_memequal32
0	runtime_memequal64	00E53D20	runtime_memequal64
0	sub_593DA0	00E53D40	runtime_memequal128
0	runtime_f32equal	00E53D60	runtime_f32equal
0	runtime_f64equal	00E53D80	runtime_f64equal
0	runtime_c64equal	00E53DA0	runtime_c64equal
0	runtime_c128equal	00E53DE0	runtime_c128equal

在符号表的帮助下，通过动态调试找到了opcode的表，并且对所有luago_vm_下的函数进行了矫正还原。可以看到opcodes表patch了name，要求reverser必须逆向处对应的函数。

```
                dd    0
opcodes         dq 20100h               ; DATA XREF: .data:off_6AFD90↑o
                dq offset aAddBorDivLenMo+549h ; "NONNAMEOctoberOsmanyaRadicalSharadaShav"...
                dq 7
                dq offset off_6EBE00
                dq 1010000h
                dq offset aAddBorDivLenMo+549h ; "NONNAMEOctoberOsmanyaRadicalSharadaShav"...
                dq 7
                dq offset off_6EBDA0
                dq 1020101h
                dq offset aAddBorDivLenMo+549h ; "NONNAMEOctoberOsmanyaRadicalSharadaShav"...
                dq 7
                dq offset off_6EBDF0
                dq 100010100h
                dq offset aAddBorDivLenMo+549h ; "NONNAMEOctoberOsmanyaRadicalSharadaShav"...
                dq 7
                dq offset off_6EBCD8
                dq 100000100h
                dq offset aAddBorDivLenMo+549h ; "NONNAMEOctoberOsmanyaRadicalSharadaShav"...
                dq 7
                dq offset off_6EBD48
                dq 200020100h
                dq offset aAddBorDivLenMo+549h ; "NONNAMEOctoberOsmanyaRadicalSharadaShav"...
                dq 7
                dq offset off_6EBDE0
                dq 1010100h
                dq offset aAddBorDivLenMo+549h ; "NONNAMEOctoberOsmanyaRadicalSharadaShav"...
                dq 7
```


于是就魔改了已有的反编译luac的工具unluac 中的opcodes表，反编译出的结果如下：

```
print("input flag:")
flag = io.read()
if #flag ~= 29 then
  print("flag is wrong")
  return
end
lst = {
  100,
  120,
  133
}
dict = {
  [9] = 101,
  [10] = 122
}

function ad(a, b)
  return a + b
end
function mul(a, b)
  return a * b
end
function check2(f)
  if string.byte(f, 18) + 11 ~= 106 then
    return false
  elseif string.byte(f, 19) ~= lst[1] + 21 then
    return false
  elseif string.byte(f, 20) ~= dict[9] + 8 then
    return false
  elseif string.byte(f, 21) ~= ad(100, 22) then
    return false
  elseif string.byte(f, 22) ~= 55 then
    return false
  elseif string.byte(f, 23) ~= mul(51, 2) then
    return false
```

```
  elseif string.byte(f, 24) - 1 ~= 108 then
    return false
  elseif string.byte(f, 25) ~= 48 then
    return false
  elseif string.byte(f, 26) ~= 100 then
    return false
  elseif string.byte(f, 27) ~= 102 then
    return false
  elseif string.byte(f, 28) ~= 120 then
    return false
  end
  return true
end
function check3(f)
  if string.byte(f, 1) ~= 66 then
    return false
  elseif string.byte(f, 2) ~= 121 then
    return false
  elseif string.byte(f, 3) ~= 116 then
    return false
  elseif string.byte(f, 4) ~= 101 then
    return false
  elseif string.byte(f, 5) ~= 67 then
    return false
  elseif string.byte(f, 6) ~= 84 then
    return false
  elseif string.byte(f, 7) ~= 70 then
    return false
  elseif string.byte(f, 8) ~= 123 then
    return false
  elseif string.byte(f, 29) ~= 125 then
    return false
  end
  return true
end
local flag1 = _(flag)
local flag2 = check2(flag)
local flag3 = check3(flag)
if flag1 and flag2 and flag3 then
  print("flag is right")
elseif true then
  print("flag is wrong")
end
```

本来以为已经要出来了，结果一看发现flag还是少了9个字节。因为题目名字叫language binding，所以猜测lua脚本中的下划线是一个预定义过的goFunction。

