Date: Sun, 18 Apr 2021 11:41:06 +0300
From: Or Cohen <orcohen@...oaltonetworks.com>
To: oss-security@...ts.openwall.com
Cc: Nadav Markus <nmarkus@...oaltonetworks.com>
Subject: CVE-2021-23133: Linux kernel: race condition in sctp sockets

Hello,

This is an announcement about CVE-2021-23133 which is a race-condition
I found in Linux kernel sctp sockets (net/sctp/socket.c). It can lead to kernel
privilege escalation from the context of a network service or from
an unprivileged process if certain conditions are met.

The bug was fixed on April 13, 2021:
https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=b166a20b07382b8bc1dcee2a448715c9c2c81b5b

=*=*=*=*=*=*=*=*=   VULNERABILITY DETAILS - sctp_destroy_sock list_del
race condition =*=*=*=*=*=*=*=*=

All of the code figures below are from kernel version 5.11

The netns_sctp struct contains sctp related information per network namespace,
one if it's fields is the auto_asconf_splist list.
As the list can be accessed from multiple threads, every access to the list
should be protected by the addr_wq_lock spinlock.

```
(include/net/netns/sctp.h - netns_sctp structure)
...
    struct list_head addr_waitq;
    struct timer_list addr_wq_timer;
    struct list_head auto_asconf_splist;
    /* Lock that protects both addr_waitq and auto_asconf_splist */
    spinlock_t addr_wq_lock;
...
```

The sctp_sock struct contains the auto_asconf_list field which is used in order
to add elements to the auto_asconf_splist.

```
(include/net/sctp/struct.h - sctp_sock structure)
...
    struct list_head auto_asconf_list;
...
```

When creating a sctp socket, the sctp_init_sock method is called, after
setting up and initializing the sock structure, the following code
is executed in the end of the function:

```
(net/sctp/socket.c - sctp_init_sock function)
...
if (net->sctp.default_auto_asconf) {
spin_lock(&sock_net(sk)->sctp.addr_wq_lock);
list_add_tail(&sp->auto_asconf_list,
    &net->sctp.auto_asconf_splist);
sp->do_auto_asconf = 1;
spin_unlock(&sock_net(sk)->sctp.addr_wq_lock);
}
...
```

net->sctp.default_auto_asconf can be set to true via writing to the
proc variable "/proc/sys/net/sctp/default_auto_asconf", which is per
network namespace. If this variable is set, the socket will be added to
the per network namespace auto_asconf_list and do_auto_asconf will be set
to 1 in the socket.

The bug lies in the sctp_destroy_sock function, this function assumes that
when it's called, the addr_wq_lock is held, so it allows itself to run the
following code without any additional locking mechanism:

```
...
    if (sp->do_auto_asconf) {
sp->do_auto_asconf = 0;
list_del(&sp->auto_asconf_list);
}
...
```

However, there are 2 places in kernel code where sk_common_release (which
calls sctp_destroy_sock) is called without taking the lock:
1. In sctp_accept, if the sctp_sock_migrate function fails.
2. In inet_create or inet6_create, if there is a bpf program
   attached to BPF_CGROUP_INET_SOCK_CREATE which denies
   creation of the sctp socket.

=*=*=*=*=*=*=*=*=   TRIGGERING THE VULNERABILITY   =*=*=*=*=*=*=*=*=

I wrote a poc (stcp_race_priv_user.c) which triggers the vulnerability
via technique (2), the poc
simply attaches BPF_CGROUP_SOCK program to BPF_CGROUP_INET_SOCK_CREATE
which denies creation of any socket, and then runs 2 threads that
each one of them creates sctp sockets in a loop. The race is then triggered
and list_add corruption is detected in sctp_init_sock. When running with
CONFIG_DEBUG_LIST the kernel is crashing immediately:

The call stack is as follows:

```
...
[   69.693724] list_add corruption. prev->next should be next
(ffffffff829fa980), but was dead000000000100. (prev=ffff8881079b8538).
[   69.694693] WARNING: CPU: 12 PID: 409 at lib/list_debug.c:28
__list_add_valid+0x4d/0x70
[   69.695345] Modules linked in:
[   69.695601] CPU: 12 PID: 409 Comm: test_sctp_race Not tainted 5.11.0 #74
[   69.696167] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996),
BIOS Ubuntu-1.8.2-1ubuntu1 04/01/2014
[   69.696949] RIP: 0010:__list_add_valid+0x4d/0x70
[   69.697336] Code: c3 48 89 c1 48 c7 c7 10 97 59 82 e8 4d 4f c1 ff
0f 0b 31 c0 c3 48 89 d1 48 c7 c7 60 97 59 82 48 89 f2 48 89 c6 e8 33
4f c1 ff <0f> 0b 31 c0 c3 48 89 fe 48 89 c1 48 c7 c7 b0 97 59 82 e8 1c
4f c1
[   69.698864] RSP: 0018:ffffc90000647e48 EFLAGS: 00010282
[   69.699300] RAX: 0000000000000000 RBX: ffff8881079a8000 RCX: 0000000000000000
[   69.699903] RDX: ffff88842fd27860 RSI: ffff88842fd17a50 RDI: ffff88842fd17a50
[   69.700487] RBP: ffffffff829fa000 R08: 0000000000000003 R09: 0000000000000001
[   69.701086] R10: ffff888100c83a60 R11: ffffc90000647c58 R12: ffff8881079b8538
[   69.701688] R13: ffff8881079a8538 R14: ffffffff829fa980 R15: 0000000000000084
[   69.702273] FS:  00007f2fb82c7b40(0000) GS:ffff88842fd00000(0000)
knlGS:0000000000000000
[   69.702950] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[   69.703426] CR2: 00007f2fb76bcff8 CR3: 0000000107960004 CR4: 00000000003706e0
[   69.704019] DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
[   69.704601] DR3: 0000000000000000 DR6: 00000000fffe0ff0 DR7: 0000000000000400
[   69.705200] Call Trace:
[   69.705414]  sctp_init_sock+0x339/0x380
[   69.705759]  inet_create+0x1ac/0x350
[   69.706054]  __sock_create+0xfd/0x200
[   69.706365]  __sys_socket+0x55/0xd0
[   69.706674]  ? exit_to_user_mode_prepare+0x2f/0x120
[   69.707079]  __x64_sys_socket+0x11/0x20
[   69.707398]  do_syscall_64+0x33/0x40
[   69.707715]  entry_SYSCALL_64_after_hwframe+0x44/0xa9
[   69.708139] RIP: 0033:0x7f2fb77a7f17
...
```

```
This specific poc (stcp_race_priv_user.c) requires CAP_BPF and
CAP_NET_ADMIN capabilities in order
to attach the bpf program, according to https://lwn.net/Articles/820560/,
this is still considered a security boundary.

=*=*=*=*=*=*=*=*=   TRIGGERING FROM UNPRIVILEGED USER  =*=*=*=*=*=*=*=*=

However, if a BPF_CGROUP_INET_SOCK_CREATE program  is already attached,
such that an unprivileged user can fail a creation of some sctp socket,
then the vulnerability can be triggered by an unprivileged user if unprivileged
 user namespaces are enabled, by creating a new user and network
namespace, setting
"/proc/sys/net/sctp/default_auto_asconf" in the new network namespace
and then racing between the 2 threads.

This can be demonstrated by the following files:

1. load_bpf_prog.c - Which loads the BPF_CGROUP_INET_SOCK_CREATE, and should
    be run from a privileged process.
2. stcp_race_unpriv_user.c - Which can be run from a regular, unprivileged
    user.

I haven't checked, but there are probably network security tools which attaches
bpf program to BPF_CGROUP_INET_SOCK_CREATE.

Regarding triggering via technique (2), which is failing sctp_sock_migrate in
sctp_accept, I've tried many tricks in order to fail sctp_sock_migrate
but eventually this requires failing some kmalloc or crypto calls,
which I couldn't
fail in a modern Ubuntu with almost the latest kernel.
However, it may be possible to do that in older kernel versions, or with
some other trick which I am not aware about, or if sctp_accept or
sctp_sock_migrate
changes in the future.

Note that by triggering via this technique, the vulnerability can be triggered
from an unprivileged user without the BPF_CGROUP_INET_SOCK_CREATE
program attached.

=*=*=*=*=*=*=*=*=    TIMELINE    =*=*=*=*=*=*=*=*=

2021-04-08: Bug reported to security () kernel org and linux-distros
() vs openwall org
2021-04-13: Patch submitted to netdev
2021-04-17: Patch committed to mainline kernel
2021-04-18: Public announcement

=*=*=*=*=*=*=*=*=     CREDIT     =*=*=*=*=*=*=*=*=

Or Cohen
Palo Alto Networks
```

Download attachment "sctp_race_priv_user.c**" of type "**application/octet-stream**" (4119 bytes)**

Download attachment "sctp_race_unpriv_user.c**" of type "**application/octet-stream**" (3331 bytes)**

Download attachment "load_bpf_prog.c**" of type "**application/octet-stream**" (2372 bytes)**

Powered by blists - more mailing lists

Please check out the Open Source Software Security Wiki, which is counterpart to this mailing list.

Confused about mailing lists and their use? Read about mailing lists on Wikipedia and check out these guidelines on proper formatting of your messages.