上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

# CVE-2011-4914

学号: 522031910439　姓名: 梁俊轩

2025 年 6 月 13 日

## 1 漏洞概述

### 1.1 CVE 描述

Linux kernel 2.6.39 之前版本在 ROSE 协议实现中存在漏洞，该漏洞源于未验证某些数据长度值与数据传送总值是否一致。远程攻击者可利用该漏洞通过特制的数据 ROSE 套接字，从内核内存中获得敏感信息或导致拒绝服务。

1. 解析 FAC_NATIONAL_DIGIS 时未验证 digipeaters 数量，导致堆溢出
2. 解析 FAC_CCITT_DEST_NSAP/FAC_CCITT_SRC_NSAP 时未验证长度值：
    - 长度 <10 导致 memcpy 下溢，引发堆破坏
    - 长度 >20 导致 callsign 数组栈溢出

### 1.2 影响的软件/组件及版本

在 2.6.39 之前的 Linux 内核。

### 1.3 分析调试环境

操作系统：Ubuntu 14.04
Linux 内核版本：2.6.34

## 2 漏洞分析

### 2.1 ROSE 协议

ROSE 协议即远程操作服务元素协议，是一种提供远程操作能力、允许分布式应用程序实体间交互作用的协议。它处于 OSI 模型的会话层，在传输层提供的服务基础上进一步加强了通信的控制和管理。ROSE 协议一旦接收到远程操作服务请求，允许接收实体执行操作并报告操作结果。在操作执行过程中，假定存在对等结构应用实体间的应用关联，通过原语建立与其服务用户间的通信连接，服务请求原语实现对服务用户的驱动过程。

## 2.2 ROSE 协议设施解析

ROSE 协议 (X.25 PLP) 中的设施字段用于协商连接参数。漏洞位于设施字段解析函数中，该函数未验证用户提供的长度字段与数据包实际长度的一致性。

## 2.3 漏洞成因

漏洞代码位于 net/rose/rose_subr.c 的 rose_parse_national 函数，未检查 digipeaters 数量上限，导致 source_digis 和 dest_digis 数组写入数据时超出数组边界：

```
static int rose_parse_national(unsigned char *p, struct
    rose_facilities_struct *facilities, int len)
{
    ......
    facilities->source_ndigis = 0;
    facilities->dest_ndigis   = 0;
    for (pt = p + 2, lg = 0 ; lg < l ; pt += AX25_ADDR_LEN, lg +=
        AX25_ADDR_LEN) {
        if (pt[6] & AX25_HBIT)
            memcpy(&facilities->dest_digis[facilities->
                dest_ndigis++], pt, AX25_ADDR_LEN);
        else
            memcpy(&facilities->source_digis[facilities->
                source_ndigis++], pt, AX25_ADDR_LEN);
    }
```

在 net/rose/rose_subr.c 的 rose_parse_ccitt 函数同时存在两个漏洞：当 l<10 时，l-10 为负数，memcpy 会将其转换为极大的无符号数，导致从 p + 12 开始的大量内存被复制，造成堆内存损坏，内核可能因非法内存访问触发 panic。当 l>20 时，callsign 数组因为为 20 字节大小，memcpy(callsign, p + 12, l - 10) 会导致栈溢出，攻击者可覆盖栈上的返回地址，执行任意代码。

```
static int rose_parse_ccitt(unsigned char *p, struct
    rose_facilities_struct *facilities, int len)
{
    ......
    case 0xC0:
        l = p[1];
        if (*p == FAC_CCITT_DEST_NSAP) {
            memcpy(&facilities->source_addr, p + 7, ROSE_ADDR_LEN
                );
            memcpy(callsign, p + 12,   l - 10);
            callsign[l - 10] = '\0';
            asc2ax(&facilities->source_call, callsign);
        }
        if (*p == FAC_CCITT_SRC_NSAP) {
            memcpy(&facilities->dest_addr, p + 7, ROSE_ADDR_LEN);
            memcpy(callsign, p + 12, l - 10);
}
```