和原框架比对，可以发现main_main函数被修改了，添加了PushGoFunction 和 PushInteger

```
__int64 __fastcall main_main(__int64 a1, __int64 a2, __int64 a3, __int64 a4)
{
  __int64 v4; // r14
  __int64 v5; // rdx
  __int64 v6; // r8
  __int64 v7; // r9
  __int64 *v8; // rax
  __int64 v9; // rdx
  __int64 v10; // r8
  __int64 v11; // r9
  __int64 v13; // [rsp+0h] [rbp-38h]
  void **v14; // [rsp+8h] [rbp-30h]
  __int64 v15; // [rsp+10h] [rbp-28h]
  void *retaddr; // [rsp+38h] [rbp+0h] BYREF

  if ( (unsigned __int64)&retaddr <= *(_QWORD *)(v4 + 16) )
    runtime_morestack_noctxt_abi0();
  v15 = qword_7C2D60;
  v14 = luago_state_New();
  ((void (*)(void))v14[82])();                  // luago_state___luaState__PushGoFunction
  ((void (*)(void))v14[83])();                  // luago_state___luaState__PushInteger
  v8 = runtime_convTslice(v15, v5, v6, v7);
  ((void (__fastcall *)(__int64 *))v14[84])(v8);
  v13 = ((__int64 (__fastcall *)(__int64))v14[75])(1i64);// luago_state___luaState__PCall
  ((void (*)(void))v14[118])();
  sub_698320(v13, v9, v10, v11);
  return ((__int64 (*)(void))v14[19])();
}
```


再回去看仓库的源代码，我们发现PushGoFunction的参数就是GoFunction。但是跟进去并没有发现有用的逻辑。

后来还是在大哥的帮助下定位到了 `luago_state___luaState__runLuaClosure` 函数，并断在下图位置进行动调。

```
__int64 __fastcall luago_state___luaState__runLuaClosure(__int64 a1, __int64 a2, __int64 a3, __int64 a4)
{
  __int64 v4; // rax
  __int64 v5; // r14
  __int64 v6; // rsi
  __int64 v7; // rdi
  unsigned __int64 v8; // rcx
  __int64 v9; // rdi
  unsigned __int64 v10; // r8
  unsigned int v11; // edi
  __int64 v12; // r8
  unsigned __int64 v13; // rdi
  __int64 result; // rax
  int v15; // [rsp+0h] [rbp-Ch]
  void *retaddr; // [rsp+Ch] [rbp+0h] BYREF
  __int64 i; // [rsp+14h] [rbp+8h]

  if ( (unsigned __int64)&retaddr <= *(_QWORD *)(v5 + 16) )
    runtime_morestack_noctxt_abi0();
  for ( i = v4; ; v4 = i )
  {
    v6 = *(_QWORD *)(v4 + 32);
    v7 = **(_QWORD **)(v6 + 40);
    v8 = *(_QWORD *)(v7 + 40);
    v9 = *(_QWORD *)(v7 + 32);
    v10 = *(_QWORD *)(v6 + 80);
    if ( v8 <= v10 )
      runtime_panicIndex(v8, a2, v10, a4);
    v11 = *(_DWORD *)(v9 + 4 * v10);
    *(_QWORD *)(v6 + 80) = v10 + 1;
    v12 = v11;
    v13 = v11 & 0x3F;
    if ( qword_6AFD98 <= v13 )
      runtime_panicIndex(qword_6AFD98, a2, v12, a4);
    result = (*(__int64 (__fastcall **)(__int64, __int64, __int64, __int64))off_6AFD90[4 * v13 + 3])(// luago_vm_Instruction_Execute
               v4,
               off_6AFD90[4 * v13 + 3],
               v12,
               a4);
    if ( v15 == 42i64 )
      break;
  }
  return result;
}
```


根据附录中已有的luac字节码我们能跟进call _ 这个指令中（即下文中的call %0 2 2）

```
setTabUp @0 const "check3" %0 ; debug name: _ENV
getTabUp %0 @0 const "_" ; debug name: _ENV
getTabUp %1 @0 const "flag" ; debug name: _ENV
call %0 2 2
```

