Linux/Android 内核 Heap Spray的几种姿势

盘古实验室　River

内容概要

- For newbies
  - 基础简介，环境搭建
  - SLUB细节和特性
- 讨论内核Heap Spray
  - 两个实例
  - 几种spray的实际应用

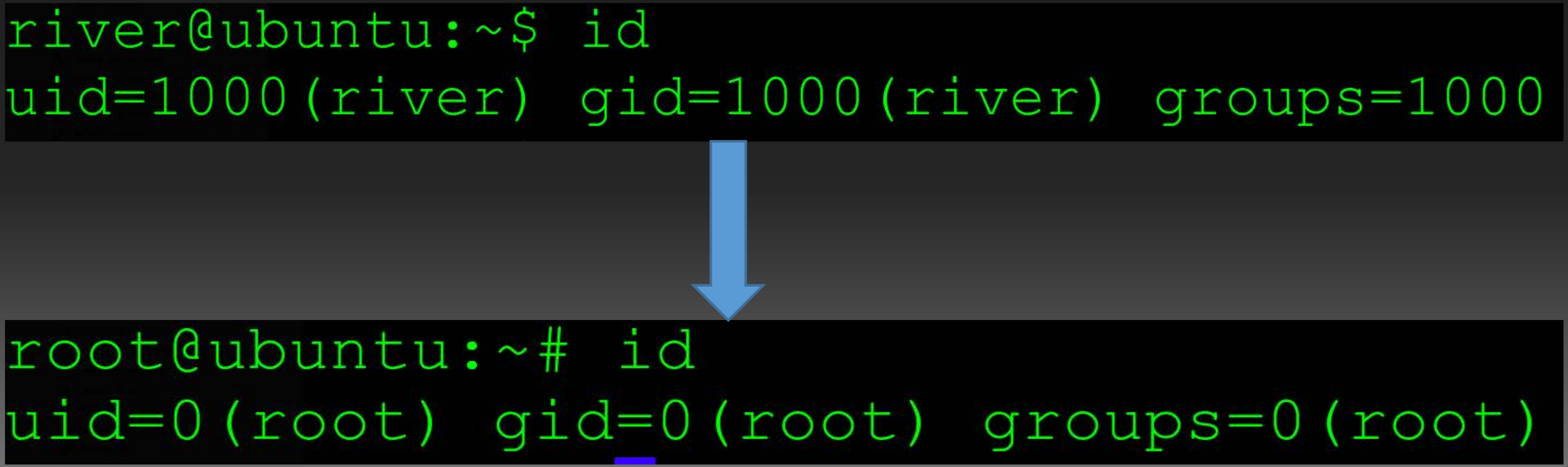
For newbies

DEFCON GROUP 0531
HACKER COMMUNITY

PANGU TEAM

Android/Linux 内核提权

- 获取Root权限（通过漏洞）
  - $ -> #
  - 改变uid，gid

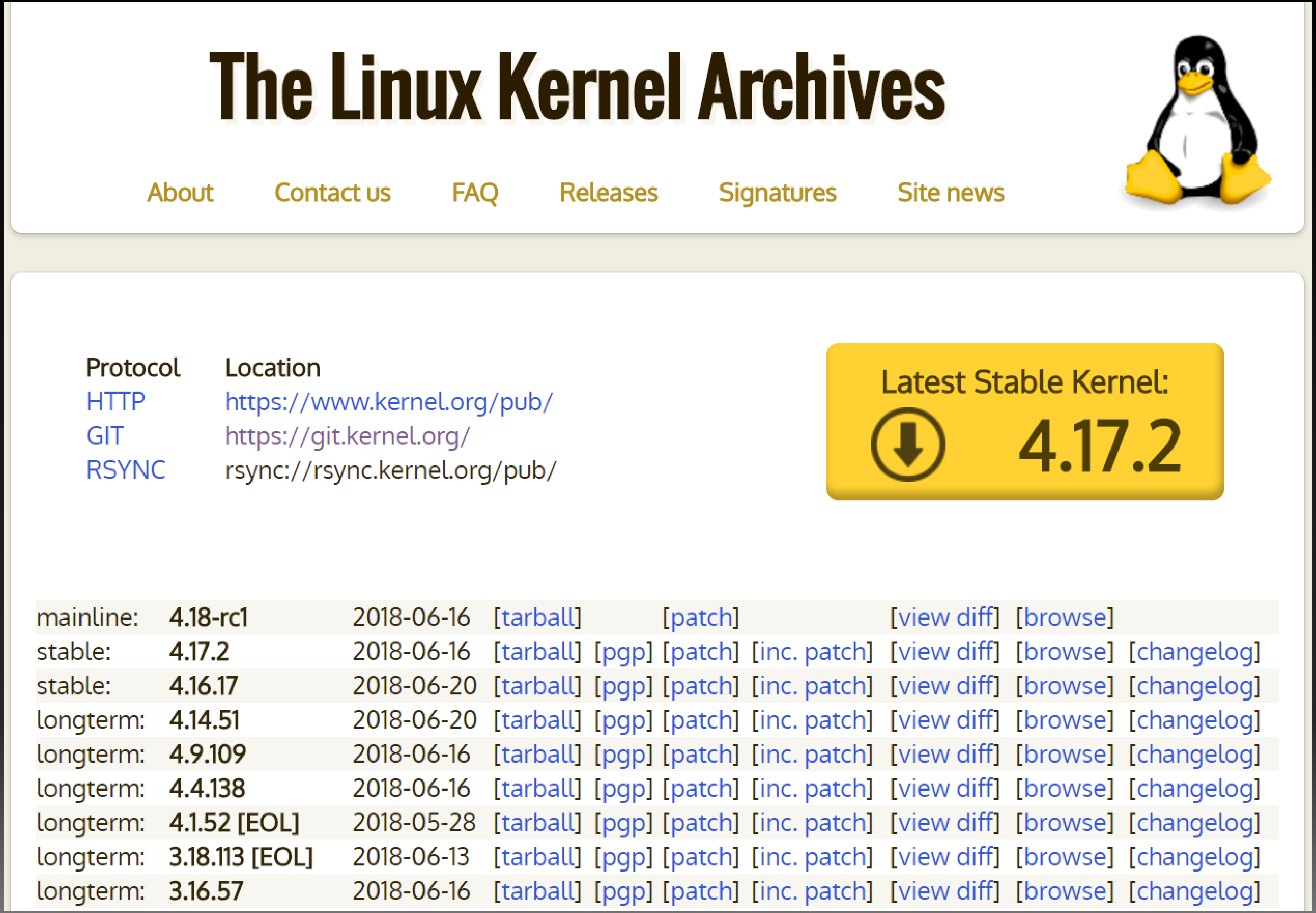


```
river@ubuntu:~$ id
uid=1000(river) gid=1000(river) groups=1000
```

```
root@ubuntu:~# id
uid=0(root) gid=0(root) groups=0(root)
```

Linux

- 开源
  - 仅仅是内核
  - 发行版(Ubuntu，Arch，Debian,CentOS...) != Linux
  - GNU+Linux = GNU/Linux --> 发行版(Ubuntu，Arch，Debian,CentOS...)
- www.kernel.org



The Linux Kernel Archives

About | Contact us | FAQ | Releases | Signatures | Site news

Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Stable Kernel: 4.17.2

mainline:	4.18-rc1	2018-06-16	[tarball]		[patch]		[view diff]	[browse]	
stable:	4.17.2	2018-06-16	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
stable:	4.16.17	2018-06-20	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.14.51	2018-06-20	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.9.109	2018-06-16	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.4.138	2018-06-16	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.1.52 [EOL]	2018-05-28	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.18.113 [EOL]	2018-06-13	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.16.57	2018-06-16	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]

DEFCON GROUP 0531
HACKER COMMUNITY

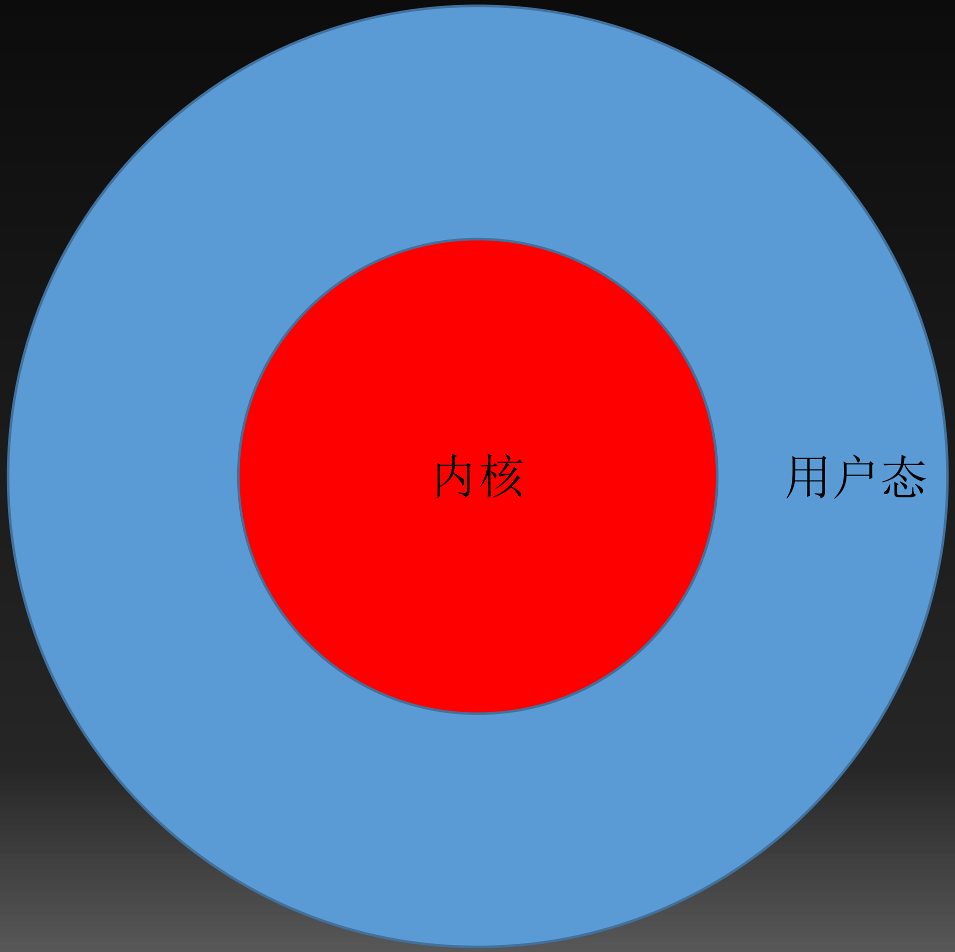
PANGU TEAM

Android

- 开源
  - 内核采用Linux(ion,ashmem...)，外加SoC厂商代码
  - AOSP+(Linux) = Android
  - 高通，海思，三星，博通，英伟达，mtk，mstar ......
- https://android.googlesource.com
  - androidxref.com

内核

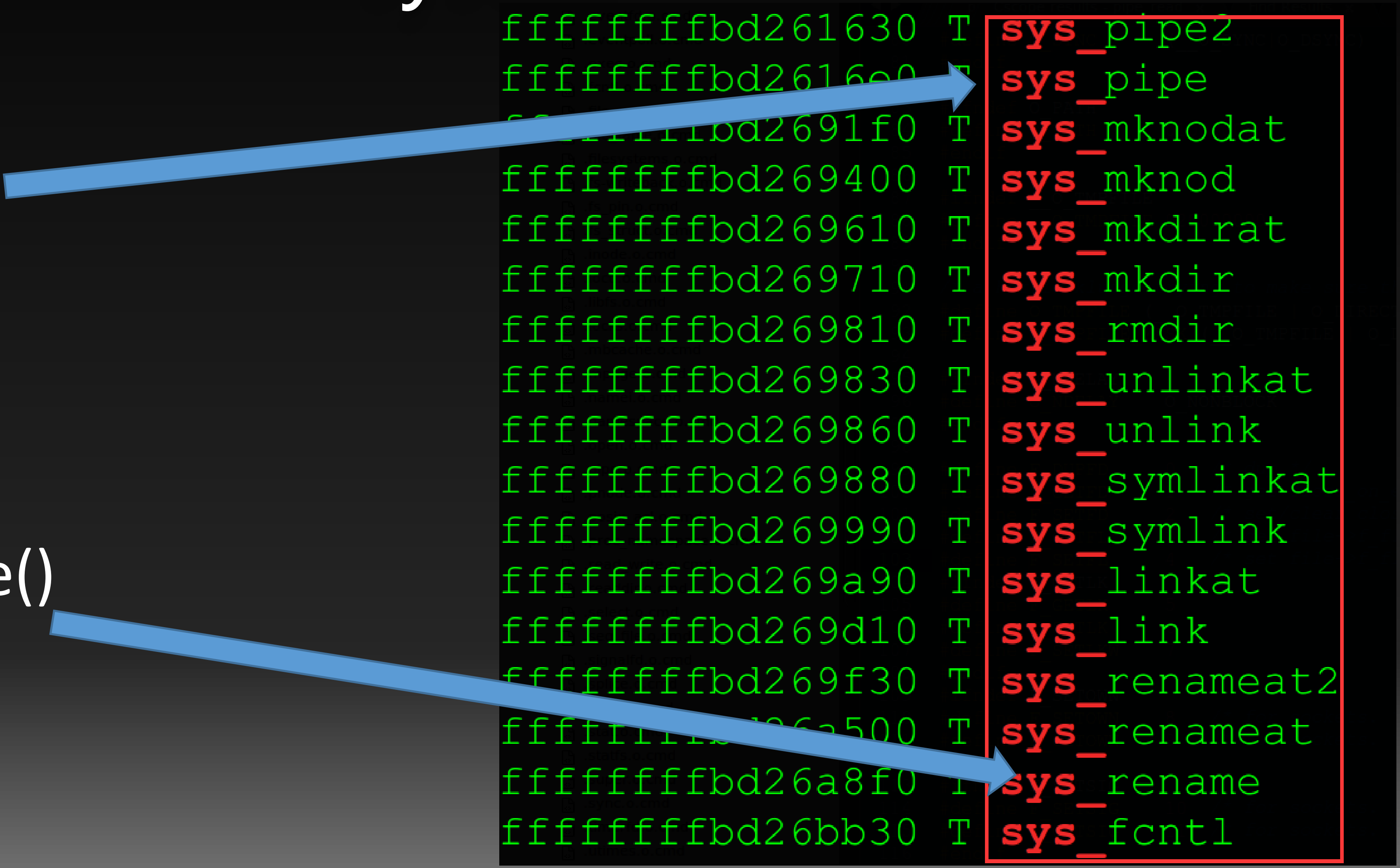
- 内核管理硬件资源
  - CPU资源、存储资源、I/O资源等
- 内核为用户态提供服务: syscall()
  - fs
  - net
  - ptrace
  - ……



syscall

pipe()

rename()



