上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

CVE-2011-0709

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 内核 2.6.35-rc5 之前版本的 net/bridge/br_multicast.c 文件中，br_mdb_ip_get 函数存在安全漏洞。远程攻击者可通过发送 IGMP 数据包，利用多播表缺失的情况触发空指针解引用，进而导致系统崩溃（拒绝服务）。

1.2　影响的软件/组件及版本

在 2.6.35-rc5 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　桥接模块的多播处理函数

桥接模块的多播处理函数是网络桥接核心功能的关键组成部分。它专门负责接收、处理和转发网络层的多播数据帧。当网桥（交换机）从某个端口接收到一个目标地址为多播 MAC 地址（例如以 01:00:5e: 开头的地址）的数据帧时，多播处理函数会被触发。其主要职责包括：学习多播组成员信息、维护动态的多播转发表、基于转发表智能决策该多播帧需要被泛洪（Flood）到除接收端口外的所有端口，还是仅精确转发到连接了该组接收者的特定端口集合。

2.2　漏洞成因

用于存储和管理所有已知的多播组信息的哈希表结构位于/usr/src/linux-2.6.34/net/bridge/br_private.c 中，:

```
1 struct net_bridge_mdb_htable
2 {
```

```
3           struct hlist_head                   *mhash;
4           struct rcu_head                     rcu;
5           struct net_bridge_mdb_htable        *old;
6           u32                                 size;
7           u32                                 max;
8           u32                                 secret;
9           u32                                 ver;
10  };
```

桥接模块的多播处理函数是位于/usr/src/linux-2.6.34/net/bridge/br_multicast.c 的 br_mdb_ip_get 函数。

```
1   static struct net_bridge_mdb_entry *__br_mdb_ip_get(
2           struct net_bridge_mdb_htable *mdb, __be32 dst, int hash)
3   {
4           struct net_bridge_mdb_entry *mp;
5           struct hlist_node *p;
6
7           hlist_for_each_entry_rcu(mp, p, &mdb->mhash[hash], hlist[
                mdb->ver]) {
8                   if (dst == mp->addr)
9                           return mp;
10          }
11
12          return NULL;
13  }
```

当通过 br_add_bridge() 函数创建桥接接口时，会调用 br_init() 函数初始化 net_bridge 结构体。此时 mdb 会被初始化为 NULL。当桥接接口需要处理多播流量时（例如接收到 IGMP/MLD 数据包），会通过 br_mdb_init() 函数分配 net_bridge_mdb_htable 实例，若此时多播表尚未初始化 (即 mdb 为 NULL)，直接调用 __br_mdb_ip_get，在上述代码的第九行，会导致空指针解引用。