## 2.4 漏洞复现（触发路径）

本人使用的 Linux 内核版本为 2.6.34。

漏洞路径如图1所示。

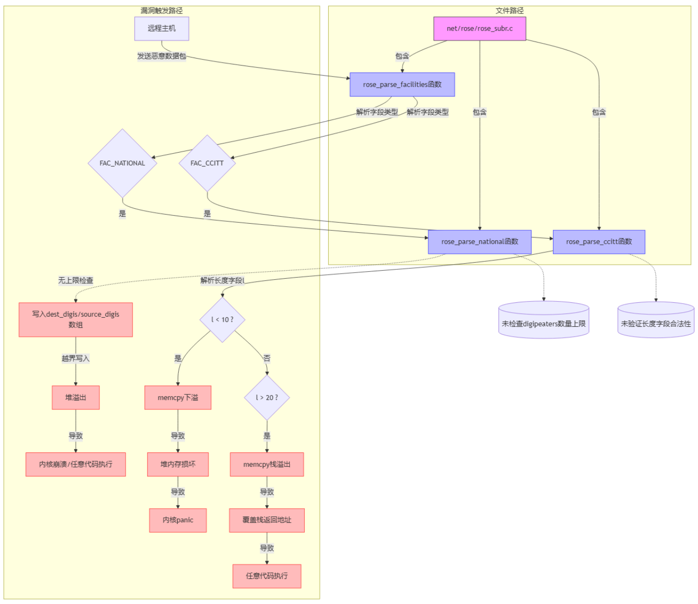

图 1 漏洞触发路径

## 2.5 漏洞可利用性分析

CVSS 向量：(AV:N/AC:L/Au:N/C:P/I:N/A:P)

具体如表1所示。

## 2.6 攻击者利用步骤

攻击者首先需要识别出运行 2.6.39 之前内核且启用 ROSE 协议的系统，之后构造恶意的 ROSE CALL REQUEST 数据包，具体是在数据包中添加包含超过 8 个 digipeaters 的 FAC_NATIONAL_DIGIS，以及长度值异常（小于 10 或大于 20）的 FAC_CCITT_DEST_NSAP，随后将该数据包发送至目标系统，最终通过触发内存破坏来实现拒绝服务或潜在的权限提升。

| 指标 | 含义 | 值 | 解读 |
|---|---|---|---|
| AV | 攻击向量 (Attack Vector) | N (Network) | 攻击者可通过网络访问目标系统 |
| AC | 攻击复杂度 (Attack Complexity) | L (Low) | 攻击所需条件简单，易于实现 |
| Au | 身份验证 (Authentication) | N (None) | 攻击者无需进行身份验证即可实施攻击 |
| C | 机密性影响 (Confidentiality Impact) | P (Partial) | 攻击会导致部分数据机密性泄露 |
| I | 完整性影响 (Integrity Impact) | N (None) | 攻击不会导致数据完整性破坏 |
| A | 可用性影响 (Availability Impact) | P (Partial) | 攻击会导致系统部分功能不可用 |

**表 1 CVSS 向量解读**

## 3 修复方案

### 3.1 已有的修复方案代码片段及分析

官方的修改方式位于 commit 中，经过了两次修改，一方面，在解析 FAC_NATIONAL_DIGIS 设施字段时，增加对 digipeaters 数量的检查，将其与 ROSE_MAX_DIGIS 进行比对，若超出上限则终止解析，以此防范堆溢出问题；另一方面，解析 FAC_CCITT_DEST_NSAP 和 FAC_CCITT_SRC_NSAP 字段时，严格校验长度值，当长度小于 10 或大于 20 时立即中止解析，避免因 memcpy 操作引发堆内存下溢、栈溢出等情况：

https://github.com/torvalds/linux/commit/e0bccd315db0c2f919e7fcf9cb60db21d9986f52

具体修改可详见附录中。

### 3.2 可能的临时缓解方案

在网络边界处设置防火墙规则，阻止外部对受影响系统的 ROSE 协议相关端口的访问，从而减少攻击者利用漏洞的机会。此外，密切监控系统的运行状态，特别是网络流量和系统日志，及时发现异常行为并采取相应的措施。

### 3.3 安全开发建议

在安全开发中，应始终将输入验证作为核心环节，对协议解析等场景中的数据长度、边界范围进行严格校验，避免如 ROSE 协议漏洞中因未检查 digipeaters 数量上限或字段长度异常导致的内存溢出问题；

