uncontained: Uncovering Container Confusion in the Linux Kernel

Jakob Koschel*, Pietro Borrello*,
Daniele Cono D'Elia, Herbert Bos, Cristiano Giuffrida

*Joint first authors

SAPIENZA
UNIVERSITÀ DI ROMA

Type confusion CS 101

```
void feedElefant(Animal *animal) {
  Elefant *elephant = (Elefant *)animal;
  …
}

Tiger *tiger = new Tiger();
feedElefant(tiger);
```

Type confusion in C

No class, no problem?

Type confusion in C

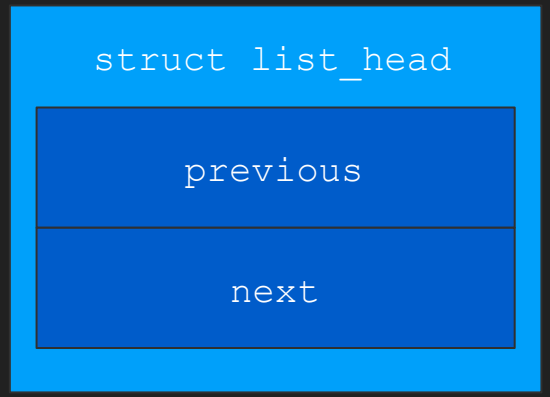
Wrong!

Teaser

We found more than **100 previously undiscovered** invalid "downcast" **bugs** in the Linux kernel!

The kernel had to upgrade the C standard.

