LINUX PLUMBERS CONFERENCE September 20-24, 2021

New Smatch Developments

Dan Carpenter <dan.carpenter@oracle.com>

Param/Key API

Old and Tired:

```
static void match_free(const char *fn, struct expression *call,...
{
    struct expression *arg;

    arg = get_argument_from_call_expr(call->args, 0);
    set_state_expr(my_id, arg, &freed);
}

add_function_hook("kfree", &match_free, NULL);
```

Param/Key API

```
 Function: kfree()
Parameter: 0
      Key: $

 Function: release_resource()
Parameter: 0
      Key: $->start
```

Param/Key API

New and Wired:

```
static void match_free(..., char *name, struct symbol *sym, ...)
{
    set_state(my_id, name, sym, &freed);
}
```

Param/Key API

Same function used for database hooks:

```
add_function_param_key_hook("kfree", &match_free, 0, "$", NULL);
select_return_param_key(FREE, &match_free);
```

Param/Key API

A giant table full of functions:

```
{ "ida_alloc_range", ALLOC, -1, "$", &int_zero, &int_max },
{ "ida_free", RELEASE, 1, "$" },
```

Param/Key API

End Micro Talk 1

Sleeping in Atomic

`GFP_KERNEL vs GFP_ATOMIC`

Sleeping in Atomic

check_preempt_info.c

Giant table full of functions that affect preempt
States: &inc, &dec, and &ignore
Records return information

Does this return path enable/disable preemption **exactly one time**?

Sleeping in Atomic

```
check_preempt.c

Tracks the current preempt count
Gets information from check_preempt_info.c
Handles if (in_atomic()) checks
Records caller information
Exports: get_preempt_cnt()
```

Sleeping in Atomic

check_sleep_info.c

Looks for GFP_KERNEL
Has list of functions that **always** sleep
Records return information

Sleeping in Atomic

```
check_sleeping_in_atomic.c

if (get_preempt_cnt() > 0)
    warn()
```

Sleeping in Atomic

```
drivers/staging/rts5208/xd.c:852 xd_set_unused_block() warn:
sleeping in atomic context

smdb.py preempt xd_set_unused_block

rtsx_exclusive_enter_ss() <- disables preempt
-> rtsx_enter_ss()
   -> rtsx_power_off_card()
      -> xd_cleanup_work()
         -> xd_delay_write()
            -> xd_finish_write()
               -> xd_set_unused_block()
```

Sleeping in Atomic

Life Lessons:

There is a lot of good stuff in the database
The &inc, &dec, &ignore pattern should work for reference counting, and locks
Split things up as much as possible

Sleeping in Atomic

End Micro Talk II

(applause)

Race Conditions

Alexander Popov (CVE-2021-26708)

```
-    const struct vsock_transport *transport = sk->transport;
+    const struct vsock_transport *transport;
+
     lock_sock(sk);

+    transport = sk->transport;
```

Race Conditions

The other thread:

```
static void vsock_deassign_transport(struct vsock_sock *vsk)
{
        if (!vsk->transport)
                return;

        vsk->transport->destruct(vsk); // frees $->transport
        module_put(vsk->transport->module);
        vsk->transport = NULL;
}
```

Race Conditions

Results: Two Bugs, Six False Positives

A type of false positive is when there are two checks.

```
if (llcp_sock->dev == NULL)
    return -EBADFD;
...
lock_sock(sk);
if (!llcp_sock->dev) {
    release_sock(sk);
    return -EBADFD;
}
```

Race Conditions

Lukas Bulwahn and Julia Lawall

Infer which locks are needed by looking at the statement directly after a lock.

```
lock_sock(sk);
if (!llcp_sock->dev) {
```

This gives 600 warnings. Some bugs. Lots of false positives. Hard to analyze.

Race Conditions

Norbert Slusarek (CVE-2021-32606)

```
+     lock_sock(sk);
      if (so->bound) {
          ret = -EISCONN;
          goto out;
      }
      ...
      if (copy_from_sockptr(&so->opt, optval, optlen)) {
          ret = -EFAULT;
          goto out;
      }
```

Race Conditions

Norbert Slusarek (CVE-2021-32606)

Just like mutex_lock(), a copy_from_user() can be controlled by the user to add a user controlled delay and exploit a race condition.

Race Conditions

```
drivers/char/pcmcia/synclink_cs.c:4069 hdlcdev_wan_ioctl()
warn: unlocked access 'info->port.count' expected '->serial_lock'

    /* return error if TTY interface open */
    if (info->port.count)
        return -EBUSY;
    ...

    if(!capable(CAP_NET_ADMIN))
        return -EPERM;
    if (copy_from_user(&new_line, line, size))
        return -EFAULT;
```

Race Conditions

Life Lessons:

Static analysis of race conditions is useful for attackers with a lot of time to invest, but horrible for maintainers.

Locking is too complicated not just for static analysis but also for humans.

Annotations might help.

Run syzbot with a delay in lock(), unlock(), and copy_from_user().

Race Conditions

Questions?

<smatch@vger.kernel.org>