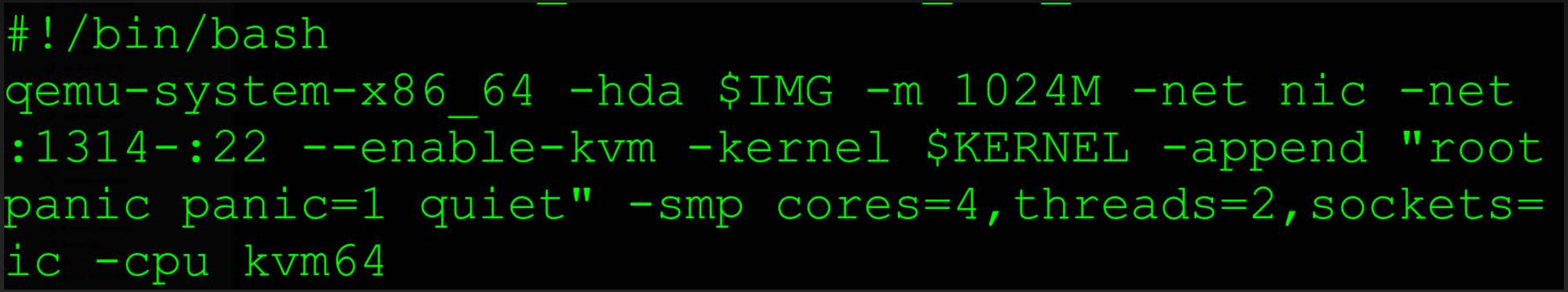
```
ffffffffbd261630 T sys_pipe2
ffffffffbd2616e0 T sys_pipe
ffffffffbd2691f0 T sys_mknodat
ffffffffbd269400 T sys_mknod
ffffffffbd269610 T sys_mkdirat
ffffffffbd269710 T sys_mkdir
ffffffffbd269810 T sys_rmdir
ffffffffbd269830 T sys_unlinkat
ffffffffbd269860 T sys_unlink
ffffffffbd269880 T sys_symlinkat
ffffffffbd269990 T sys_symlink
ffffffffbd269a90 T sys_linkat
ffffffffbd269d10 T sys_link
ffffffffbd269f30 T sys_renameat2
ffffffffbd26a500 T sys_renameat
ffffffffbd26a8f0 T sys_rename
ffffffffbd26bb30 T sys_fcntl
```

环境搭建

- git clone
  - https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git
  - git tags、checkout 、git stash
- 编译

环境搭建

- qemu运行内核



```
#!/bin/bash
qemu-system-x86_64 -hda $IMG -m 1024M -net nic -net
:1314-:22 --enable-kvm -kernel $KERNEL -append "root
panic panic=1 quiet" -smp cores=4,threads=2,sockets=
ic -cpu kvm64
```

环境搭建

- git clone
  - msm 、华为、mtk、三星 .....
  - goldfish !
- 编译
  - ndk toolchain、aarch64

环境搭建

- emulator
  - aosp里的build-kernel.sh 脚本（可编译适配的内核）
- 实体机
  - msm ...
  - abootimg 替换内核

调试

- gdb
  - i r、x、b、disas、watch、s/n、si/ni
- kasan、slub_info
- 插log
  - WARN_ON（1）、printk ......