# A 附录

### A.1 官方解决方案

```
diff --git a/include/net/rose.h b/include/net/rose.h
index 5ba9f02..555dd19 100644
--- a/include/net/rose.h
+++ b/include/net/rose.h
```

```
@@ -14,6 +14,12 @@

 #define         ROSE_MIN_LEN                    3

+#define         ROSE_CALL_REQ_ADDR_LEN_OFF      3
+#define         ROSE_CALL_REQ_ADDR_LEN_VAL      0xAA    /* each
    address is 10 digits */
+#define         ROSE_CALL_REQ_DEST_ADDR_OFF     4
+#define         ROSE_CALL_REQ_SRC_ADDR_OFF      9
+#define         ROSE_CALL_REQ_FACILITIES_OFF    14
+
 #define         ROSE_GFI                        0x10
 #define         ROSE_Q_BIT                      0x80
 #define         ROSE_D_BIT                      0x40
@@ -214,7 +220,7 @@ extern void rose_requeue_frames(struct sock
    *);
 extern int  rose_validate_nr(struct sock *, unsigned short);
 extern void rose_write_internal(struct sock *, int);
 extern int  rose_decode(struct sk_buff *, int *, int *, int *,
     int *, int *);
-extern int  rose_parse_facilities(unsigned char *, struct
    rose_facilities_struct *);
+extern int  rose_parse_facilities(unsigned char *, unsigned int,
     struct rose_facilities_struct *);
 extern void rose_disconnect(struct sock *, int, int, int);

 /* rose_timer.c */
diff --git a/net/rose/af_rose.c b/net/rose/af_rose.c
index 5ee0c62..a80aef6 100644
--- a/net/rose/af_rose.c
+++ b/net/rose/af_rose.c
@@ -978,7 +978,7 @@ int rose_rx_call_request(struct sk_buff *skb,
     struct net_device *dev, struct ros
         struct sock *make;
         struct rose_sock *make_rose;
         struct rose_facilities_struct facilities;
-        int n, len;
+        int n;

         skb->sk = NULL;         /* Initially we don't know who it
```

```
                    's for */

@@ -987,9 +987,9 @@ int rose_rx_call_request(struct sk_buff *skb,
        struct net_device *dev, struct ros
              */
        memset(&facilities, 0x00, sizeof(struct
            rose_facilities_struct));

-       len  = (((skb->data[3] >> 4) & 0x0F) + 1) >> 1;
-       len += (((skb->data[3] >> 0) & 0x0F) + 1) >> 1;
-       if (!rose_parse_facilities(skb->data + len + 4, &
     facilities)) {
+       if (!rose_parse_facilities(skb->data +
     ROSE_CALL_REQ_FACILITIES_OFF,
+                                  skb->len -
     ROSE_CALL_REQ_FACILITIES_OFF,
+                                  &facilities)) {
                rose_transmit_clear_request(neigh, lci,
                    ROSE_INVALID_FACILITY, 76);
                return 0;
        }
diff --git a/net/rose/rose_loopback.c b/net/rose/rose_loopback.c
index ae4a9d9..3444562 100644
--- a/net/rose/rose_loopback.c
+++ b/net/rose/rose_loopback.c
@@ -73,9 +73,20 @@ static void rose_loopback_timer(unsigned long
     param)
        unsigned int lci_i, lci_o;

        while ((skb = skb_dequeue(&loopback_queue)) != NULL) {
+               if (skb->len < ROSE_MIN_LEN) {
+                       kfree_skb(skb);
+                       continue;
+               }
                lci_i     = ((skb->data[0] << 8) & 0xF00) + ((skb
                    ->data[1] << 0) & 0x0FF);
                frametype = skb->data[2];
-               dest      = (rose_address *)(skb->data + 4);
+               if (frametype == ROSE_CALL_REQUEST &&
+                   (skb->len <= ROSE_CALL_REQ_FACILITIES_OFF ||
+                               skb->data[ROSE_CALL_REQ_ADDR_LEN_OFF] !=
+                               ROSE_CALL_REQ_ADDR_LEN_VAL)) {
+                                   kfree_skb(skb);
+                                   continue;
+                       }
+                       dest       = (rose_address *)(skb->data +
    ROSE_CALL_REQ_DEST_ADDR_OFF);
                        lci_o      = ROSE_DEFAULT_MAXVC + 1 - lci_i;

                        skb_reset_transport_header(skb);
diff --git a/net/rose/rose_route.c b/net/rose/rose_route.c
index 88a77e9..08dcd2f 100644
--- a/net/rose/rose_route.c
+++ b/net/rose/rose_route.c
@@ -861,7 +861,7 @@ int rose_route_frame(struct sk_buff *skb,
    ax25_cb *ax25)
        unsigned int lci, new_lci;
        unsigned char cause, diagnostic;
        struct net_device *dev;
-       int len, res = 0;
+       int res = 0;
        char buf[11];

 #if 0
@@ -869,10 +869,17 @@ int rose_route_frame(struct sk_buff *skb,
    ax25_cb *ax25)
                return res;
 #endif

+       if (skb->len < ROSE_MIN_LEN)
+               return res;
        frametype = skb->data[2];
        lci = ((skb->data[0] << 8) & 0xF00) + ((skb->data[1] <<
            0) & 0x0FF);
-       src_addr  = (rose_address *)(skb->data + 9);
-       dest_addr = (rose_address *)(skb->data + 4);
+       if (frametype == ROSE_CALL_REQUEST &&