发现其调用链如下：

`luago_state___luaState__Call` --> `luago_state___luaState__callGoClosure`

且因为`luago_state___luaState__callGoClosure`仅有一个动态调用，所以动调查看下图的

rcx，终于跟到了核心逻辑在 sub_CA8D80。

```
loc_68A599:                             ; CODE XREF: luago_state_
                lea     rdi, [rax+58h]
                nop     dword ptr [rax]
                call    runtime_gcWriteBarrierR8
                mov     rdi, rcx
                call    runtime_gcWriteBarrier

loc_68A5AD:                             ; CODE XREF: luago_state_
                inc     qword ptr [rbx+28h]
                mov     rdx, [rsi+8]
                mov     rcx, [rdx]
                lea     rax, off_714150
                nop
                call    rcx
                mov     rcx, [rsp+88h+arg_0]
                mov     rsi, [rcx+20h]
                nop
                mov     rdi, [rsi+58h]
                cmp     cs:dword_816FE0, 0
                jnz     short loc_68A5EA
                mov     [rcx+20h], rdi
                mov     qword ptr [rsi+58h], 0
                jmp     short loc_68A605
; ---------------------------------------------------------------------------
```



```
__int64 __fastcall sub_CA8D80()
{
  __int64 v0; // rax
  __int64 v1; // r14
  unsigned __int64 v2; // rbx
  __int64 v3; // rax
  __int64 v4; // r8
  __int64 v5; // r9
  __int64 v6; // rcx
  __int64 v7; // rdx
  char v9[8]; // [rsp+0h] [rbp-3Ah]
  _BYTE v10[10]; // [rsp+8h] [rbp-32h] BYREF
  void *retaddr; // [rsp+3Ah] [rbp+0h] BYREF
  __int64 v12; // [rsp+42h] [rbp+8h]

  if ( (unsigned __int64)&retaddr <= *(_QWORD *)(v1 + 16) )
    runtime_morestack_noctxt_abi0();
  v12 = v0;
  qmemcpy(v10, "9negozc9aj", sizeof(v10));
  v2 = (*(__int64 (__fastcall **)(_QWORD))(v0 + 1000))(*(_QWORD *)(v0 + 1000));
```

```
  v3 = runtime_stringtoslicebyte(-1i64);
  v6 = 8i64;
  v7 = 1i64;
  while ( v6 < 17 )
  {
    if ( v2 <= v6 )
      runtime_panicIndex(v2, v7, v4, v5);
    if ( v9[v6] != ((unsigned __int8)v6 ^ *(_BYTE *)(v3 + v6)) )
      v7 = 0i64;
    ++v6;
  }
  (*(void (**)(void))(v12 + 624))();
  return 1i64;
}
```