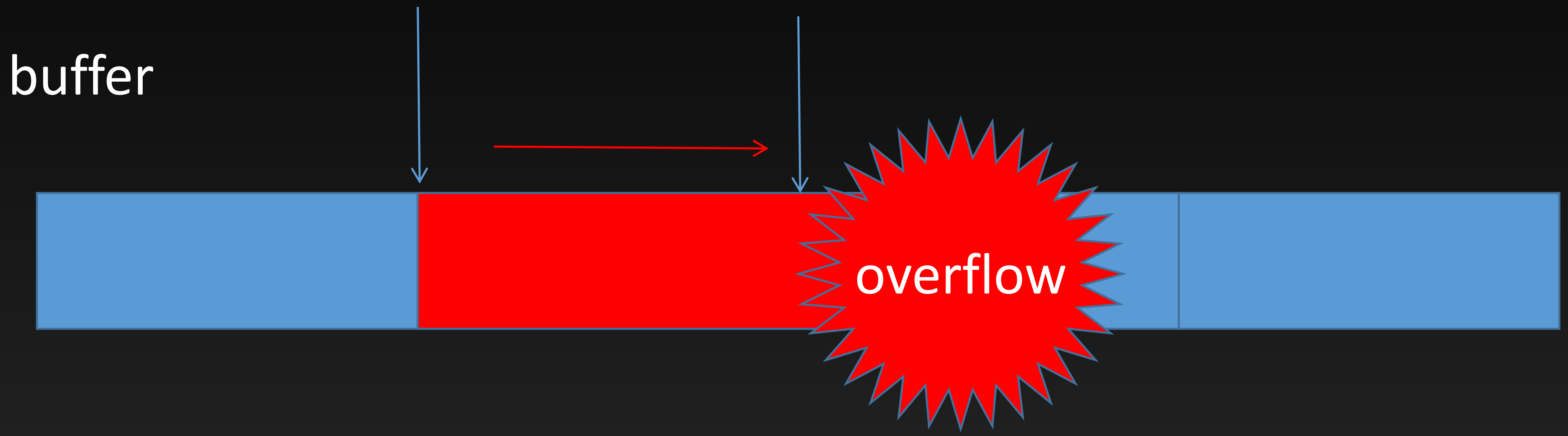
调试

- strace
- ftrace
- 串口

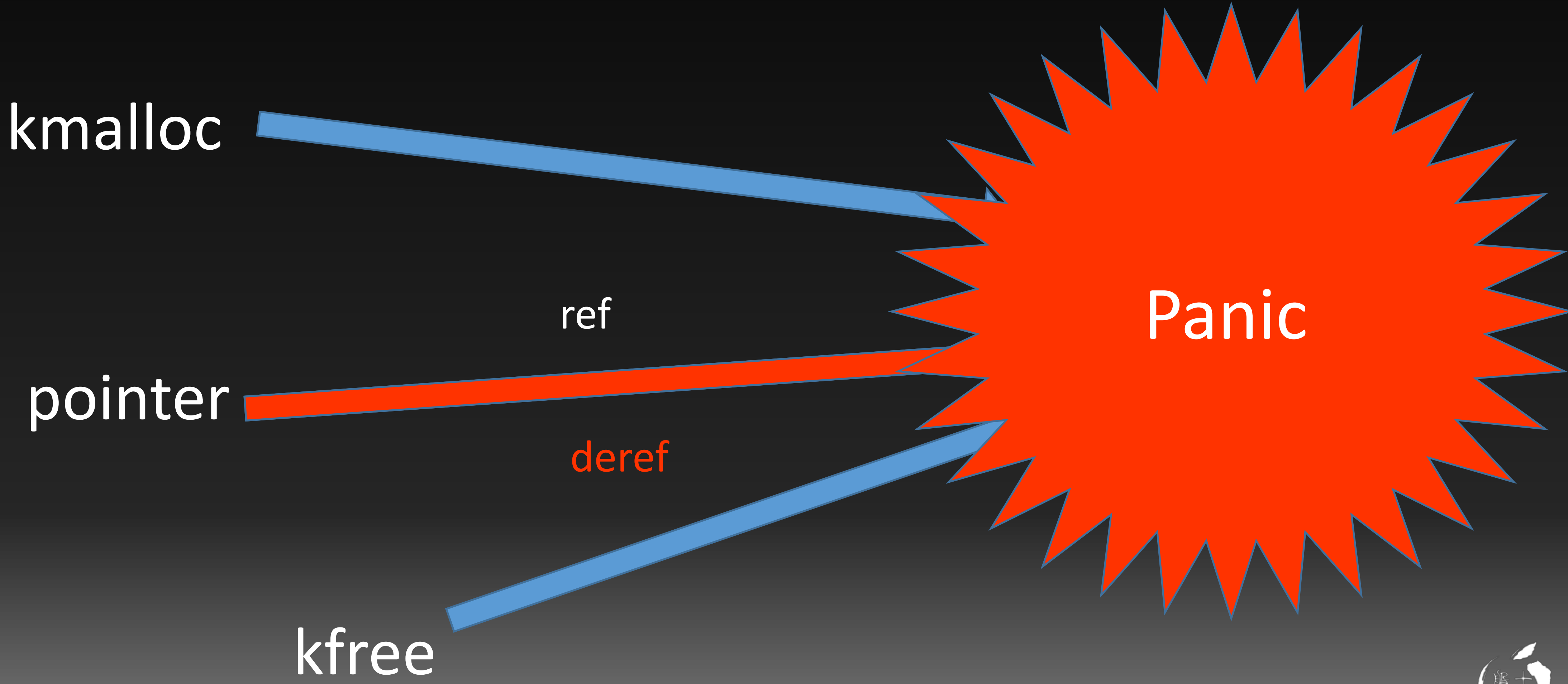
内核漏洞类型

- oob
- stack/heap overflow
- use-after-free
- double free
- race condition ……

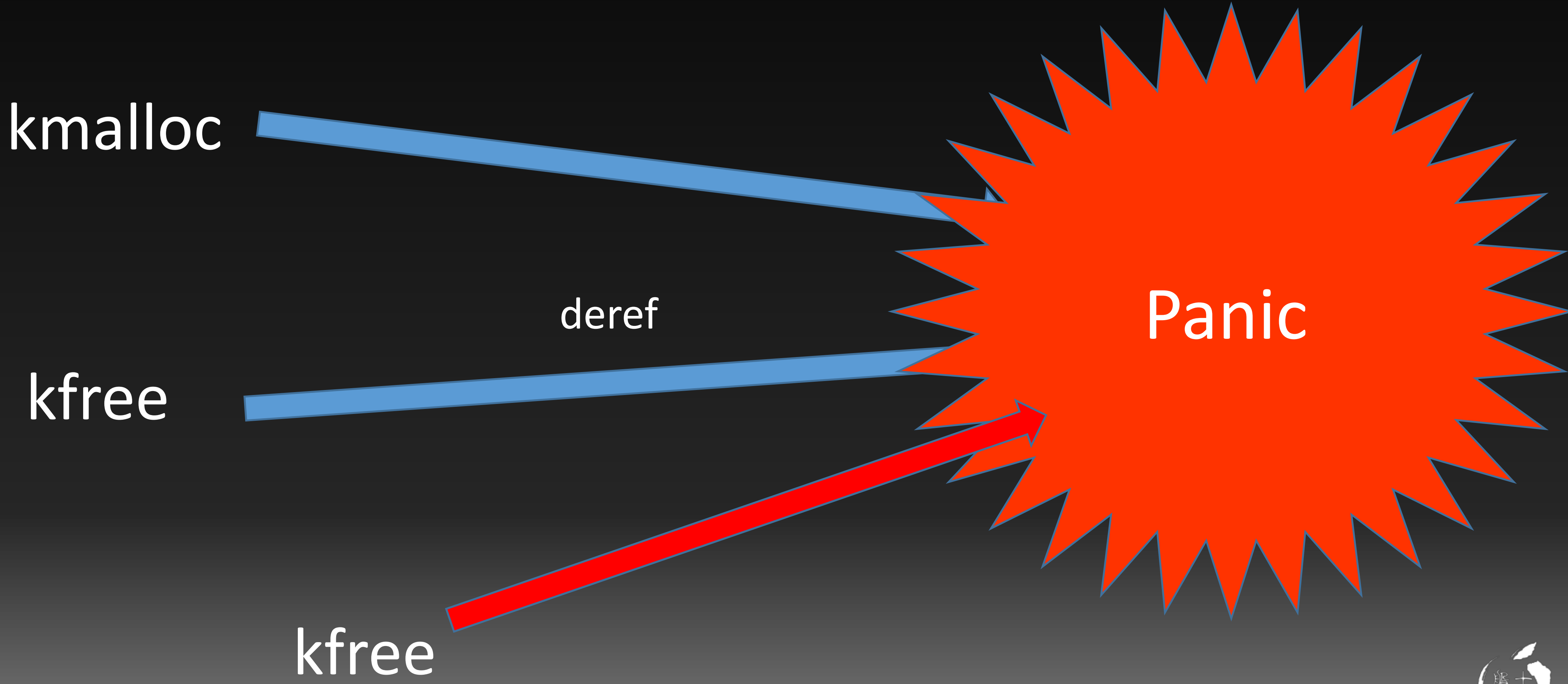
overflow



UAF

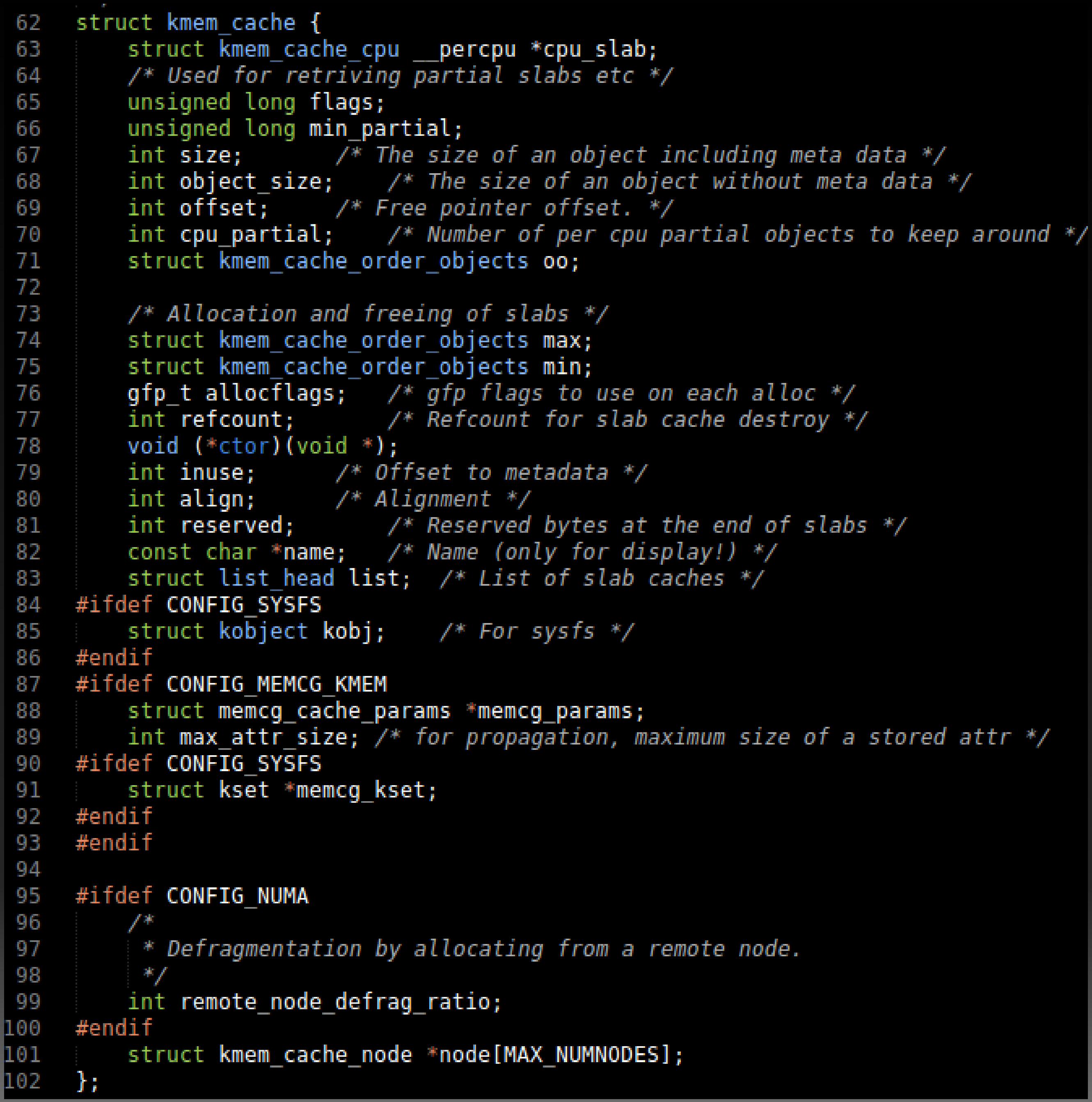


Double Free



SLUB

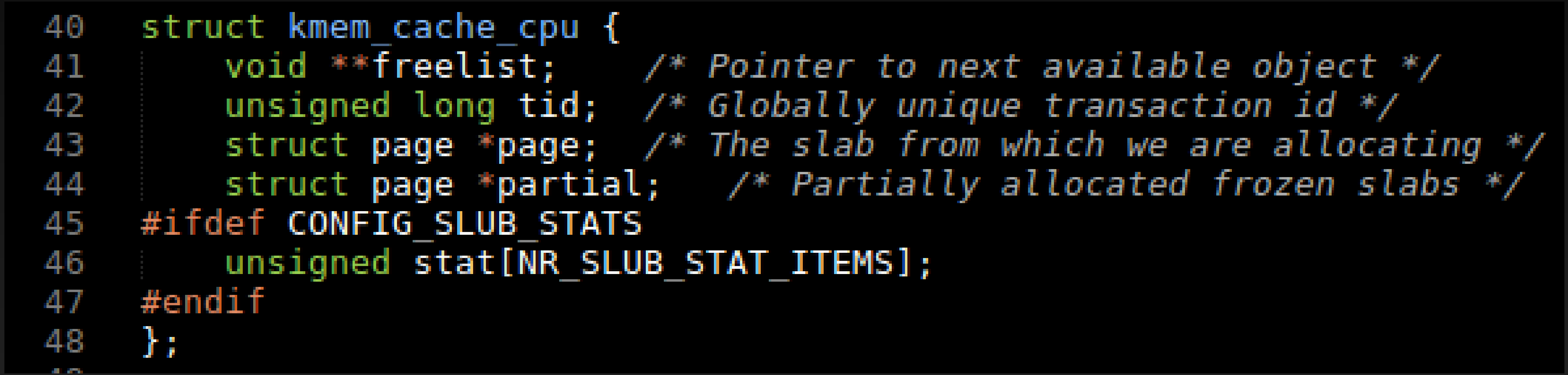
- 由buddy system 提供页支持
  - slob、slab、slub
- slub 里划分为多个obj空间
  - 专为小内存分配而生
  - kmem_cache_create
  - kmalloc-64，kmalloc-128，kmalloc-256 ......



```c
struct kmem_cache {
    struct kmem_cache_cpu __percpu *cpu_slab;
    /* Used for retriving partial slabs etc */
    unsigned long flags;
    unsigned long min_partial;
    int size;       /* The size of an object including meta data */
    int object_size;    /* The size of an object without meta data */
    int offset;     /* Free pointer offset. */
    int cpu_partial;    /* Number of per cpu partial objects to keep around */
    struct kmem_cache_order_objects oo;

    /* Allocation and freeing of slabs */
    struct kmem_cache_order_objects max;
    struct kmem_cache_order_objects min;
    gfp_t allocflags;   /* gfp flags to use on each alloc */
    int refcount;       /* Refcount for slab cache destroy */
    void (*ctor)(void *);
    int inuse;      /* Offset to metadata */
    int align;      /* Alignment */
    int reserved;       /* Reserved bytes at the end of slabs */
    const char *name;   /* Name (only for display!) */
    struct list_head list;  /* List of slab caches */
#ifdef CONFIG_SYSFS
    struct kobject kobj;    /* For sysfs */
#endif
#ifdef CONFIG_MEMCG_KMEM
    struct memcg_cache_params *memcg_params;
    int max_attr_size; /* for propagation, maximum size of a stored attr */
#ifdef CONFIG_SYSFS
    struct kset *memcg_kset;
#endif
#endif

#ifdef CONFIG_NUMA
    /*
     * Defragmentation by allocating from a remote node.
     */
    int remote_node_defrag_ratio;
#endif
    struct kmem_cache_node *node[MAX_NUMNODES];
};
```

一个per cpu变量，对于每个cpu来说，相当于一个本地内存缓存池。当分配内存的时候优先从本地cpu分配内存以保证cache的命中率

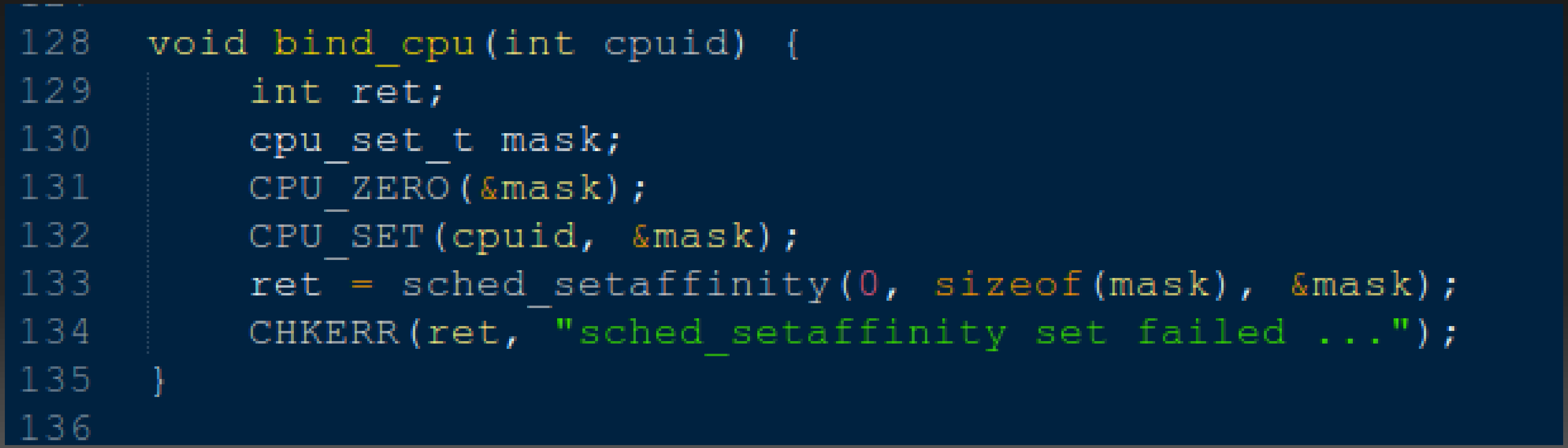
freelist



```
struct kmem_cache_cpu {
	void **freelist;	/* Pointer to next available object */
	unsigned long tid;	/* Globally unique transaction id */
	struct page *page;	/* The slab from which we are allocating */
	struct page *partial;	/* Partially allocated frozen slabs */
#ifdef CONFIG_SLUB_STATS
	unsigned stat[NR_SLUB_STAT_ITEMS];
#endif
};
```

SLUB

- 故而，slab内存块的释放和CPU id息息相关
  - refill或者shape heap时，exp的某些线程注意绑定某CPU id



```
void bind_cpu(int cpuid) {
    int ret;
    cpu_set_t mask;
    CPU_ZERO(&mask);
    CPU_SET(cpuid, &mask);
    ret = sched_setaffinity(0, sizeof(mask), &mask);
    CHKERR(ret, "sched_setaffinity set failed ...");
}
```

缓解措施

- Linux
  - smep、smap、no namespace、kaslr ......
- Android
  - selinux、pxn、pan、kaslr ......

内核 Heap Spray

Heap Spray

- uaf/double-free、race
  - 有时利用需要refill obj
- 堆溢出/泄露类
  - shape heap/fengshui

UAF / Double Free

- uaf/double free
  - CVE-2015-3636
  - CVE-2017-0403
  - CVE-2016-6787
  - CVE-2017-8824
  - CVE-2017-8890
  - CVE-2017-17053
  - CVE-2017-6074 ……

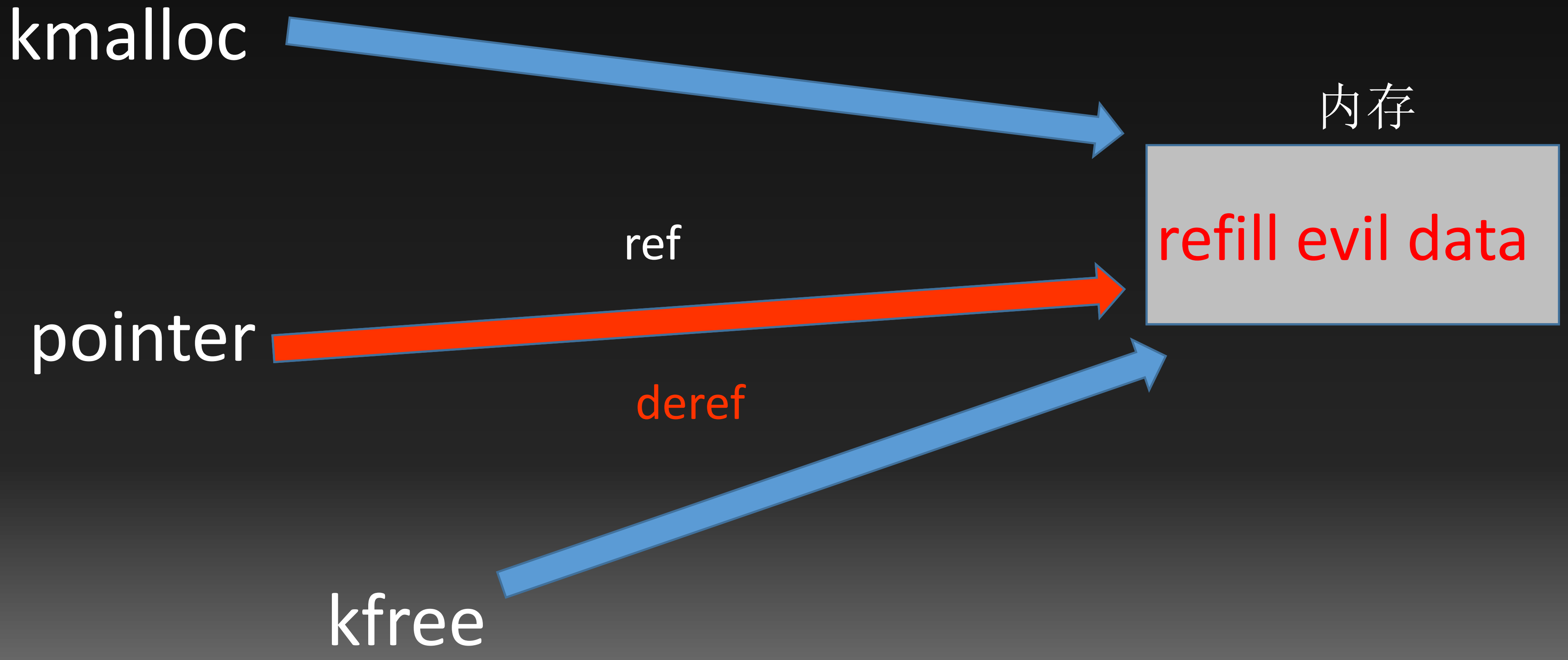
Race Condition

- race condition
  - CVE-2016-1805
  - CVE-2017-10661
  - CVE-2017-7533 (heap overflow or UAF)
  - ……