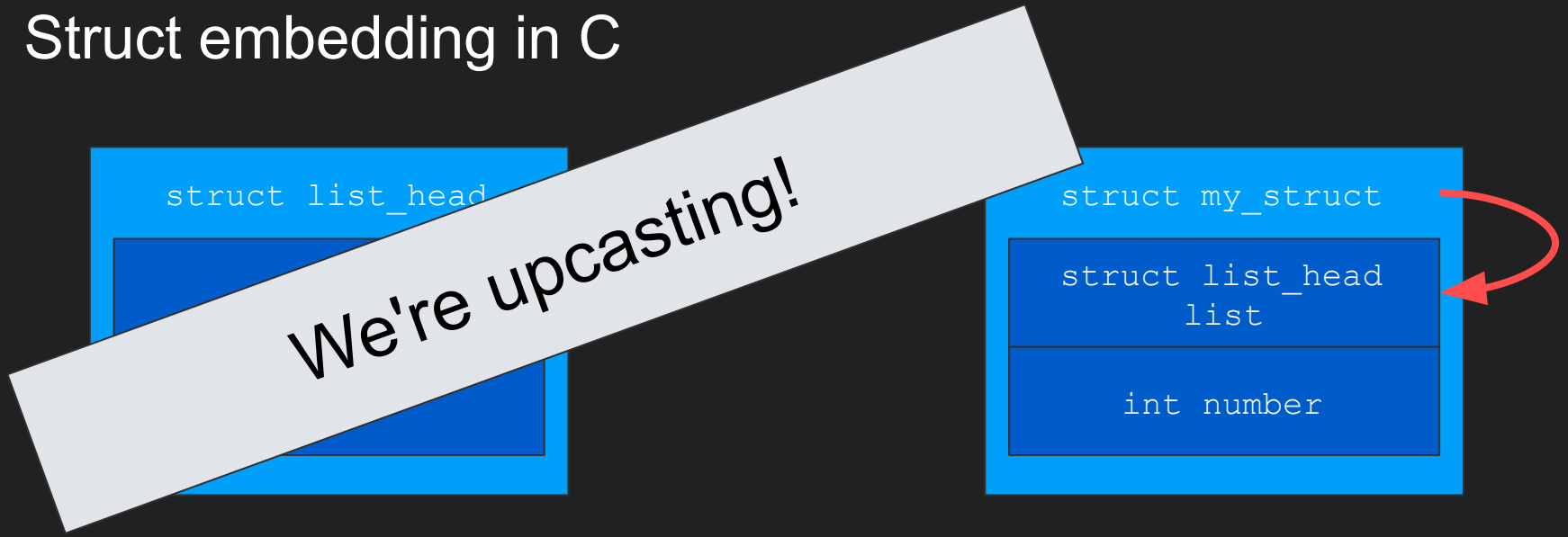
Struct embedding in C



Struct embedding in C

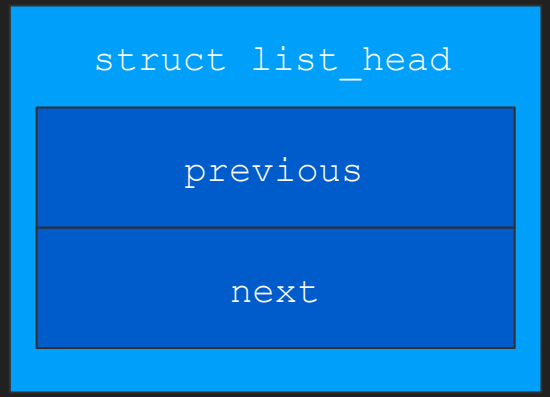

```
struct my_struct *s =  ...;
struct list_head *l = &s->list;
```

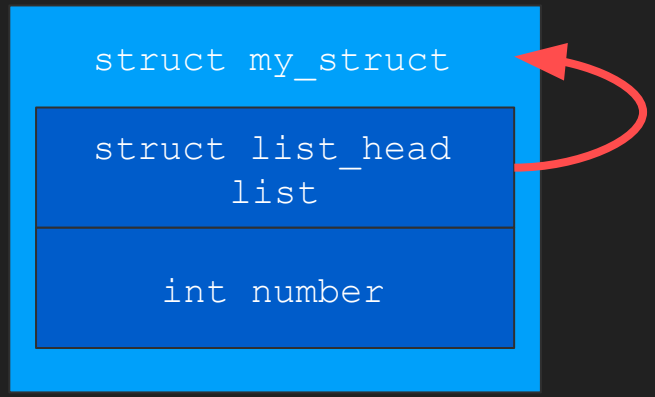