+           (skb->len <= ROSE_CALL_REQ_FACILITIES_OFF ||
+            skb->data[ROSE_CALL_REQ_ADDR_LEN_OFF] !=
+            ROSE_CALL_REQ_ADDR_LEN_VAL))
+                       return res;
+           src_addr  = (rose_address *)(skb->data +
    ROSE_CALL_REQ_SRC_ADDR_OFF);
+           dest_addr = (rose_address *)(skb->data +
    ROSE_CALL_REQ_DEST_ADDR_OFF);

            spin_lock_bh(&rose_neigh_list_lock);
            spin_lock_bh(&rose_route_list_lock);
@@ -1010,12 +1017,11 @@ int rose_route_frame(struct sk_buff *skb,
     ax25_cb *ax25)
                        goto out;
            }

-           len  = (((skb->data[3] >> 4) & 0x0F) + 1) >> 1;
-           len += (((skb->data[3] >> 0) & 0x0F) + 1) >> 1;
-
            memset(&facilities, 0x00, sizeof(struct
                rose_facilities_struct));

-           if (!rose_parse_facilities(skb->data + len + 4, &
    facilities)) {
+           if (!rose_parse_facilities(skb->data +
    ROSE_CALL_REQ_FACILITIES_OFF,
+                                       skb->len -
    ROSE_CALL_REQ_FACILITIES_OFF,
+                                      &facilities)) {
                        rose_transmit_clear_request(rose_neigh, lci,
                            ROSE_INVALID_FACILITY, 76);
                        goto out;
            }
 diff --git a/net/rose/rose_subr.c b/net/rose/rose_subr.c
 index 174d51c..f6c71ca 100644
--- a/net/rose/rose_subr.c
+++ b/net/rose/rose_subr.c
@@ -142,7 +142,7 @@ void rose_write_internal(struct sock *sk, int
     frametype)
                        *dptr++ = ROSE_GFI | lci1;
                        *dptr++ = lci2;
                        *dptr++ = frametype;
-                       *dptr++ = 0xAA;
```

```
+                       *dptr++ = ROSE_CALL_REQ_ADDR_LEN_VAL;
                        memcpy(dptr, &rose->dest_addr,  ROSE_ADDR_LEN);
                        dptr    += ROSE_ADDR_LEN;
                        memcpy(dptr, &rose->source_addr, ROSE_ADDR_LEN);
@@ -246,12 +246,16 @@ static int rose_parse_national(unsigned
    char *p, struct rose_facilities_struct *
        do {
                switch (*p & 0xC0) {
                case 0x00:
+                       if (len < 2)
+                               return -1;
                        p   += 2;
                        n   += 2;
                        len -= 2;
                        break;

                case 0x40:
+                       if (len < 3)
+                               return -1;
                        if (*p == FAC_NATIONAL_RAND)
                                facilities->rand = ((p[1] << 8) &
                                    0xFF00) + ((p[2] << 0) & 0
                                   x00FF);
                        p   += 3;
@@ -260,32 +264,48 @@ static int rose_parse_national(unsigned
    char *p, struct rose_facilities_struct *
                        break;

                case 0x80:
+                       if (len < 4)
+                               return -1;
                        p   += 4;
                        n   += 4;
                        len -= 4;
                        break;

                case 0xC0:
+                       if (len < 2)
+                               return -1;
                        l = p[1];
```

```
+                       if (len < 2 + 1)
+                               return -1;
                        if (*p == FAC_NATIONAL_DEST_DIGI) {
                                if (!fac_national_digis_received) {
+                                       if (1 < AX25_ADDR_LEN)
+                                               return -1;
                                        memcpy(&facilities->source_digis[0], p + 2, AX25_ADDR_LEN);
                                        facilities->source_ndigis = 1;
                                }
                        }
                        else if (*p == FAC_NATIONAL_SRC_DIGI) {
                                if (!fac_national_digis_received) {
+                                       if (1 < AX25_ADDR_LEN)
+                                               return -1;
                                        memcpy(&facilities->dest_digis[0], p + 2, AX25_ADDR_LEN);
                                        facilities->dest_ndigis = 1;
                                }
                        }
                        else if (*p == FAC_NATIONAL_FAIL_CALL) {
+                               if (1 < AX25_ADDR_LEN)
+                                       return -1;
                                memcpy(&facilities->fail_call, p + 2, AX25_ADDR_LEN);
                        }
                        else if (*p == FAC_NATIONAL_FAIL_ADD) {