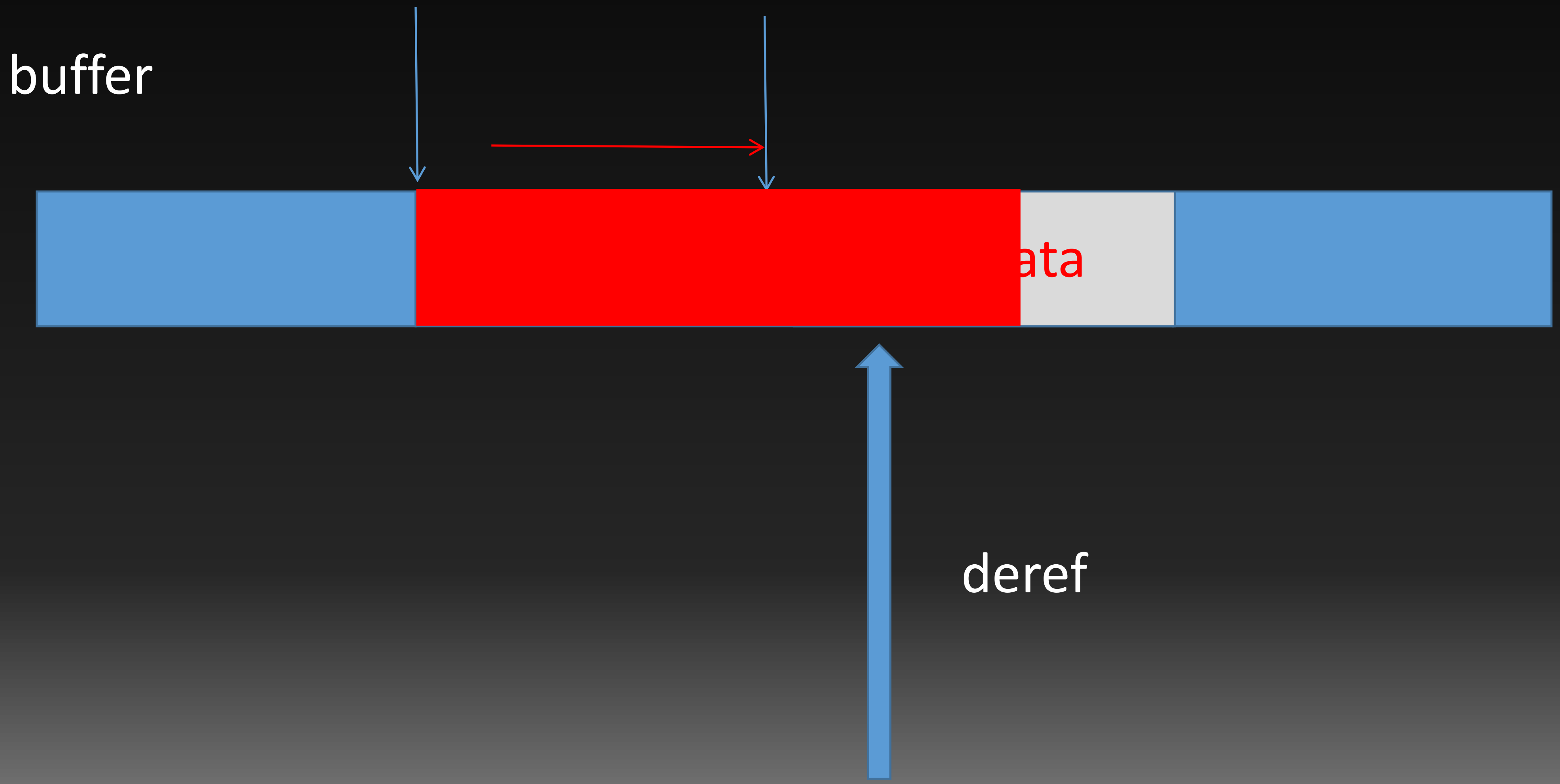
DEFCON GROUP 0531 HACKER COMMUNITY

PANGU TEAM

数据流劫持(refill)

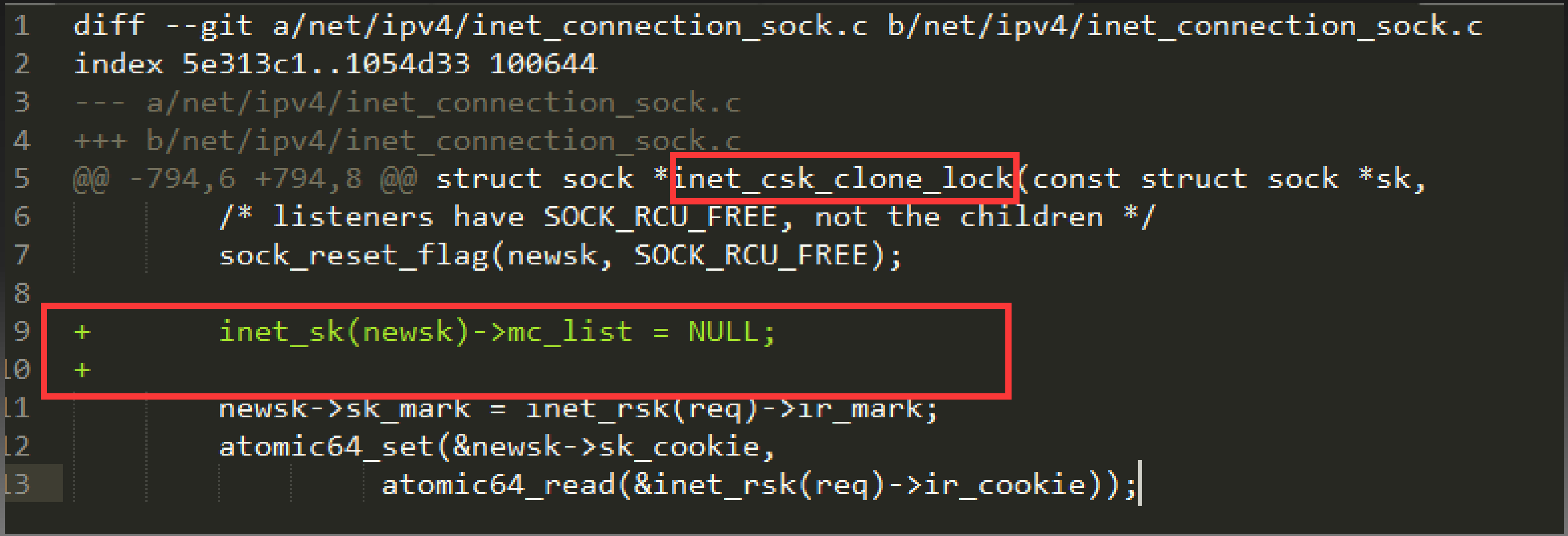


数据流劫持(shape heap)



实例分析1

- Double free: CVE-2017-8890
  - 指针残留导致的UAF

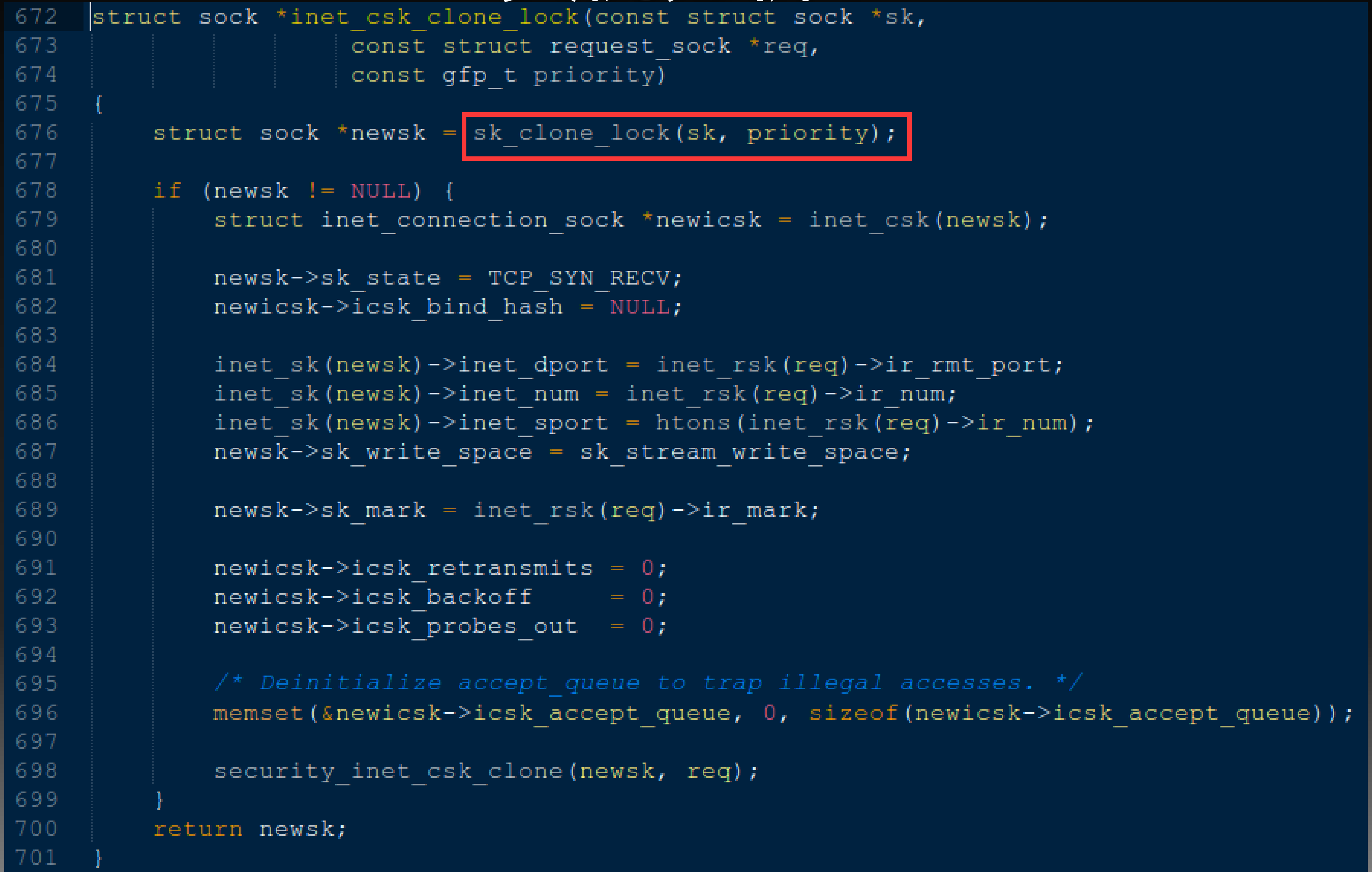


```
diff --git a/net/ipv4/inet_connection_sock.c b/net/ipv4/inet_connection_sock.c
index 5e313c1..1054d33 100644
--- a/net/ipv4/inet_connection_sock.c
+++ b/net/ipv4/inet_connection_sock.c
@@ -794,6 +794,8 @@ struct sock *inet_csk_clone_lock(const struct sock *sk,
		/* listeners have SOCK_RCU_FREE, not the children */
		sock_reset_flag(newsk, SOCK_RCU_FREE);

+		inet_sk(newsk)->mc_list = NULL;
+
		newsk->sk_mark = inet_rsk(req)->ir_mark;
		atomic64_set(&newsk->sk_cookie,
			     atomic64_read(&inet_rsk(req)->ir_cookie));
```

实例分析1

- accept4() -> …… -> inet_csk_clone_lock()-> sock_clone_lock
- 产生一个新的sock

实例分析1

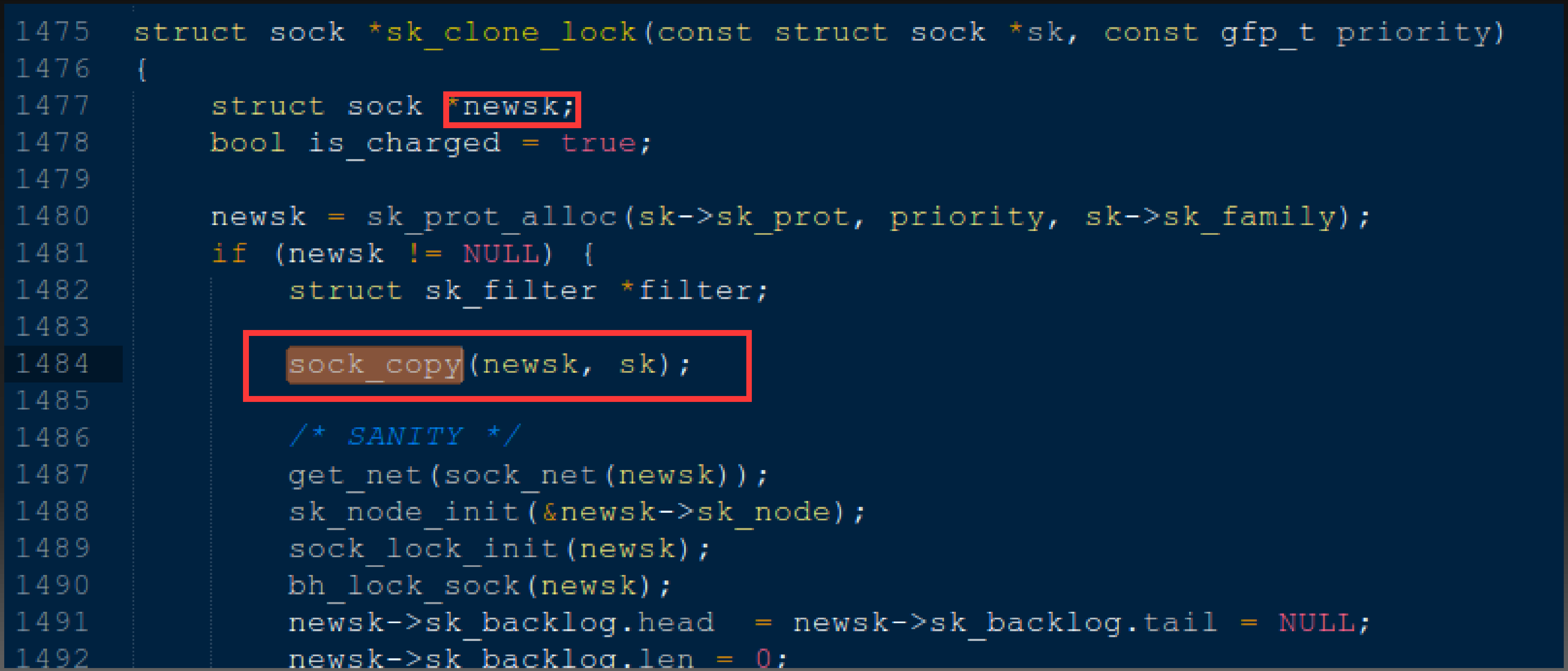


```
struct sock *inet_csk_clone_lock(const struct sock *sk,
                                 const struct request_sock *req,
                                 const gfp_t priority)
{
        struct sock *newsk = sk_clone_lock(sk, priority);

        if (newsk != NULL) {
                struct inet_connection_sock *newicsk = inet_csk(newsk);

                newsk->sk_state = TCP_SYN_RECV;
                newicsk->icsk_bind_hash = NULL;

                inet_sk(newsk)->inet_dport = inet_rsk(req)->ir_rmt_port;
                inet_sk(newsk)->inet_num = inet_rsk(req)->ir_num;
                inet_sk(newsk)->inet_sport = htons(inet_rsk(req)->ir_num);
                newsk->sk_write_space = sk_stream_write_space;

                newsk->sk_mark = inet_rsk(req)->ir_mark;

                newicsk->icsk_retransmits = 0;
                newicsk->icsk_backoff     = 0;
                newicsk->icsk_probes_out  = 0;

                /* Deinitialize accept_queue to trap illegal accesses. */
                memset(&newicsk->icsk_accept_queue, 0, sizeof(newicsk->icsk_accept_queue));

                security_inet_csk_clone(newsk, req);
        }
        return newsk;
}
```