Struct embedding in C




```
struct my_struct *s =  ...;
struct list_head *l = &s->list;
```

Struct embedding in C



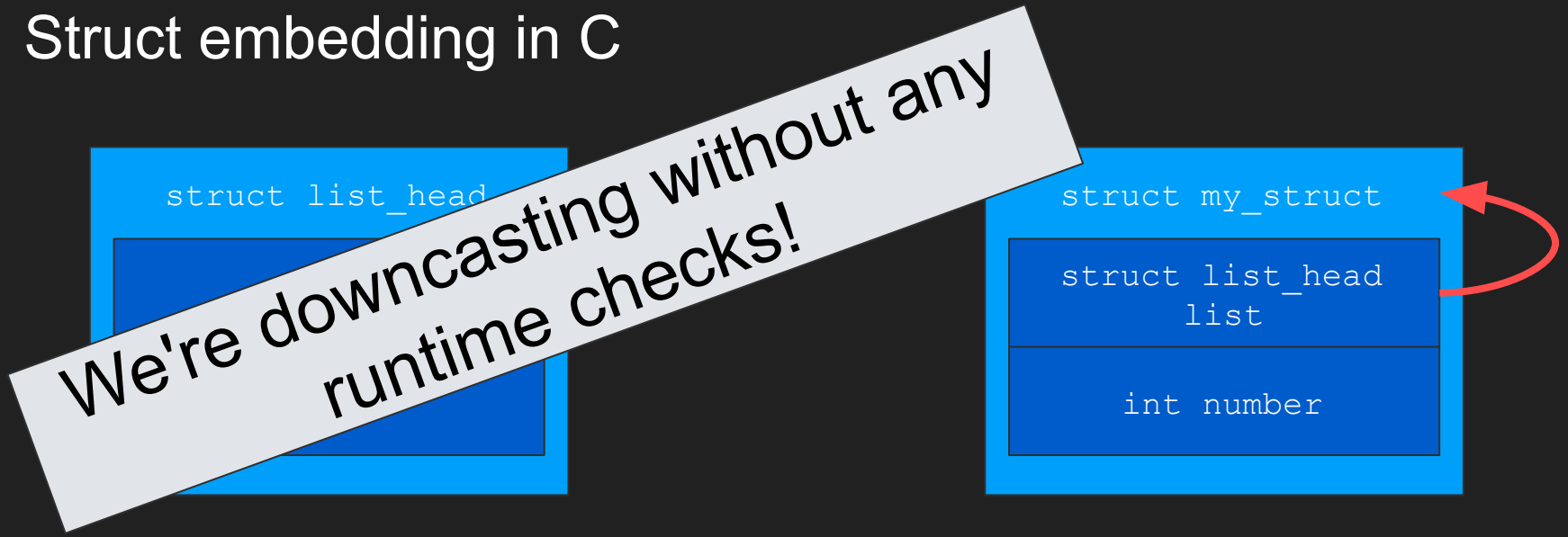



```
struct list_head *l = ...;
struct my_struct *s =
```

