IOActive Security Advisory

Title	AppleTalk Response Packet Parsing Array Over-indexing Vulnerability
Severity	High to Critical
Date Discovered	03.03.09
Date Reported	03.03.09
Date Disclosed	08.05.09
CVE-ID	CVE-2009-2193
Author	Ilja van Sprundel

Affected Products

Mac OSX 10.5.x up through 10.5.7 in xnu-1228.12.14 (earlier versions also are likely to be vulnerable).

Description

The Mac OS X AppleTalk stack contains an array over indexing vulnerability that, if exploited correctly while AppleTalk is powered on, could lead to a remote system compromise. Even if only partially exploited, it could lead to denial of service and cause a kernel panic remotely, effectively shutting down the system.

Description

The vulnerability exists in the `atp_rput()` function, which is accessed by way through a couple of other functions. The AppleTalk network entry point is `at_input_packet()`, which is registered dynamically in `atalk_load()` with a call to `proto_register_input()`. It then passes from `at_input_packet()` to `ddp_input()`, which calls `atp_input()`, a function that calls `atp_rput()`. All the Mac OSX AppleTalk code resides in the xnu package's `bsd/netat/` directory:

```
drv_dep.c
static void
at_input_packet(
      __unused protocol_family_t    protocol,
      mbuf_t                                    m)
{
...
      llc_header_t *llc_header;
...
      enet_header_t *enet_header;

...
      if (!appletalk_inited) {
            m_freem(m);
```

```
            return;
      }
...
        enet_header = mtod(m, enet_header_t *);
...
            llc_header = (llc_header_t *)(enet_header+1);
...
            else if (LLC_PROTO_EQUAL(llc_header->protocol,
snap_proto_ddp)) {
...
                  MCHTYPE(m, MSG_DATA); /* set the mbuf type */
...
                  ddp_input(m, ifID);
...
}


ddp.c
void ddp_input(mp, ifID)
     register gbuf_t   *mp;
     register at_ifaddr_t *ifID;
{
      register at_ddp_t *ddp;       /* DDP header */
...
      ddp = (at_ddp_t *)gbuf_rptr(mp);
...
            if (ddp->type == DDP_ATP) {
                  ...
                  atp_input(mp);
                  goto out; /* return; */
              }
...
}


atp_open.c
int atp_input(mp)
      gbuf_t *mp;
{
...
      atp_rput(gref, mp);
...
}


atp_read.c
void atp_rput(gref, m)
gref_t  *gref;
gbuf_t   *m;
{
...
      register at_atp_t *athp;
...
      switch(gbuf_type(m)) {
      case MSG_DATA:
...
```

```
          athp = AT_ATP_HDR(m);
...
          switch (athp->cmd) {
          case ATP_CMD_TRESP:
          {
...
            register struct atp_trans *trp;
            register int    seqno;
...
            /*
             * we just got a response, find the trans record
             */

            for (trp = atp->atp_trans_wait.head; trp; trp = trp-
>tr_list.next) {
                if (trp->tr_tid == UAS_VALUE_NTOH(athp->tid))
                  break;
            }

            seqno = athp->bitmap;
...
            /*
             * If we have already received it, ignore it
             */
            if (!(trp->tr_bitmap&atp_mask[seqno]) || trp-
>tr_rcv[seqno]) { ... } <-- out of bound read access
...
            if (athp->eom)
                trp->tr_bitmap &= atp_lomask[seqno]; ← out of
bound read access
            else
                trp->tr_bitmap &= ~atp_mask[seqno]; ← out of
bound read access
...
            trp->tr_rcv[seqno] = m; ← out of bound write access !
...
            return;
          }
...
      }
}
```

The unsigned character `seqno` is taken out of the `atp` header and used as an index into several arrays without being validated. As it turns out, `atp_mask[]` and `tr_rcv[]` can hold only up to eight elements while `atp_lomask[]` can hold up to nine elements, as shown in the following example, `atp.h`:

```
atp.h
unsigned char atp_mask [] = {
      0x01, 0x02, 0x04, 0x08,
      0x10, 0x20, 0x40, 0x80,
};
```

```
unsigned char atp_lomask [] = {
      0x00, 0x01, 0x03, 0x07,
      0x0f, 0x1f, 0x3f, 0x7f,
      0xff
};
...
struct atp_trans {
      struct atp_trans_q      tr_list;          /* trans list */
      struct atp_state  *tr_queue;        /* state data structure
*/
      gbuf_t                  *tr_xmt;          /* message being
sent */
      gbuf_t                  *tr_rcv[8];       /* message being
rcvd */
      ...
};
```

Remediation

This vulnerability was fixed in Mac OS X 10.5.8 and resolved in Apple Security Update 2009-003. If you're using Mac OS X version 10.5.8 or earlier, it is recommended that you upgrade it.