DEFCON GROUP 0531 HACKER COMMUNITY

PANGU TEAM

实例分析1



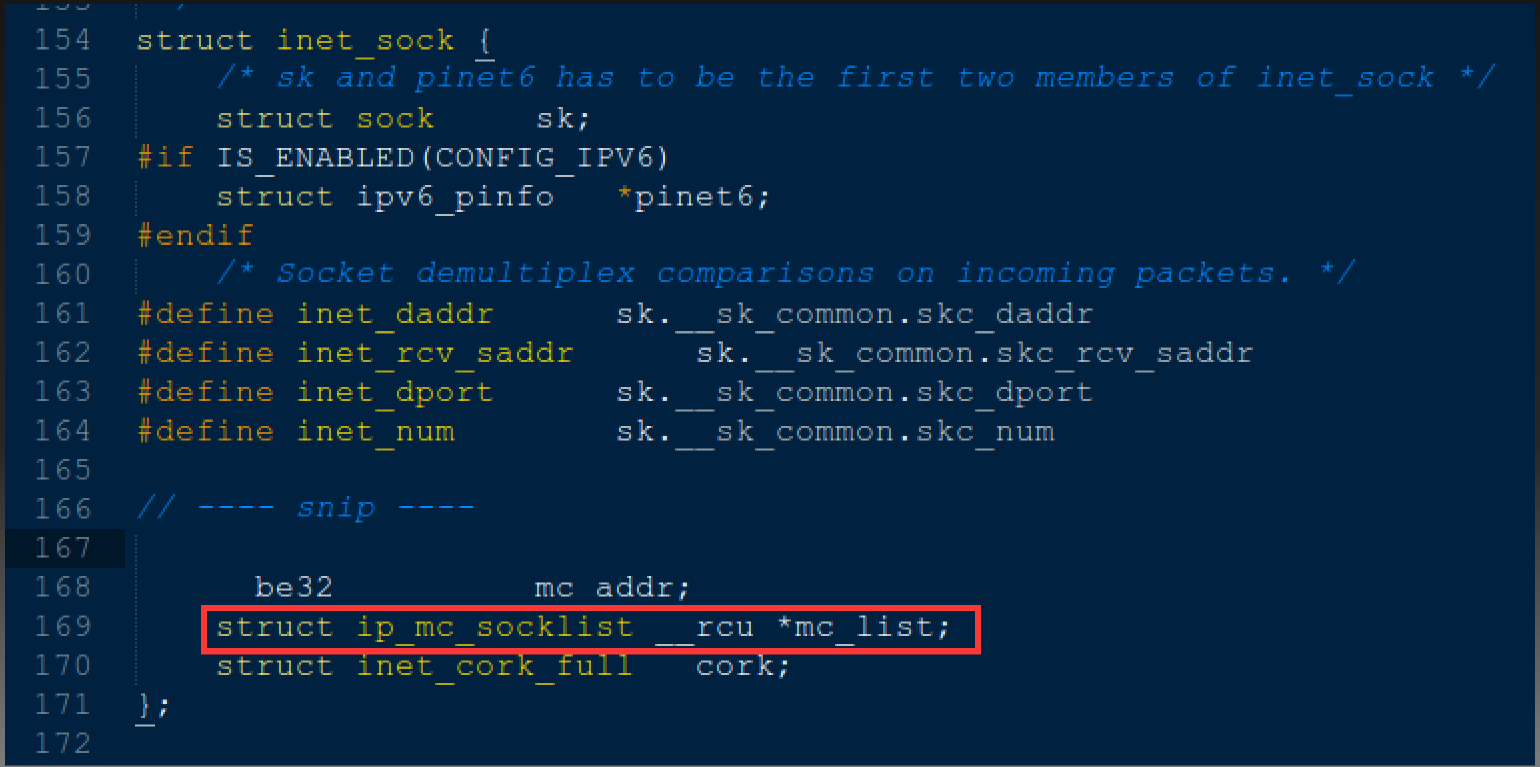
```
struct sock *sk_clone_lock(const struct sock *sk, const gfp_t priority)
{
    struct sock *newsk;
    bool is_charged = true;

    newsk = sk_prot_alloc(sk->sk_prot, priority, sk->sk_family);
    if (newsk != NULL) {
        struct sk_filter *filter;

        sock_copy(newsk, sk);

        /* SANITY */
        get_net(sock_net(newsk));
        sk_node_init(&newsk->sk_node);
        sock_lock_init(newsk);
        bh_lock_sock(newsk);
        newsk->sk_backlog.head  = newsk->sk_backlog.tail = NULL;
        newsk->sk_backlog.len = 0;
```

DEFCON
GROUP 0531
HACKER COMMUNITY

PANGU TEAM

实例分析1

- mc_list 指向同一块内存



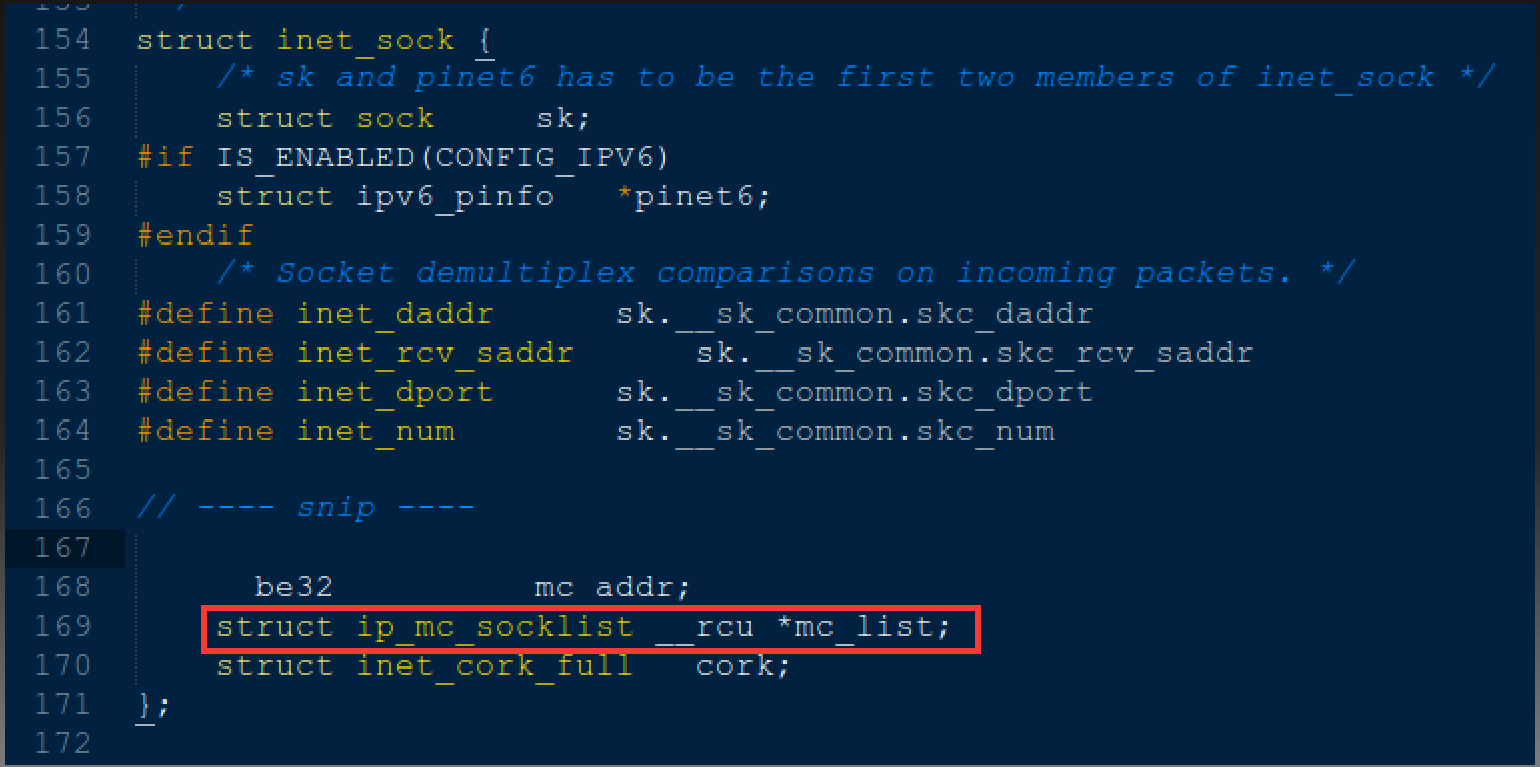
```
struct inet_sock {
	/* sk and pinet6 has to be the first two members of inet_sock */
	struct sock		sk;
#if IS_ENABLED(CONFIG_IPV6)
	struct ipv6_pinfo	*pinet6;
#endif
	/* Socket demultiplex comparisons on incoming packets. */
#define inet_daddr		sk.__sk_common.skc_daddr
#define inet_rcv_saddr		sk.__sk_common.skc_rcv_saddr
#define inet_dport		sk.__sk_common.skc_dport
#define inet_num		sk.__sk_common.skc_num

// ---- snip ----

	 be32			mc addr;
	struct ip_mc_socklist __rcu *mc_list;
	struct inet_cork_full	cork;
};
```

DEFCON GROUP 0531
HACKER COMMUNITY

PANGU TEAM

实例分析1

- mc_list 指向同一块内存，但copy时候，并没有置为NULL



```
struct inet_sock {
	/* sk and pinet6 has to be the first two members of inet_sock */
	struct sock		sk;
#if IS_ENABLED(CONFIG_IPV6)
	struct ipv6_pinfo	*pinet6;
#endif
	/* Socket demultiplex comparisons on incoming packets. */
#define inet_daddr		sk.__sk_common.skc_daddr
#define inet_rcv_saddr		sk.__sk_common.skc_rcv_saddr
#define inet_dport		sk.__sk_common.skc_dport
#define inet_num		sk.__sk_common.skc_num

// ---- snip ----

	 __be32			mc_addr;
	struct ip_mc_socklist __rcu *mc_list;
	struct inet_cork_full	cork;
};
```

实例分析1

- 两个sock最终参与close()时，会发生double free

PoC

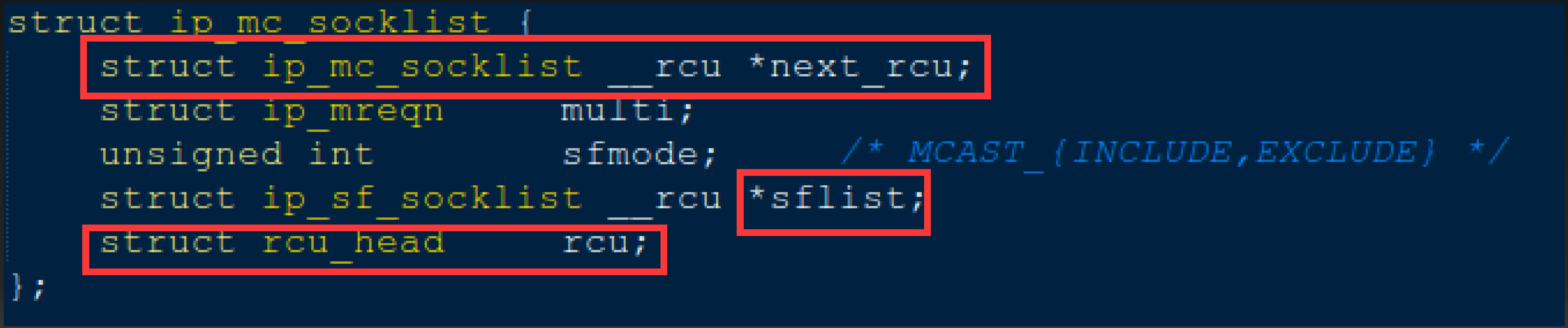
- sockfd = socket(AF_INET, xx, IPPROTO_TCP);
- setsockopt(sockfd, SOL_IP, MCAST_JOIN_GROUP, xxxx, xxxx);
- bind(sockfd, xxxx, xxxx);
- listen(sockfd, xxxx);
- newsockfd = accept(sockfd, xxxx, xxxx); conect ( )
- close(newsockfd) // first free (kfree_rcu)
- sleep(x) // wait rcu free(real free)
- close(sockfd) // double free