Struct embedding in C


```
struct list_head *l = ...;
struct my_struct *s = container_of(l, struct my_struct, list);
```

Struct embedding in C




```
struct list_head *l = ...;
struct my_struct *s = container_of(l, struct my_struct, list);
```

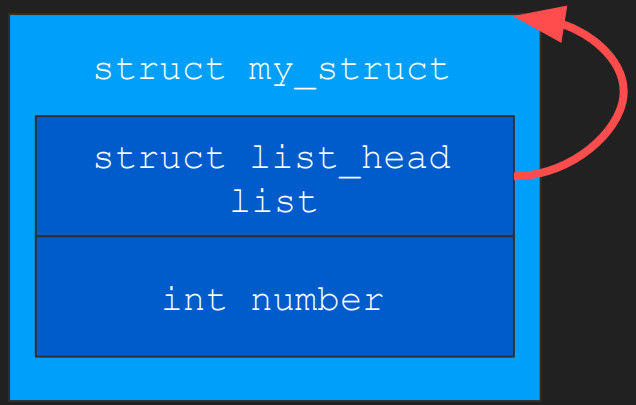
more than 50,000 occurrences of

`container_of` in the Linux kernel

with **~4,000 structure types**!

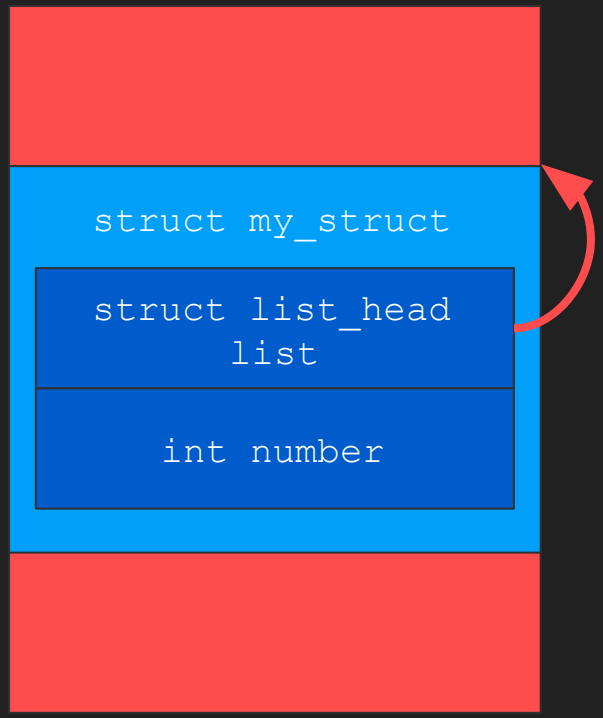
Uncontained sanitizer