2.3 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。内核错误的信息详见附录。

2.4 漏洞可利用性分析

CVSS 向量：CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H
具体如表1所示。

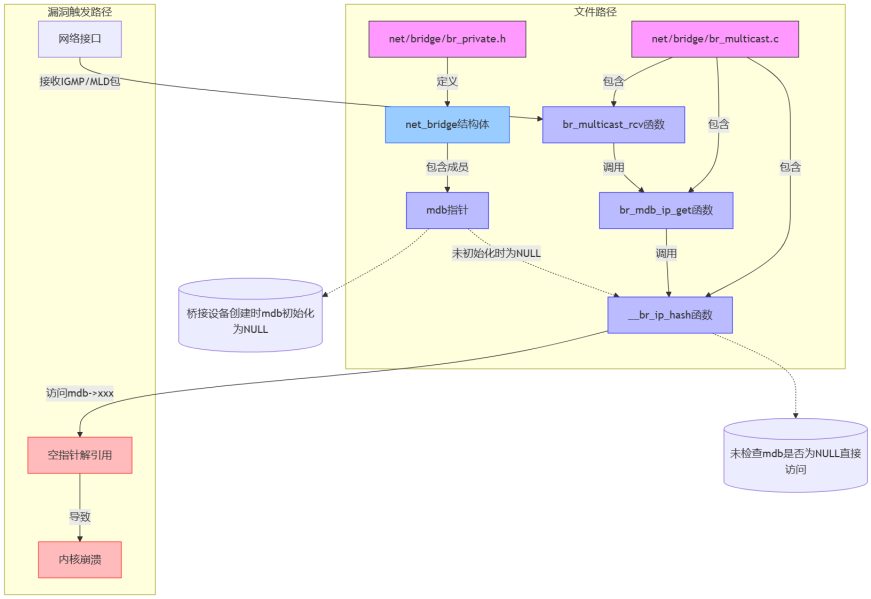

图 1 漏洞触发路径

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
PR	所需权限 (Privileges Required)	N (None)	攻击者无需任何系统权限
UI	用户交互 (User Interaction)	N (None)	攻击无需用户参与即可触发
S	范围 (Scope)	U (Unchanged)	攻击影响范围仅限于目标组件
C	机密性影响 (Confidentiality Impact)	N (None)	攻击不会导致数据机密性泄露
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)	H (High)	攻击会导致系统完全不可用（内核崩溃）

表 1 CVSS 向量解读

2.5 攻击者利用步骤

攻击者可以利用 Scapy 等工具构造包含多播地址的 IGMP 数据包，持续向目标桥接接口发送该数据包，触发桥接模块多播表查询时的空指针解引用漏洞，最终导致内核崩溃并造成系统拒绝服务。

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，重写了 br_mdb_ip_get 函数，分为了 ipv4 和 ipv6 两种情况，在调用 mdb 前会检查是否为空，避免了空指针调用的漏洞：

https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=7f285fa78d4b81b8458f05e77fb6b4624512

```
diff --git a/net/bridge/br_multicast.c b/net/bridge/br_multicast.c
index 9d21d98ae5fa98..27ae946363f16c 100644
--- a/net/bridge/br_multicast.c
+++ b/net/bridge/br_multicast.c
@@ -99,6 +99,15 @@ static struct net_bridge_mdb_entry *__br_mdb_ip_get(
        return NULL;
 }

+static struct net_bridge_mdb_entry *br_mdb_ip_get(
+       struct net_bridge_mdb_htable *mdb, struct br_ip *dst)
+{
+       if (!mdb)
+               return NULL;
+
+       return __br_mdb_ip_get(mdb, dst, br_ip_hash(mdb, dst));
+}
+
 static struct net_bridge_mdb_entry *br_mdb_ip4_get(
        struct net_bridge_mdb_htable *mdb, __be32 dst)
 {
@@ -107,7 +116,7 @@ static struct net_bridge_mdb_entry *br_mdb_ip4_get(
        br_dst.u.ip4 = dst;
        br_dst.proto = htons(ETH_P_IP);

-       return __br_mdb_ip_get(mdb, &br_dst, __br_ip4_hash(mdb,
```

```
      dst));
+         return br_mdb_ip_get(mdb, &br_dst);
  }

 #if defined(CONFIG_IPV6) || defined(CONFIG_IPV6_MODULE)
@@ -119,23 +128,17 @@ static struct net_bridge_mdb_entry *
      br_mdb_ip6_get(
          ipv6_addr_copy(&br_dst.u.ip6, dst);
          br_dst.proto = htons(ETH_P_IPV6);

-         return __br_mdb_ip_get(mdb, &br_dst, __br_ip6_hash(mdb,
      dst));
+         return br_mdb_ip_get(mdb, &br_dst);
  }
 #endif

-static struct net_bridge_mdb_entry *br_mdb_ip_get(
-         struct net_bridge_mdb_htable *mdb, struct br_ip *dst)
-{
-         return __br_mdb_ip_get(mdb, dst, br_ip_hash(mdb, dst));
-}
-
 struct net_bridge_mdb_entry *br_mdb_get(struct net_bridge *br,
                                         struct sk_buff *skb)
 {
          struct net_bridge_mdb_htable *mdb = br->mdb;
          struct br_ip ip;

-         if (!mdb || br->multicast_disabled)
+         if (br->multicast_disabled)
                  return NULL;

          if (BR_INPUT_SKB_CB(skb)->igmp)
```

3.2 可能的临时缓解方案

可通过 iptables 在桥接接口过滤 IGMPv2 Membership Report 数据包（如 iptables -A INPUT -i br0 -p igmp –igmp-type 0x16 -j DROP），或临时禁用桥接多播功能（sysctl -w net.bridge.bridge-multicast-snooping=0），同时关闭非必要桥接接口（ifconfig br0 down）。