DEFCON GROUP 0531 HACKER COMMUNITY

PANGU TEAM

Heap Spray 1

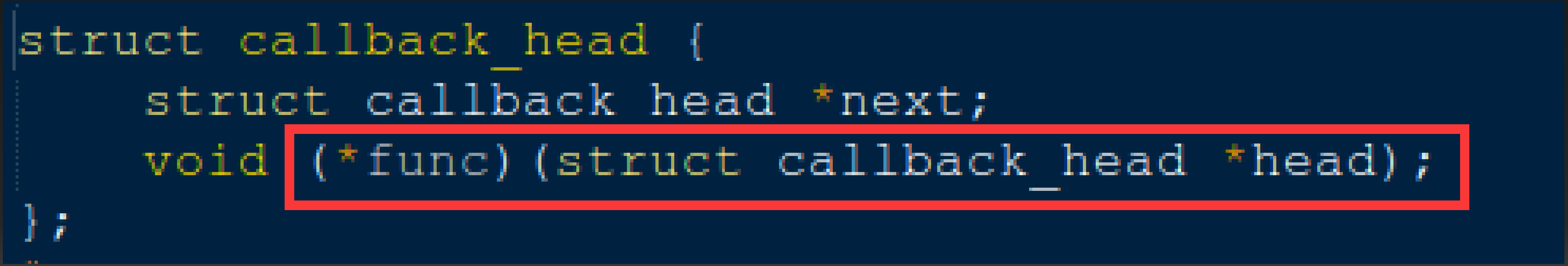
- uaf obj: ip_mc_socklist ——> refill 点 ——> kmalloc-64



```
struct ip_mc_socklist {
    struct ip_mc_socklist __rcu *next_rcu;
    struct ip_mreqn         multi;
    unsigned int            sfmode;        /* MCAST_{INCLUDE,EXCLUDE} */
    struct ip_sf_socklist __rcu *sflist;
    struct rcu_head         rcu;
};
```

Heap Spray 1

- rcu_head
  - 包含函数指针，利于PC指针劫持（rcu_process_callbacks）

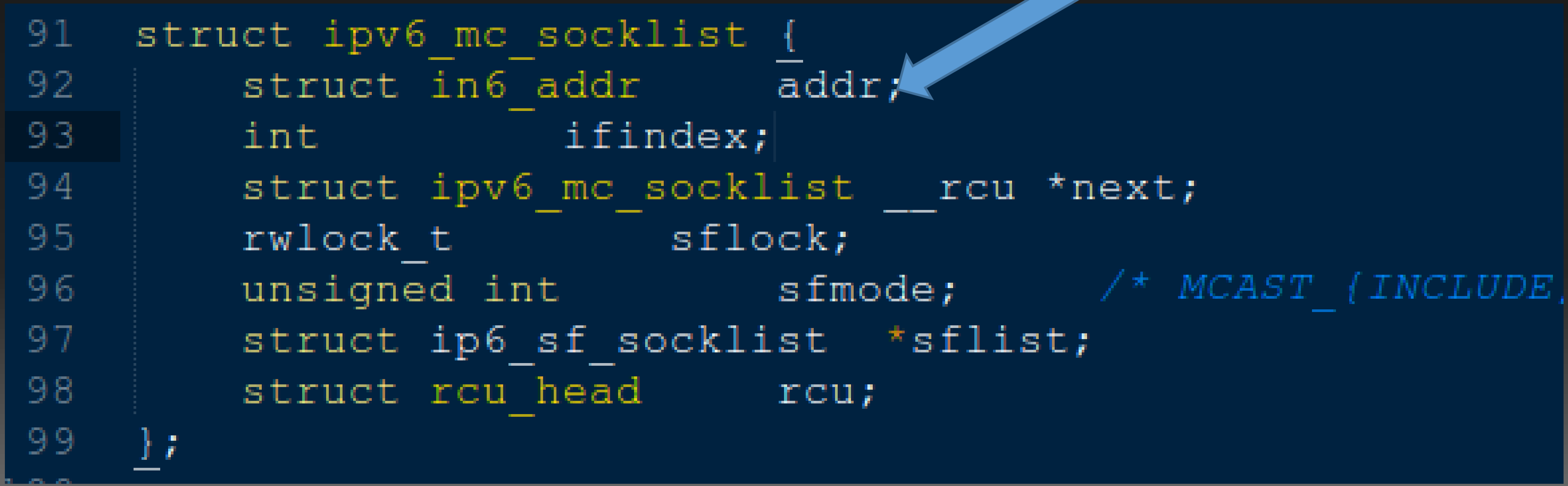


```
struct callback_head {
    struct callback head *next;
    void (*func)(struct callback_head *head);
};
```

Heap Spray 1

- refill obj 的选择
  - 例如: ipv6_mc_socklist   kmalloc-64 !!

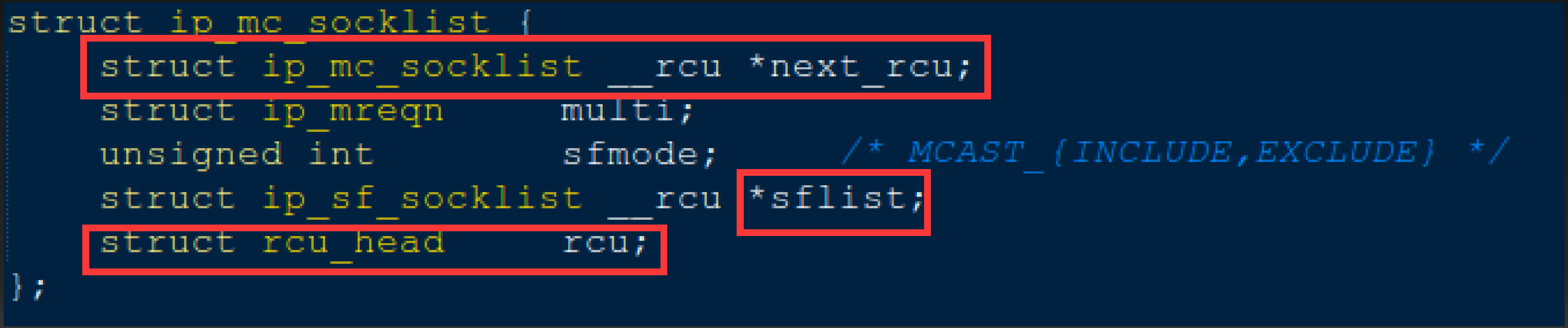
用户态可控!



```
struct ipv6_mc_socklist {
	struct in6_addr		addr;
	int			ifindex;
	struct ipv6_mc_socklist __rcu *next;
	rwlock_t		sflock;
	unsigned int		sfmode;		/* MCAST_{INCLUDE,
	struct ip6_sf_socklist	*sflist;
	struct rcu_head		rcu;
};
```

Heap Spray 1

- ipv6_mc_list 之addr 覆盖 ip_mc_socklist之 next_rcu



```
struct ip_mc_socklist {
	struct ip_mc_socklist __rcu *next_rcu;
	struct ip_mreqn		multi;
	unsigned int		sfmode;		/* MCAST_{INCLUDE,EXCLUDE} */
	struct ip_sf_socklist __rcu *sflist;
	struct rcu_head		rcu;
};
```

DEFCON GROUP 0531 HACKER COMMUNITY

PANGU TEAM

Heap Spray 1

- spray 函数: ipv6 setsockopt ()
  - kmalloc ipv6_mc_socklist

DEFCON GROUP 0531
HACKER COMMUNITY

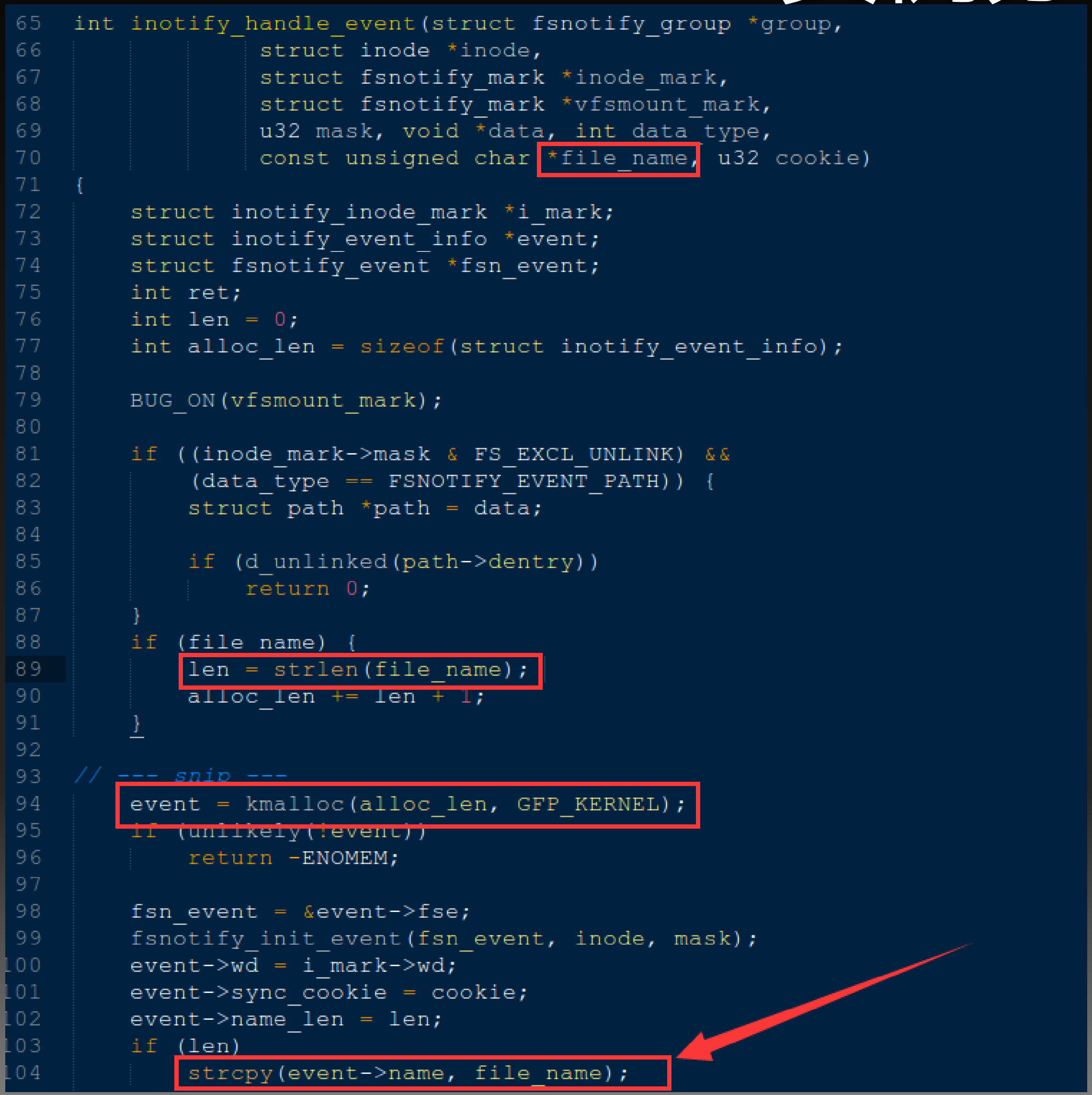
PANGU TEAM

实例分析2

- race导致的UAF和heap overflow：CVE-2017-7533

部分参考:Rooting Android 8 with a Kernel Space Mirroring Attack.pdf from Yong Wang

实例分析2



```c
int inotify_handle_event(struct fsnotify_group *group,
                         struct inode *inode,
                         struct fsnotify_mark *inode_mark,
                         struct fsnotify_mark *vfsmount_mark,
                         u32 mask, void *data, int data_type,
                         const unsigned char *file_name, u32 cookie)
{
    struct inotify_inode_mark *i_mark;
    struct inotify_event_info *event;
    struct fsnotify_event *fsn_event;
    int ret;
    int len = 0;
    int alloc_len = sizeof(struct inotify_event_info);

    BUG_ON(vfsmount_mark);

    if ((inode_mark->mask & FS_EXCL_UNLINK) &&
        (data_type == FSNOTIFY_EVENT_PATH)) {
        struct path *path = data;

        if (d_unlinked(path->dentry))
            return 0;
    }
    if (file_name) {
        len = strlen(file_name);
        alloc_len += len + 1;
    }

// --- snip ---
    event = kmalloc(alloc_len, GFP_KERNEL);
    if (unlikely(!event))
        return -ENOMEM;

    fsn_event = &event->fse;
    fsnotify_init_event(fsn_event, inode, mask);
    event->wd = i_mark->wd;
    event->sync_cookie = cookie;
    event->name_len = len;
    if (len)
        strcpy(event->name, file_name);
```

strcpy(event->name,file_name)