+                               if (1 < 1 + ROSE_ADDR_LEN)
+                                       return -1;
                                memcpy(&facilities->fail_addr, p + 3, ROSE_ADDR_LEN);
                        }
                        else if (*p == FAC_NATIONAL_DIGIS) {
+                               if (1 % AX25_ADDR_LEN)
+                                               return -1;
                                        fac_national_digis_received = 1;
                                        facilities->source_ndigis = 0;
                                        facilities->dest_ndigis   = 0;
@@ -319,24 +339,32 @@ static int rose_parse_ccitt(unsigned char *p, struct rose_facilities_struct *fac
        do {
                switch (*p & 0xC0) {
                case 0x00:
+                       if (len < 2)
+                               return -1;
                        p   += 2;
                        n   += 2;
                        len -= 2;
                        break;

                case 0x40:
+                       if (len < 3)
+                               return -1;
                        p   += 3;
                        n   += 3;
                        len -= 3;
                        break;

                case 0x80:
+                       if (len < 4)
+                               return -1;
                        p   += 4;
                        n   += 4;
                        len -= 4;
                        break;

                case 0xC0:
+                       if (len < 2)
+                               return -1;
                        l = p[1];

                        /* Prevent overflows*/
@@ -365,49 +393,44 @@ static int rose_parse_ccitt(unsigned char *p, struct rose_facilities_struct *fac
                return n;
 }

-int rose_parse_facilities(unsigned char *p,
+int rose_parse_facilities(unsigned char *p, unsigned packet_len,
        struct rose_facilities_struct *facilities)
 {
        int facilities_len, len;

        facilities_len = *p++;

-       if (facilities_len == 0)
+       if (facilities_len == 0 || (unsigned)facilities_len > packet_len)
                return 0;

-       while (facilities_len > 0) {
-               if (*p == 0x00) {
-                       facilities_len--;
-                       p++;
-
-                       switch (*p) {
-                       case FAC_NATIONAL:              /* National */
-                               len = rose_parse_national(p + 1, facilities, facilities_len - 1);
-                               if (len < 0)
-                                       return 0;
-                               facilities_len -= len + 1;
-                               p += len + 1;
-                               break;
-
-                       case FAC_CCITT:         /* CCITT */
-                               len = rose_parse_ccitt(p + 1, facilities, facilities_len - 1);
-                               if (len < 0)
-                                       return 0;
-                               facilities_len -= len + 1;
-                               p += len + 1;
-                               break;
-
-                       default:
-                               printk(KERN_DEBUG "ROSE: rose_parse_facilities - unknown facilities family %02X\n", *p);
-                               facilities_len--;
-                               p++;
-                               break;
-                       }
-               } else
-                       break;  /* Error in facilities format */
+       while (facilities_len >= 3 && *p == 0x00) {
+               facilities_len--;
+               p++;
+
+               switch (*p) {
+               case FAC_NATIONAL:              /* National */
+                       len = rose_parse_national(p + 1, facilities, facilities_len - 1);
+                       break;
+
+               case FAC_CCITT:         /* CCITT */
+                       len = rose_parse_ccitt(p + 1, facilities, facilities_len - 1);
+                       break;
+
+               default:
+                       printk(KERN_DEBUG "ROSE: rose_parse_facilities - unknown facilities family %02X\n", *p);
+                       len = 1;
+                       break;
+               }
+
+               if (len < 0)
+                       return 0;
+               if (WARN_ON(len >= facilities_len))
+                       return 0;
+               facilities_len -= len + 1;
+               p += len + 1;
```

```
             }

-            return 1;
+            return facilities_len == 0;
    }

    static int rose_create_facilities(unsigned char *buffer, struct
        rose_sock *rose)
--
```