最后的flag为：ByteCTF{1golcwm6q_ymz7fm0dfx}

附录

luac脚本反编译出的字节码

```
.upvalues 1

.func main 5 0 1 ; source: @flag.lua ; maxstacksize: 5, params: 0, vararg: 1 (uses vararg)
.begin_const
   "print"
   "input flag:"
   "flag"
   "io"
   "read"
   29.000000
   "flag is wrong"
   "lst"
   100.000000
   120.000000
   133.000000
   "dict"
   9.000000
   101.000000
   10.000000
   122.000000
   "ad"
   "mul"
   "check2"
   "check3"
   "_"
   "flag is right"
.end_const


.begin_upvalue
   1 0
.end_upvalue


.begin_code
   getTabUp %0 @0 const "print" ; debug name: _ENV
   loadK %1 const "input flag:"
```

```
   call %0 2 1
   getTabUp %1 @0 const "io" ; debug name: _ENV
   getTable %0 %1 const "read"
   call %0 1 2
   setTabUp @0 const "flag" %0 ; debug name: _ENV
   getTabUp %2 @0 const "flag" ; debug name: _ENV
   length %1 %2
   eq false %1 const 29.000000
   jmp 0 $location_12
   loadBool %0 false 1
location_12:
   loadBool %0 true 0
   test %0 false
   jmp 0 $location_19
   getTabUp %0 @0 const "print" ; debug name: _ENV
   loadK %1 const "flag is wrong"
   call %0 2 1
   return %0 1
location_19:
   newTable %0 3 0
   loadK %1 const 100.000000
   loadK %2 const 120.000000
   loadK %3 const 133.000000
   setList %0 3 1
   setTabUp @0 const "lst" %0 ; debug name: _ENV
   newTable %0 0 2
   loadK %1 const 9.000000
   loadK %2 const 101.000000
   setTable %0 %1 %2
   loadK %1 const 10.000000
   loadK %2 const 122.000000
   setTable %0 %1 %2
   setTabUp @0 const "dict" %0 ; debug name: _ENV
   closure %0 subroutine_2
   setTabUp @0 const "ad" %0 ; debug name: _ENV
   closure %0 subroutine_3
   setTabUp @0 const "mul" %0 ; debug name: _ENV
   closure %0 subroutine_4
   setTabUp @0 const "check2" %0 ; debug name: _ENV
   closure %0 subroutine_5
   setTabUp @0 const "check3" %0 ; debug name: _ENV
   getTabUp %0 @0 const "_" ; debug name: _ENV
   getTabUp %1 @0 const "flag" ; debug name: _ENV
   call %0 2 2
   getTabUp %1 @0 const "check2" ; debug name: _ENV
   getTabUp %2 @0 const "flag" ; debug name: _ENV
   call %1 2 2
   getTabUp %2 @0 const "check3" ; debug name: _ENV
   getTabUp %3 @0 const "flag" ; debug name: _ENV
   call %2 2 2
   testSet %4 %0 false
   jmp 0 $location_53
   move %4 %1
location_53:
   testSet %3 %4 false
   jmp 0 $location_56
   move %3 %2
location_56:
   test %3 false
   jmp 0 $location_62
```

```
   getTabUp %3 @0 const "print" ; debug name: _ENV
   loadK %4 const "flag is right"
   call %3 2 1
   jmp 0 $location_68
location_62:
   loadBool %3 true 0
   test %3 false
   jmp 0 $location_68
   getTabUp %3 @0 const "print" ; debug name: _ENV
   loadK %4 const "flag is wrong"
   call %3 2 1
location_68:
   return %0 1
.end_code


.func subroutine_2 3 2 0 ; maxstacksize: 3, params: 2, vararg: 0 (does not use varag)
.begin_const
.end_const


.begin_upvalue
.end_upvalue


.begin_code
   add %2 %0 %1
   return %2 2
   return %0 1
.end_code



.func subroutine_3 3 2 0 ; maxstacksize: 3, params: 2, vararg: 0 (does not use varag)
.begin_const
.end_const


.begin_upvalue
.end_upvalue


.begin_code
   mul %2 %0 %1
   return %2 2
   return %0 1
.end_code



.func subroutine_4 6 1 0 ; maxstacksize: 6, params: 1, vararg: 0 (does not use varag)
.begin_const
   "string"
   "byte"
   18.000000
   11.000000
   106.000000
   19.000000
   "lst"
```

```
   1.000000
   21.000000
   20.000000
   "dict"
   9.000000
   8.000000
   "ad"
   100.000000
   22.000000
   55.000000
   23.000000
   "mul"
   51.000000
   2.000000
   24.000000
   108.000000
   25.000000
   48.000000
   26.000000
   27.000000
   102.000000
   28.000000
   120.000000
.end_const


.begin_upvalue
   0 0
.end_upvalue


.begin_code
   getTabUp %4 @0 const "string" ; debug name: _ENV
   getTable %3 %4 const "byte"
   move %4 %0
   loadK %5 const 18.000000
   call %3 3 2
   add %2 %3 const 11.000000
   eq false %2 const 106.000000
   jmp 0 $location_9
   loadBool %1 false 1
location_9:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_15
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_169
location_15:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 19.000000
   call %2 3 2
   getTabUp %5 @0 const "lst" ; debug name: _ENV
   getTable %4 %5 const 1.000000
   add %3 %4 const 21.000000
   eq false %2 %3
   jmp 0 $location_26
   loadBool %1 false 1
```

```
   loadBool %1 false 1
location_26:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_32
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_169
location_32:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 20.000000
   call %2 3 2
   getTabUp %5 @0 const "dict" ; debug name: _ENV
   getTable %4 %5 const 9.000000
   add %3 %4 const 8.000000
   eq false %2 %3
   jmp 0 $location_43
   loadBool %1 false 1
location_43:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_49
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_169
location_49:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 21.000000
   call %2 3 2
   getTabUp %3 @0 const "ad" ; debug name: _ENV
   loadK %4 const 100.000000
   loadK %5 const 22.000000
   call %3 3 2
   eq false %2 %3
   jmp 0 $location_61
   loadBool %1 false 1
location_61:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_67
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_169
location_67:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 22.000000
   call %2 3 2
   eq false %2 const 55.000000
   jmp 0 $location_75
   loadBool %1 false 1