file_name 可在其他线程用 reame 系统调用进行扩大名字长度从而导致溢出！

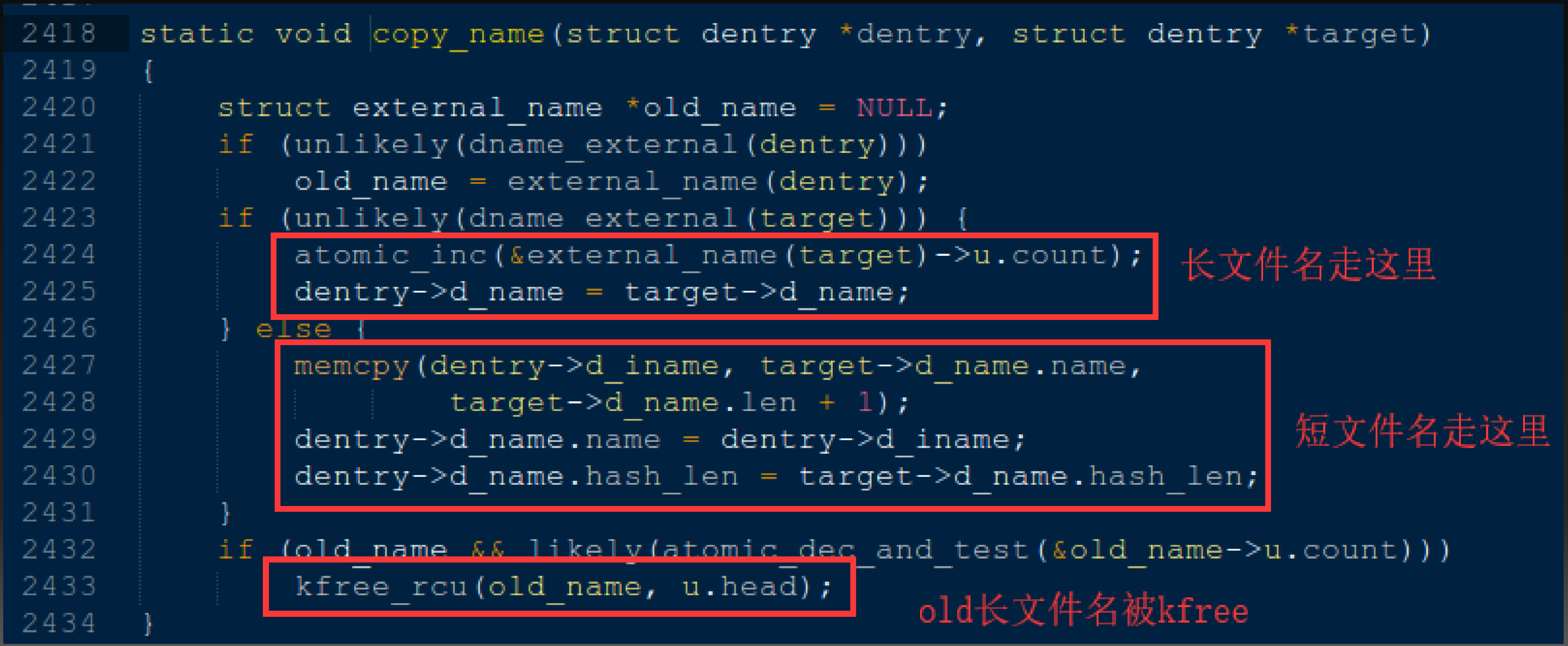
DEFCON GROUP 0531
HACKER COMMUNITY

PANGU TEAM

实例分析2

- 但是有两种情况…
  - 短文件名触发 -> heap overflow
  - 长文件名触发 -> uaf

实例分析2



```
static void copy_name(struct dentry *dentry, struct dentry *target)
{
	struct external_name *old_name = NULL;
	if (unlikely(dname_external(dentry)))
		old_name = external_name(dentry);
	if (unlikely(dname_external(target))) {
		atomic_inc(&external_name(target)->u.count);
		dentry->d_name = target->d_name;
	} else {
		memcpy(dentry->d_iname, target->d_name.name,
				target->d_name.len + 1);
		dentry->d_name.name = dentry->d_iname;
		dentry->d_name.hash_len = target->d_name.hash_len;
	}
	if (old_name && likely(atomic_dec_and_test(&old_name->u.count)))
		kfree_rcu(old_name, u.head);
}
```

长文件名走这里

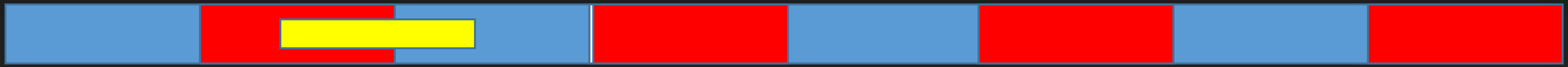
短文件名走这里

old长文件名被kfree

Heap Spray 2

- 短文件名方式(目前很多机器不能用?)
  - 故技新用 ： spray ipv6_mc_socklist

1. setsockopt() -> spray mc_list
2. 间隔close() -> free mc_list
3. open() -> fill hole



4. rename() -> overflow

kmalloc-64

Heap Spray 3

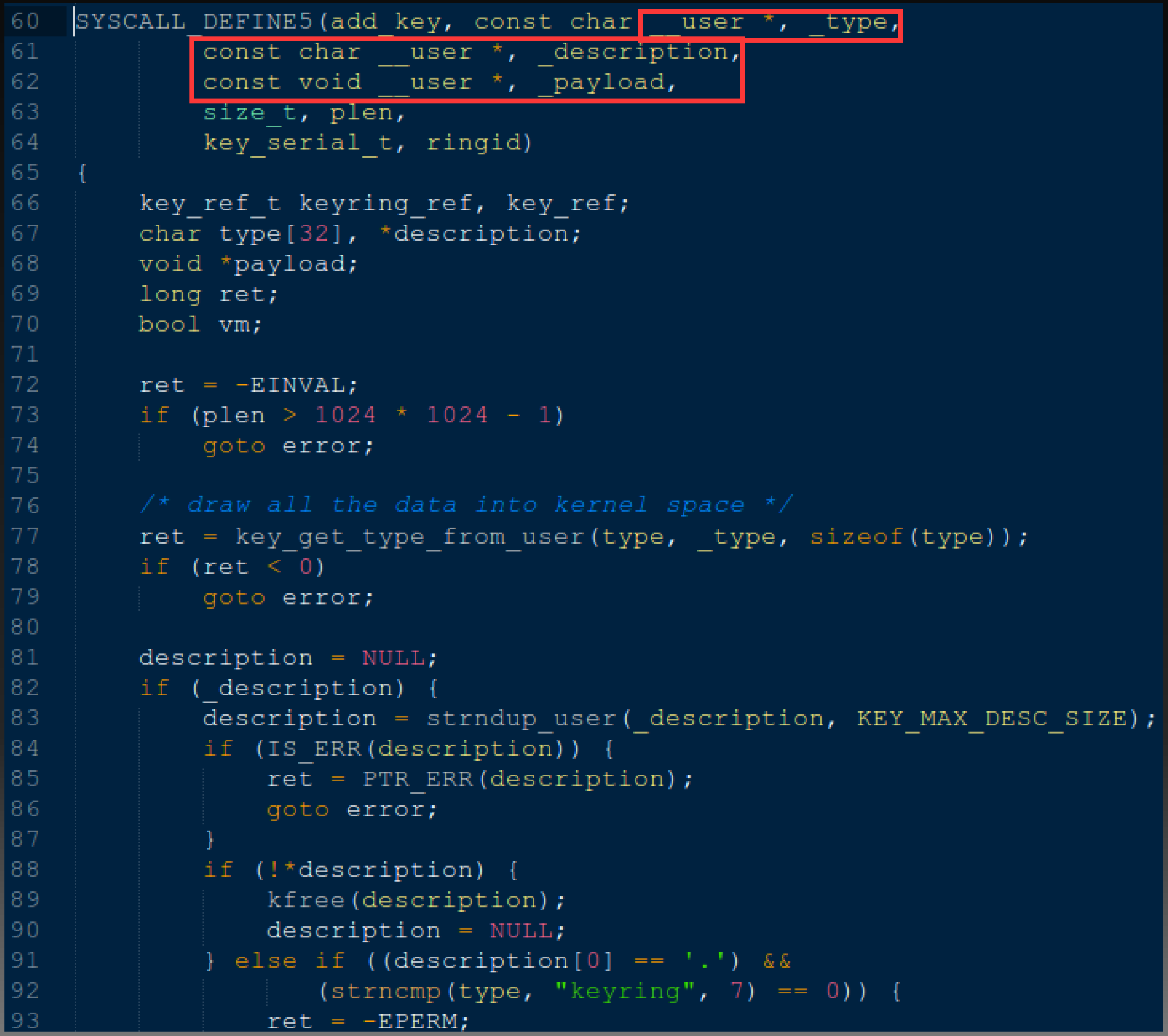
- 长文件名方式
  - 重点: 如何refill file_name 所在内存，并最终扩大name长度呢
    - 大量线程进程race+spray

Heap Spray 3

- Spray 函数:长度和内容都可控，此为佳
  - send , sendmsg ......
  - add_key（android已经不能用此方法了）

Heap Spray 3

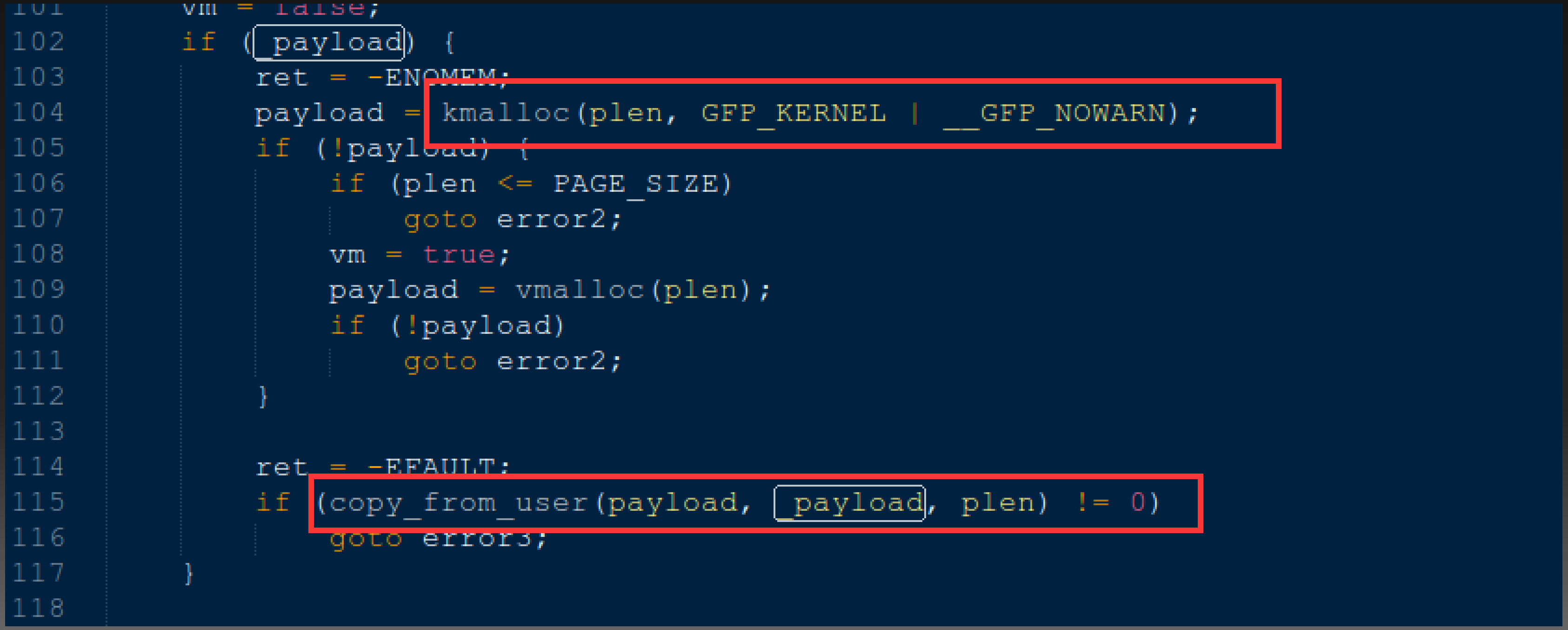
- 探讨add_key



```c
SYSCALL_DEFINE5(add_key, const char __user *, _type,
        const char __user *, _description,
        const void __user *, _payload,
        size_t, plen,
        key_serial_t, ringid)
{
    key_ref_t keyring_ref, key_ref;
    char type[32], *description;
    void *payload;
    long ret;
    bool vm;

    ret = -EINVAL;
    if (plen > 1024 * 1024 - 1)
        goto error;

    /* draw all the data into kernel space */
    ret = key_get_type_from_user(type, _type, sizeof(type));
    if (ret < 0)
        goto error;

    description = NULL;
    if (_description) {
        description = strndup_user(_description, KEY_MAX_DESC_SIZE);
        if (IS_ERR(description)) {
            ret = PTR_ERR(description);
            goto error;
        }
        if (!*description) {
            kfree(description);
            description = NULL;
        } else if ((description[0] == '.') &&
               (strncmp(type, "keyring", 7) == 0)) {
            ret = -EPERM;
```

Heap Spray 3

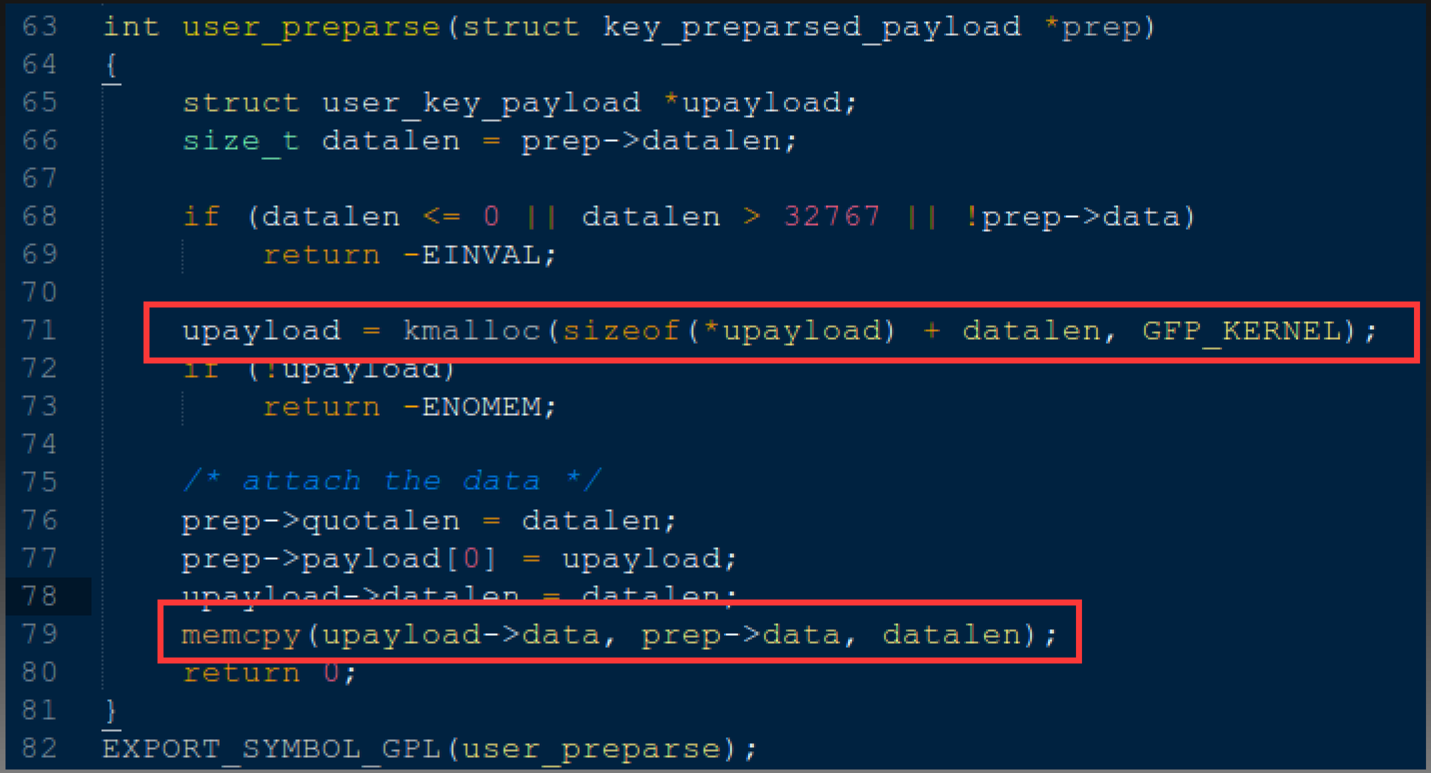
- spray payload 1



```
    vm = false;
    if (_payload) {
        ret = -ENOMEM;
        payload = kmalloc(plen, GFP_KERNEL | __GFP_NOWARN);
        if (!payload) {
            if (plen <= PAGE_SIZE)
                goto error2;
            vm = true;
            payload = vmalloc(plen);
            if (!payload)
                goto error2;
        }

        ret = -EFAULT;
        if (copy_from_user(payload, _payload, plen) != 0)
            goto error3;
    }
```