3.3 安全开发建议

在开发时要注意检查指针是否为空，否则将无法调用指针指向的地址。所有函数接收指针参数时需验证有效性。

A 附录

A.1 oops

```
BUG: unable to handle kernel NULL pointer dereference at
    0000000000000028
IP: [<ffffffffa0196da0>] __br_ip4_hash+0x0/0x7c [bridge]
PGD 0
Oops: 0000 [#1] SMP
last sysfs file: /sys/module/lockd/initstate
CPU 3
Modules linked in: nfsd exportfs nfs lockd nfs_acl auth_rpcgss
    sunrpc bridge stp ipv6 kvm_amd kvm snd_hda_codec_atihdmi
snd_hda_intel snd_hda_codec snd_hwdep snd_seq snd_seq_device
    snd_pcm snd_timer snd pcspkr serio_raw ata_generic r8169 so
undcore i2c_piix4 pata_acpi i2c_core joydev snd_page_alloc mii
    pata_atiixp shpchp [last unloaded: scsi_wait_scan]

Pid: 0, comm: swapper Not tainted 2.6.35.20100705_8dea564-1.fc11.
    osrc.x86_64 #1 GA-MA74GM-S2H/GA-MA74GM-S2H
RIP: 0010:[<ffffffffa0196da0>]  [<ffffffffa0196da0>]
    __br_ip4_hash+0x0/0x7c [bridge]
RSP: 0018:ffff880001b838a8  EFLAGS: 00010246
RAX: ffff880126028000 RBX: 0000000000000000 RCX: ffff880127b3a828
RDX: 0000000001b80008 RSI: 0000000064ffffef RDI: 0000000000000000
RBP: ffff880001b838b0 R08: ffff8800054c3870 R09: 0000000000000000
R10: 0000000000000000 R11: 0000000000000000 R12: ffff880001b83a00
R13: ffff880001b83a00 R14: ffff880127b3a800 R15: ffff880125ccc400
FS:  00007f17d45ea6f0(0000) GS:ffff880001b80000(0000) knlGS
    :0000000000000000
CS:  0010 DS: 0000 ES: 0000 CR0: 000000008005003b
CR2: 0000000000000028 CR3: 00000000016b0000 CR4: 00000000000006e0
DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
Process swapper (pid: 0, threadinfo ffff880127ab4000, task
```

```
    ffff880127ab96b0)
Stack:
 ffffffffa0196f48 ffff880001b838d0 ffffffffa01970be
    ffff880126028640
<0> ffff880125ccc400 ffff880001b83910 ffffffffa0197511
    ffff880001b83900
<0> ffff880127b3a800 ffff8800054c3868 ffff880126028640
    ffff880127b3a800
Call Trace:
 <IRQ>
 [<ffffffffa0196f48>] ? br_ip_hash+0x1f/0x28 [bridge]
 [<ffffffffa01970be>] br_mdb_ip_get+0x12/0x24 [bridge]
 [<ffffffffa0197511>] br_multicast_leave_group+0x62/0x160 [bridge
    ]
 [<ffffffffa0199028>] br_multicast_rcv+0x60e/0xcda [bridge]
 ...（调用栈中间部分省略）...
 [<ffffffff8100459c>] do_IRQ+0xa7/0xbe
 [<ffffffff8136a1d3>] ret_from_intr+0x0/0x11
 <EOI>
 [<ffffffff8102036c>] ? native_safe_halt+0x6/0x8
 [<ffffffff8136d161>] ? atomic_notifier_call_chain+0x13/0x15
 [<ffffffff81009696>] default_idle+0x27/0x44
 [<ffffffff81001d3a>] cpu_idle+0x58/0x93
 [<ffffffff81364944>] start_secondary+0x1a4/0x1a8
Code: 7e 66 81 fa 81 00 74 0d 31 c0 66 81 fa 88 64 0f 94 c0 c1 e0
    03 89 c2 48 29 93 e0 00 00 00 01 43 68 31 c0 5b 41 5c
c9 c3 90 90 90 <8b> 47 28 89 f1 ba b9 79 37 9e c1 e9 0d 29 f2 55
    29 f0 48 89 e5
RIP  [<ffffffffa0196da0>] __br_ip4_hash+0x0/0x7c [bridge]
 RSP <ffff880001b838a8>
CR2: 0000000000000028
---[ end trace c0f05a4e3727475d ]---
Kernel panic - not syncing: Fatal exception in interrupt
```