location_75:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_81
```

```
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_169
location_81:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 23.000000
   call %2 3 2
   getTabUp %3 @0 const "mul" ; debug name: _ENV
   loadK %4 const 51.000000
   loadK %5 const 2.000000
   call %3 3 2
   eq false %2 %3
   jmp 0 $location_93
   loadBool %1 false 1
location_93:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_99
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_169
location_99:
   getTabUp %4 @0 const "string" ; debug name: _ENV
   getTable %3 %4 const "byte"
   move %4 %0
   loadK %5 const 24.000000
   call %3 3 2
   sub %2 %3 const 1.000000
   eq false %2 const 108.000000
   jmp 0 $location_108
   loadBool %1 false 1
location_108:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_114
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_169
location_114:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 25.000000
   call %2 3 2
   eq false %2 const 48.000000
   jmp 0 $location_122
   loadBool %1 false 1
location_122:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_128
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_169
location_128:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
```

```
   move %3 %0
   loadK %4 const 26.000000
   call %2 3 2
   eq false %2 const 100.000000
   jmp 0 $location_136
   loadBool %1 false 1
location_136:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_142
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_169
location_142:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 27.000000
   call %2 3 2
   eq false %2 const 102.000000
   jmp 0 $location_150
   loadBool %1 false 1
location_150:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_156
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_169
location_156:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 28.000000
   call %2 3 2
   eq false %2 const 120.000000
   jmp 0 $location_164
   loadBool %1 false 1
location_164:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_169
   loadBool %1 false 0
   return %1 2
location_169:
   loadBool %1 true 0
   return %1 2
   return %0 1
.end_code



.func subroutine_5 5 1 0 ; maxstacksize: 5, params: 1, vararg: 0 (does not use varag)
.begin_const
   "string"
   "byte"
   1.000000
   66.000000
   2.000000
   121.000000
```

```
   3.000000
   116.000000
   4.000000
   101.000000
   5.000000
   67.000000
   6.000000
   84.000000
   7.000000
   70.000000
   8.000000
   123.000000
   29.000000
   125.000000
.end_const


.begin_upvalue
   0 0
.end_upvalue


.begin_code
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 1.000000
   call %2 3 2
   eq false %2 const 66.000000
   jmp 0 $location_8
   loadBool %1 false 1
location_8:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_14
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_125
location_14:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 2.000000
   call %2 3 2
   eq false %2 const 121.000000
   jmp 0 $location_22
   loadBool %1 false 1
location_22:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_28
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_125
location_28:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 3.000000
   call %2 3 2
```

```
   call %2 3 2
   eq false %2 const 116.000000
   jmp 0 $location_36
   loadBool %1 false 1
location_36:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_42
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_125
location_42:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 4.000000
   call %2 3 2
   eq false %2 const 101.000000
   jmp 0 $location_50
   loadBool %1 false 1
location_50:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_56
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_125
location_56:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 5.000000
   call %2 3 2
   eq false %2 const 67.000000
   jmp 0 $location_64
   loadBool %1 false 1
location_64:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_70
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_125
location_70:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 6.000000
   call %2 3 2
   eq false %2 const 84.000000
   jmp 0 $location_78
   loadBool %1 false 1
location_78:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_84
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_125
location_84:
```

```
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 7.000000
   call %2 3 2
   eq false %2 const 70.000000
   jmp 0 $location_92
   loadBool %1 false 1
location_92:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_98
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_125
location_98:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 8.000000
   call %2 3 2
   eq false %2 const 123.000000
   jmp 0 $location_106
   loadBool %1 false 1
location_106:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_112
   loadBool %1 false 0
   return %1 2
   jmp 0 $location_125
location_112:
   getTabUp %3 @0 const "string" ; debug name: _ENV
   getTable %2 %3 const "byte"
   move %3 %0
   loadK %4 const 29.000000
   call %2 3 2
   eq false %2 const 125.000000
   jmp 0 $location_120
   loadBool %1 false 1
location_120:
   loadBool %1 true 0
   test %1 false
   jmp 0 $location_125
   loadBool %1 false 0
   return %1 2
location_125:
   loadBool %1 true 0
   return %1 2
   return %0 1
.end_code
```