Idea 💡



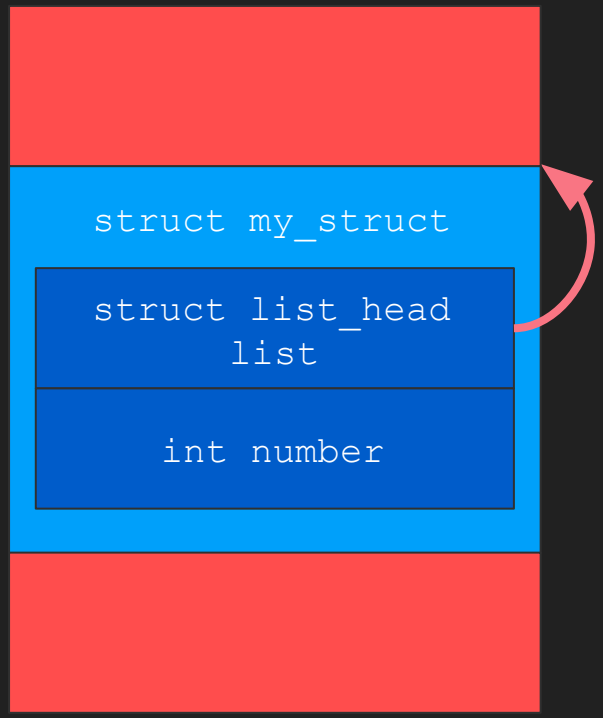
Uncontained sanitizer

Idea 💡

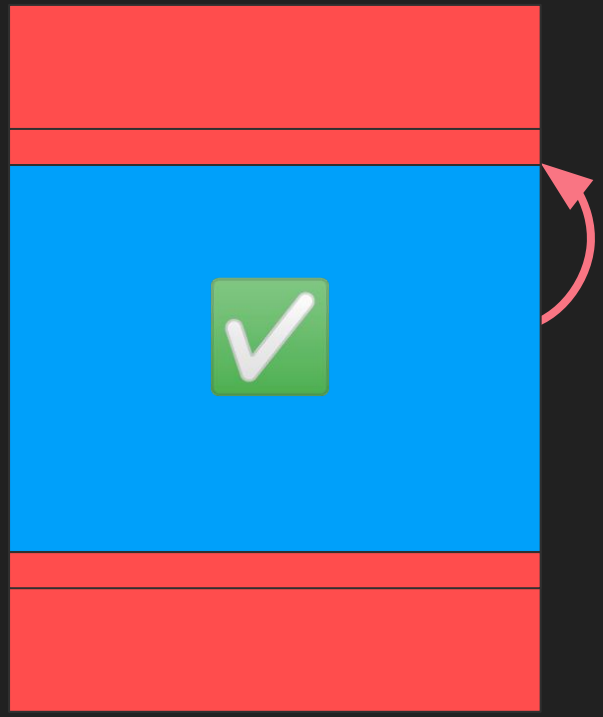


Uncontained sanitizer

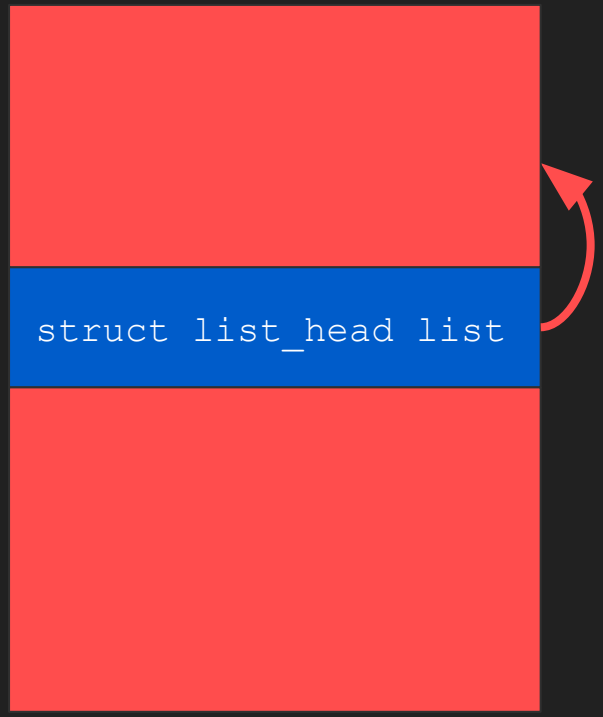
Idea 💡



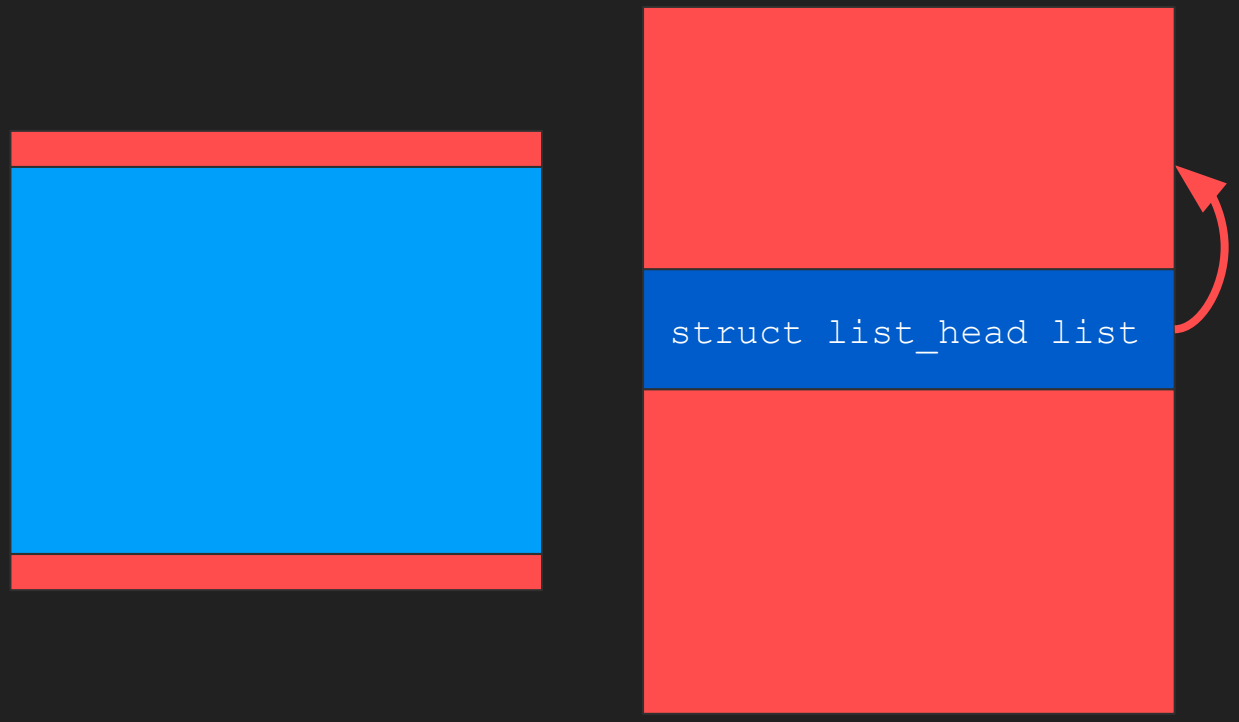
Uncontained sanitizer



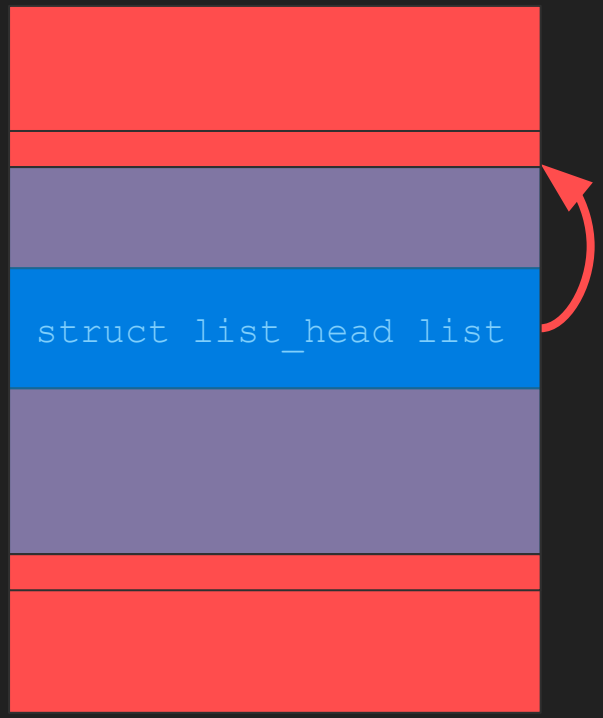
Uncontained sanitizer



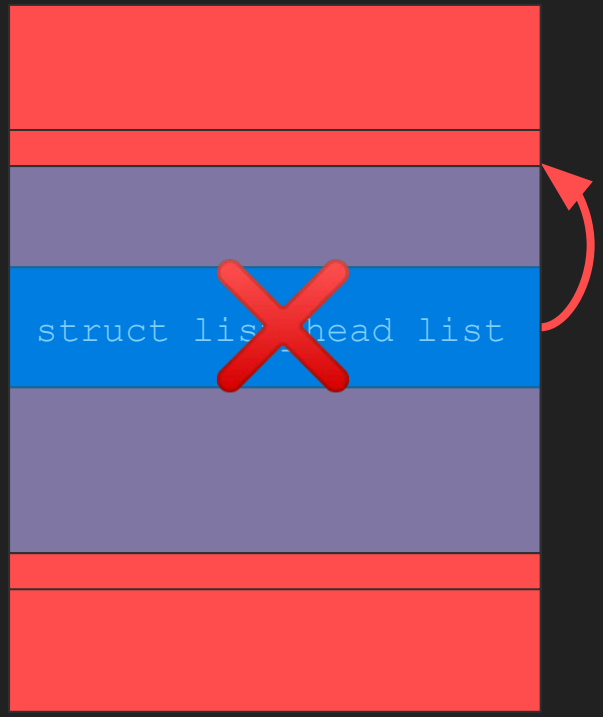
Uncontained sanitizer



Uncontained sanitizer



Uncontained sanitizer



Workflow

syzkaller