Heap Spray 3

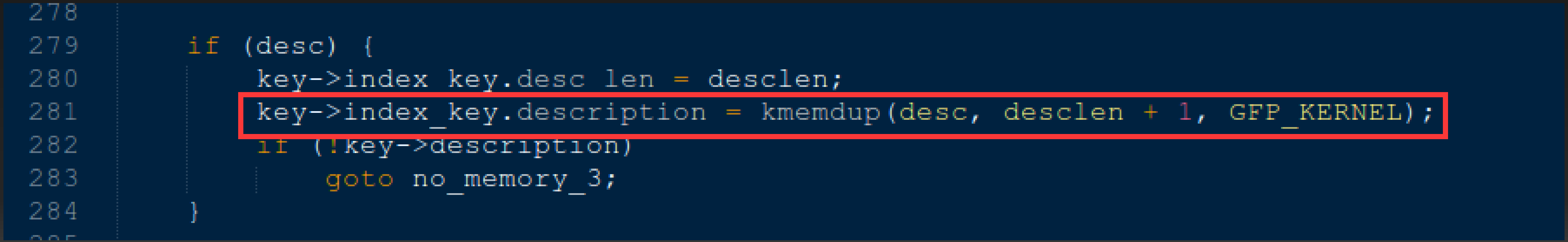
- spray payload 2 （18 bytes header+总量限制）



```c
int user_preparse(struct key_preparsed_payload *prep)
{
    struct user_key_payload *upayload;
    size_t datalen = prep->datalen;

    if (datalen <= 0 || datalen > 32767 || !prep->data)
        return -EINVAL;

    upayload = kmalloc(sizeof(*upayload) + datalen, GFP_KERNEL);
    if (!upayload)
        return -ENOMEM;

    /* attach the data */
    prep->quotalen = datalen;
    prep->payload[0] = upayload;
    upayload->datalen = datalen;
    memcpy(upayload->data, prep->data, datalen);
    return 0;
}
EXPORT_SYMBOL_GPL(user_preparse);
```

Heap Spray 3

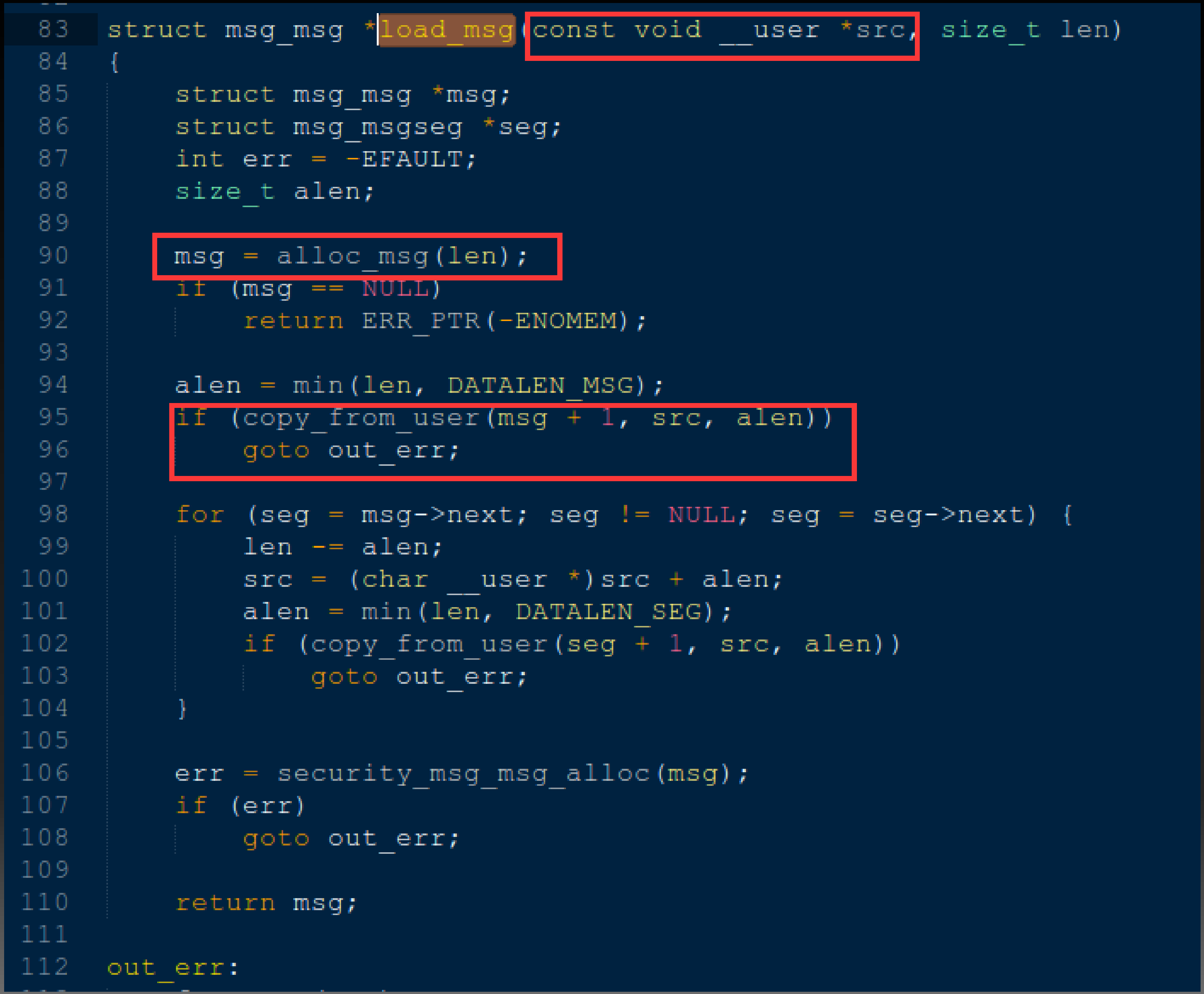
- spray description （较好）



```
    if (desc) {
        key->index_key.desc_len = desclen;
        key->index_key.description = kmemdup(desc, desclen + 1, GFP_KERNEL);
        if (!key->description)
            goto no_memory_3;
    }
```

Heap Spray 3

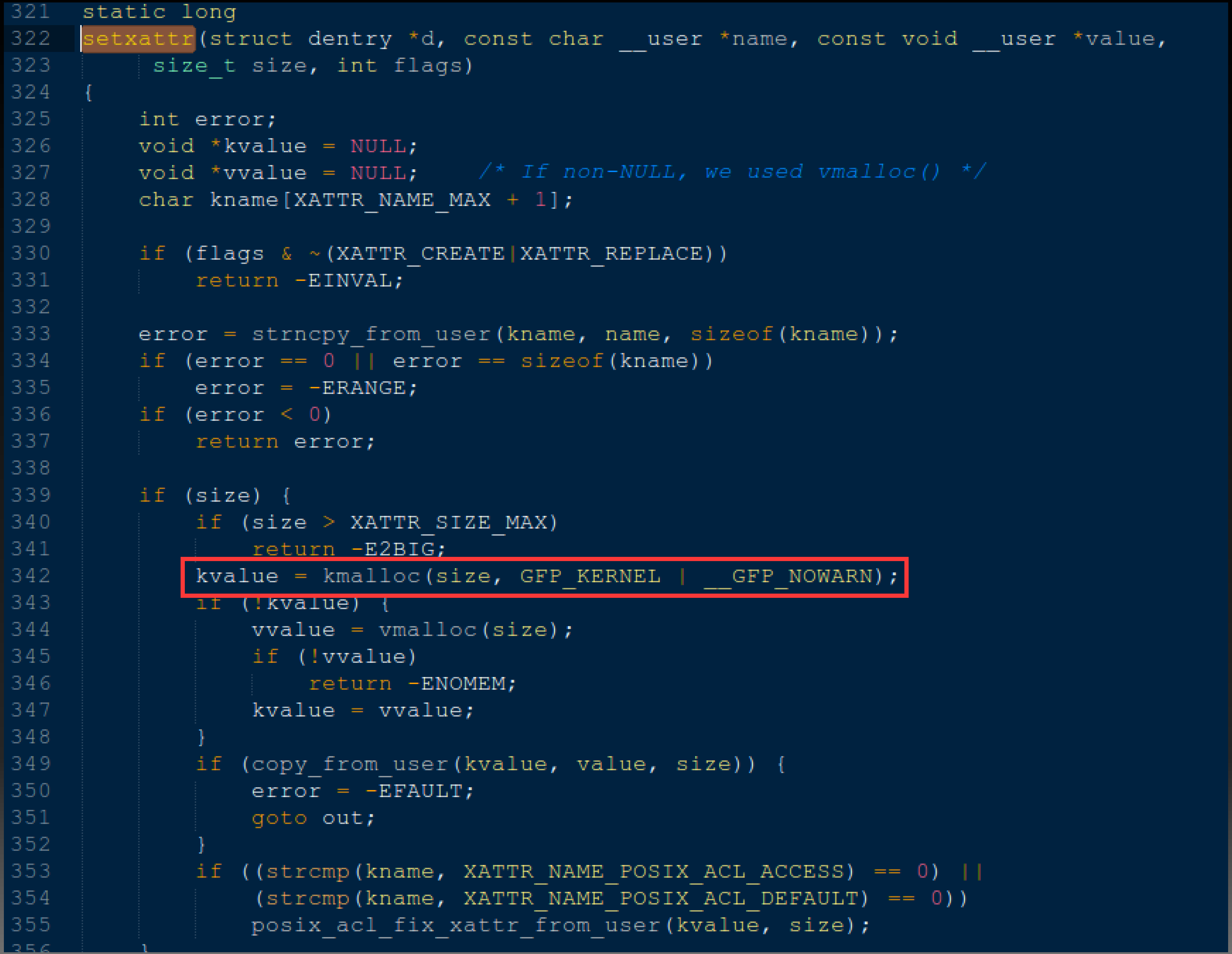
- 探讨其他的spray方法
  - msgsnd
  - 但是不利于CVE-2017-7533，可控数据前有可能有'\x00'截断
  - 也是比较好的spray函数



```
struct msg_msg *load_msg(const void __user *src, size_t len)
{
	struct msg_msg *msg;
	struct msg_msgseg *seg;
	int err = -EFAULT;
	size_t alen;

	msg = alloc_msg(len);
	if (msg == NULL)
		return ERR_PTR(-ENOMEM);

	alen = min(len, DATALEN_MSG);
	if (copy_from_user(msg + 1, src, alen))
		goto out_err;

	for (seg = msg->next; seg != NULL; seg = seg->next) {
		len -= alen;
		src = (char __user *)src + alen;
		alen = min(len, DATALEN_SEG);
		if (copy_from_user(seg + 1, src, alen))
			goto out_err;
	}

	err = security_msg_msg_alloc(msg);
	if (err)
		goto out_err;

	return msg;

out_err:
```

Heap Spray 3

- 探讨其他的spray方法
  - setxattr等设置文件附加属性函数
    - 长度和内容可控
    - 可以通过单个文件或者大量文件进行spray
    - 对于CVE-2017-7533 race下的占位，效果也比较好



```c
static long
setxattr(struct dentry *d, const char __user *name, const void __user *value,
         size_t size, int flags)
{
    int error;
    void *kvalue = NULL;
    void *vvalue = NULL;    /* If non-NULL, we used vmalloc() */
    char kname[XATTR_NAME_MAX + 1];

    if (flags & ~(XATTR_CREATE|XATTR_REPLACE))
        return -EINVAL;

    error = strncpy_from_user(kname, name, sizeof(kname));
    if (error == 0 || error == sizeof(kname))
        error = -ERANGE;
    if (error < 0)
        return error;

    if (size) {
        if (size > XATTR_SIZE_MAX)
            return -E2BIG;
        kvalue = kmalloc(size, GFP_KERNEL | __GFP_NOWARN);
        if (!kvalue) {
            vvalue = vmalloc(size);
            if (!vvalue)
                return -ENOMEM;
            kvalue = vvalue;
        }
        if (copy_from_user(kvalue, value, size)) {
            error = -EFAULT;
            goto out;
        }
        if ((strcmp(kname, XATTR_NAME_POSIX_ACL_ACCESS) == 0) ||
            (strcmp(kname, XATTR_NAME_POSIX_ACL_DEFAULT) == 0))
            posix_acl_fix_xattr_from_user(kvalue, size);
    }
```

Heap Spray 3

- 探讨其他的spray方法
  - more！
  - 有优劣，视情况而定

Heap Spray 3

- race成功后
  - 采用spray+free iovec的方法产生内存holes，然后填充再溢出
  - 类似前面提到的

参考

- http://www.wowotech.net/memory_management/426.html
- Mirror Mirror: Rooting Android 8 with a Kernel Space Mirroring Attack (Yong Wang)
- https://github.com/hardenedlinux/offensive_poc
- https://github.com/retme7/My-Slides

End

PANGU TEAM