uncontained: Uncovering Container Confusion in the Linux Kernel

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

*Joint first authors
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

No class, no problem?

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

```c
struct list_head
{
    struct list_head *previous;
    struct list_head *next;
};
```
Struct embedding in C

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

We're upcasting!
Struct embedding in C

```
struct list_head
{
    previous
    next
};

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

```
struct list_head

previous

next

struct my_struct

struct list_head

list

int number

struct list_head *l = ...;
struct my_struct *s = container_of(l, struct my_struct, list);
Struct embedding in C

We're downcasting without any runtime checks!

```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💡

struct my_struct

struct list_head

list

int number
Uncontained sanitizer

Idea 🌟

```c
struct my_struct
{
  struct list_head list;
  int number;
};
```
Uncontained sanitizer

Idea 🌟

struct my_struct

struct list_head
    list

int number
Uncontained sanitizer
Uncontained sanitizer

struct list_head list
Uncontained sanitizer

struct list_head list
Uncontained sanitizer

struct list_head list
Uncontained sanitizer
Workflow

Instrument → syzkaller → Fuzzing
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

```c
struct sctp_bind_addr *bind_addr = &asoc->base.bind_addr;
...

laddr = container_of(
    bind_addr->address_list.next,
    struct sctp_sockaddr_entry,
    list)->a;
...
```
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

```c
struct sctp_bind_addr *bind_addr = &asoc->base.bind_addr;
...
```

```c
laddr = container_of(
    bind_addr->address_list->next,
    struct sctp_sockaddr_entry,
    list)->a;
...
```
struct sctp_bind_addr

struct sctp_bind_addr !=
struct sctp_sockaddr_entry

struct sctp_bind_addr

Bug Patterns

- Incompatible Container
- Empty List Confusion
- Mismatch on Data Structure Operator
- Past the End Iterator
- Container with Contract
Past the End Iterator

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

if (iter->req != req)
    return ERR;
```
Past the End Iterator

```c
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

Questions?

📧 j.koschel@vu.nl

#sec23-0811-s1-t1

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