Instrument

Fuzzing

Case Study

```
struct sctp_bind_addr *bind_addr = &asoc->base.bind_addr;
...
laddr = container_of(
      bind_addr->address_list.next,
      struct sctp_sockaddr_entry,
      list)->a;
...
```

Case Study

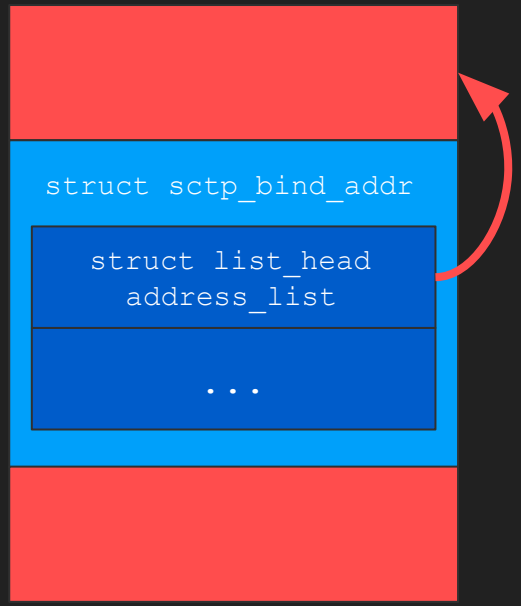
```
struct sctp_bind_addr *bind_addr = &asoc->base.bind_addr;
...
laddr = container_of(
        bind_addr->address_list.next,
        struct sctp_sockaddr_entry,
        list)->a;
...
```

