网络安全集成项目——符号执行部分demo解释

对OS实验室负责的四个工具，各举一个用工具发现的真实bug的例子。

Kint

Kint用于检测整数错误。

> CVE-2009-1265:
> Integer overflow in rose_sendmsg (sys/net/af_rose.c) in the Linux kernel 2.6.24.4, and other versions before 2.6.30-rc1, might allow remote attackers to obtain sensitive information via a large length value, which causes "garbage" memory to be sent.

相关代码：

```
// net/rose/af_rose.c
static int rose_sendmsg(struct kiocb *iocb, struct socket *sock,
            struct msghdr *msg, size_t len)
...
    skb_put(skb, len);
```

```
// net/core/skbuff.c

static void skb_over_panic(struct sk_buff *skb, int sz, void *here)
{
    printk(KERN_EMERG "skb_over_panic: text:%p len:%d put:%d head:%p "
              "data:%p tail:%#lx end:%#lx dev:%s\n",
           here, skb->len, sz, skb->head, skb->data,
           (unsigned long)skb->tail, (unsigned long)skb->end,
           skb->dev ? skb->dev->name : "<NULL>");
    BUG();
}

unsigned char *skb_put(struct sk_buff *skb, unsigned int len)
{
    unsigned char *tmp = skb_tail_pointer(skb);
    SKB_LINEAR_ASSERT(skb);
    skb->tail += len;
    skb->len  += len;
```

```
    if (unlikely(skb->tail > skb->end))
        skb_over_panic(skb, len, __builtin_return_address(0));
    return tmp;
}
EXPORT_SYMBOL(skb_put);
```

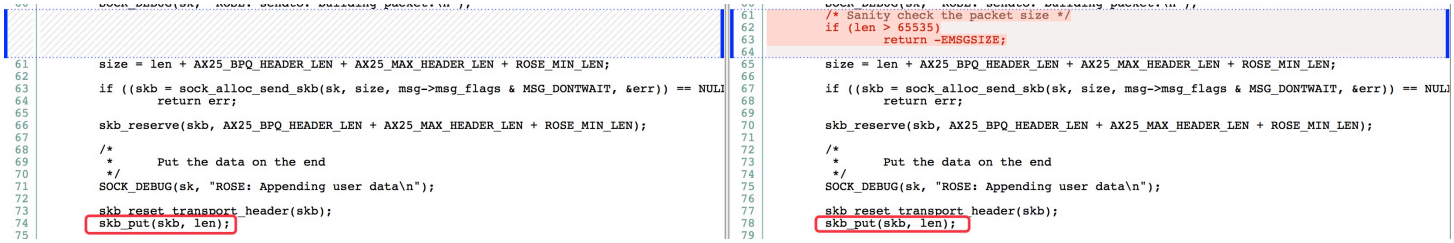
Bug fix:

```
diff --git a/net/rose/af_rose.c b/net/rose/af_rose.c
index 6501396..0f36e8d 100644
--- a/net/rose/af_rose.c
+++ b/net/rose/af_rose.c
@@ -1124,6 +1124,10 @@ static int rose_sendmsg(struct kiocb *iocb, struct so
cket *sock,

    /* Build a packet */
    SOCK_DEBUG(sk, "ROSE: sendto: building packet.\n");
+   /* Sanity check the packet size */
+   if (len > 65535)
+       return -EMSGSIZE;
+
    size = len + AX25_BPQ_HEADER_LEN + AX25_MAX_HEADER_LEN + ROSE_MIN_LEN;

    if ((skb = sock_alloc_send_skb(sk, size, msg->msg_flags & MSG_DONTWAIT,
&err)) == NULL)
```

修改前后对比：



Juxta

Juxta是通过对符合POSIX标准的文件系统进行横向对比，来发现不同文件系统之间对同一个接口的实现行为的不同，进而找出可能存在的漏洞。

例子相关代码：

```
// v4.0-rc2, /fs/gfs2/glock.c:2050
int gfs2_create_debugfs_file(struct gfs2_sbd *sdp)
{
```

```
    sdp->debugfs_dir = debugfs_create_dir(sdp->sd_table_name, gfs2_root);
    if (!sdp->debugfs_dir)
        return -ENOMEM;
```

```
// v4.0-rc2, /fs/ubifs/debug.c:2834
int dbg_debugfs_init_fs(struct ubifs_info *c)
{
...
    fname = d->dfs_dir_name;
    dent = debugfs_create_dir(fname, dfs_rootdir);
    if (IS_ERR_OR_NULL(dent))
        goto out;
    d->dfs_dir = dent;
```

GFS2只检测`debugfs_create_dir()`的返回值是否为NULL，但UBIFS会检测`debugfs_create_dir()`的返回值是否为空或者是一个error code。经核查，如果`DEBUG_FS`选项没有配置的话，`debugfs_create_dir()`会返回`-ENODEV`，所以调用者需要检测返回的错误代码，否则系统可能会崩溃。

APISan

例子相关代码：

```
// drivers/media/usb/pvrusb2/pvrusb2-context.c:194
// in Linux v4.5-rc4g
int pvr2_context_global_init(void)
{
    pvr2_context_thread_ptr = kthread_run(pvr2_context_thread_func,
                          NULL,
                          "pvrusb2-context");
    return (pvr2_context_thread_ptr ? 0 : -ENOMEM);
}
```

`kthread_run()`函数在执行成功时返回一个`task_struct`类型的指针，在执行失败时返回一个error code，`pvr2_context_thread_ptr`永远不为`NULL`，从而导致`return`的值永远为0。

RID

例子相关代码

```
// drivers/gpu/drm/radeon/radeon_display.c
```

```
// in Linux v3.17
static int
radeon_crtc_set_config(struct drm_mode_set *set)
{
...
    ret = pm_runtime_get_sync(dev->dev);
    if (ret < 0)
        return ret;
}
```

这段代码中，调用`pm_runtime_get_sync()`函数后，如果返回值小于0，则直接返回，这是Linux中常见的一种对API specification的误解。开发者会想当然的认为如果返回一个小于0的error code，那么这个函数就没有对refcount进行修改。然而实际上，这个函数不论返回何值，都会对refcount进行加1操作，这就造成了bug。