Case Study

```
struct sctp_bind_addr *bind_addr = &asoc->base.bind_addr;
...
laddr = container_of(
        bind_addr->address_list.next,
        struct sctp_sockaddr_entry,
        list)->a;
...
```

Case Study

```
struct sctp_bind_addr *bind_addr = &asoc->base.bind_addr;
...
laddr = container_of(
        bind_addr->address_list.next,
        struct sctp_sockaddr_entry,
        list)->a;
...
```

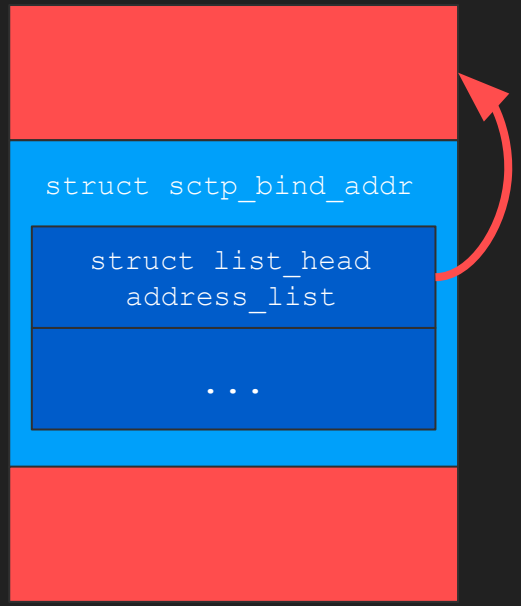


Case Study

```
struct sctp_bind_addr
!=
struct sctp_sockaddr_entry
```

```
struct                                  &asoc->base.bind_addr;

                                        (
                  _addr->address_list.next,
        struct sctp_sockaddr_entry,
        list)->a;
...
```



Bug Patterns

Incompatible Container

Empty List Confusion

Mismatch on Data Structure Operator

Past the End Iterator

Container with Contract

Past the End Iterator

```
struct usb_request *iter;
list_for_each_entry(iter, &request_list, list) {
    if (iter->req == req)
        break;
}

if (iter->req != req)
    return ERR;
```

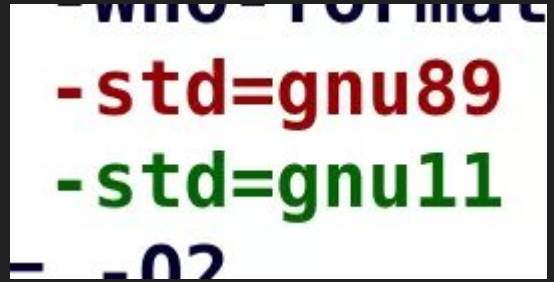
Past the End Iterator

```
struct usb_request *iter;
list_for_each_entry(iter, &request_list, list) {
    if (iter->req == req) {
        found = true;
        break;
    }
}
if (!found)
    return ERR;
```

We built **static dataflow analyzers** and discovered an **additional 80 bugs** with **5 different patterns**

Conclusion

- Type confusions are not only a C++ problem
- `container_of()` causes type confusions all over the kernel
- Automatically discovered more than 100 bugs!
- Over 150 kernel patches submitted
- 8 CVEs assigned
- Caused the kernel to upgrade from c89 to c11



https://www.phoronix.net/image.php?id=2022&image=c11_linux_kernel

Questions?

j.koschel@vu.nl

#sec23-0811-s1-t1