Gaining bounds-checking on trailing arrays in the Upstream Linux Kernel

Gustavo A. R. Silva
gustavoars@kernel.org
fosstodon.org/@gustavoars

Supported by
The Linux Foundation & Alpha-Omega

Everything Open 2024
April 17, 2024
Gladstone, Australia

Who am I?

Who am I?

- **Upstream first** – 8 years.
- Upstream Linux Kernel Engineer.
    - Focused on security.

Who am I?

- **Upstream first** – 8 years.
- Upstream Linux Kernel Engineer.
  - Focused on security.

- Kernel Self-Protection Project (**KSPP**).
- Google Open Source Security Team (**GOSST**).
  - Linux Kernel division.

Agenda

- **Introduction**
  - Arrays in C and The Land of Possibilities.
  - Trailing arrays as Variable Length Objects (VLOs).
  - Flexible arrays and Flexible structures.
- **Gaining bounds-checking on trailing arrays**
  - Ambiguous flexible-arrays declarations
  - Problems and flexible-array transformations.
  - Fortified memcpy() and trailing arrays.
  - The case of UAPI.
  - Bleeding-edge upstream kernel hardening.
- **Conclusions**

Arrays in C and The Land of Possibilities

```c
int happy_array[10];
```

Arrays in C and The Land of Possibilities

- Contiguously allocated objects of the same element type.

- We can iterate over it through indexes from **0 to N - 1**, where N is the maximum number of elements in the array.

int happy_array[10];

indexes: [0-9]

Arrays in C and The Land of Possibilities

- Contiguously allocated objects of the same element type.

- We can iterate over it through indexes from **0 to N - 1**, where N is the maximum number of elements in the array.

- However, C doesn't enforce array's boundaries.

- It's up to the developers to enforce them.

int happy_array[10];

indexes: [0-9]

Arrays in C and The Land of Possibilities

- Contiguously allocated objects of the same element type.

- We can iterate over it through indexes from **0 to N - 1**, where N is the maximum number of elements in the array.

- However, C doesn't enforce array's boundaries.

- It's up to the developers to enforce them.

- Otherwise, you arrive in The Land of Possibilities (a.k.a. UB).

int happy_array[10];

indexes: [0-9]

Arrays in C and The Land of Possibilities

miserable_array[-1]

Trailing arrays

Trailing arrays in the kernel

- Arrays declared at the end of a structure.

```c
struct trailing {
        ...
        some members;
        int happy_array[10];
};
```

Flexible arrays & flexible structures

Flexible arrays & flexible structures

- Flexible array
  - **Trailing** array as **Variable Length Object** (**VLO**).
  - Size is determined at **run-time**.

Flexible arrays & flexible structures

- Flexible array
  - **Trailing** array as **Variable Length Object** (**VLO**).
  - Size is determined at **run-time**.
- Flexible structure
  - Structure that contains a **flexible array**.

```c
struct flex_struct {
        ...
        size_t count;
        struct foo flex_array[];
};
```

Ambiguous flex-array declarations

Ambiguous flex-array declarations

Fake flexible arrays.

- One-element arrays (**buggy hack**).
- Zero-length arrays (**GNU extension**).

Ambiguous flex-array declarations

Fake flexible arrays.

- One-element arrays (**buggy hack**).

- Zero-length arrays (**GNU extension**).

```c
struct fake_flex_1 {
   ...
   size_t count;
   struct foo fake_flex[1];
};
```

```c
struct fake_flex_0 {
   ...
   size_t count;
   struct foo fake_flex[0];
};
```

Ambiguous flex-array declarations

True flexible arrays.

- – "Modern" C99 flexible-array member.

Ambiguous flex-array declarations

True flexible arrays.

- – "Modern" C99 flexible-array member.
- – The last member of an otherwise *non-empty* structure.

```c
struct flex_struct {
    ...
    size_t count;
    struct foo flex_array[];
};
```

Problems with fake flexible arrays

- Three different ways to declare a Variable Length Object (VLO).

Problems with 1-element arrays

```c
struct fake_flex_1 {
        ...
        size_t count;
        struct foo fake_flex[1];
} *p;

alloc_size = sizeof(*p) + sizeof(struct foo) * (count − 1);
p = kmalloc(alloc_size, GFP_KERNEL);
p->count = count;
```

Problems with 1-element arrays

- Prone to **off-by-one** problems.

- Always "contribute" with **size-of-one-element** to the size of the enclosing structure.

```c
struct fake_flex_1 {
        ...
        size_t count;
        struct foo fake_flex[1];
} *p;

alloc_size = sizeof(*p) + sizeof(struct foo) * (count − 1);
p = kmalloc(alloc_size, GFP_KERNEL);
p->count = count;
```

Problems with 1-element arrays

- Prone to **off-by-one** problems.

- Always "contribute" with **size-of-one-element** to the size of the enclosing structure.

- Developers have to remember to subtract **1** from **count**, or **sizeof**(**struct foo**) from **sizeof**(**struct fake_flex_1**).

```
struct fake_flex_1 {
        ...
        size_t count;
        struct foo fake_flex[1];
} *p;

alloc_size = sizeof(*p) + sizeof(struct foo) * (count – 1);
p = kmalloc(alloc_size, GFP_KERNEL);
p->count = count;
```

Problems with 1-element arrays

- **_-Warray-bounds_** false positives.

```c
struct fake_flex_1 {
        ...
        size_t count;
        struct foo fake_flex[1];
} *p;

...
for(i = 0; i < 10; i++)
        p->fake_flex[i] = thingy;
```

Problems with 1-element arrays

- ***-Warray-bounds*** false positives.

```
struct fake_flex_1 {
        ...
        size_t count;
        struct foo fake_flex[1];
} *p;

...
for(i = 0; i < 10; i++)          i == 0 is fine :)
        p->fake_flex[i] = thingy;    i >= 1 is not :/
```

Problems with 1-element arrays

- *-Warray-bounds* false positives.

```
struct fake_flex_1 {
        ...
        size_t count;
        struct foo fake_flex[1];
} *p;

...
for(i = 0; i < 10; i++)            i == 0 is fine :)
        p->fake_flex[i] = thingy;     i >= 1 is not :/
```

**warning: array subscript 1 is above array bounds of
'struct foo[1]' [-Warray-bounds]**

GNU extension: 0-length arrays

```c
struct fake_flex_0 {
        ...
        size_t count;
        struct foo fake_flex[0];
} *p;

alloc_size = sizeof(*p) + sizeof(struct foo) * count;
p = kmalloc(alloc_size, GFP_KERNEL);
p->count = count;
```

GNU extension: 0-length arrays

- Not part of the C standard.

- They don't contribute to the size of the flex struct.

```c
struct fake_flex_0 {
        ...
        size_t count;
        struct foo fake_flex[0];
} *p;

alloc_size = sizeof(*p) + sizeof(struct foo) * count;
p = kmalloc(alloc_size, GFP_KERNEL);
p->count = count;
```

GNU extension: 0-length arrays

- Not part of the C standard.

- They don't contribute to the size of the flex struct.

- Slightly less buggy, but still…

- Be aware of **sizeof(p->fake_flex) == 0**

```
struct fake_flex_0 {
        ...
        size_t count;
        struct foo fake_flex[0];
} *p;

alloc_size = sizeof(*p) + sizeof(struct foo) * count;
p = kmalloc(alloc_size, GFP_KERNEL);
p->count = count;
```

GNU extension: 0-length arrays

- Not part of the C standard.

- They don't contribute to the size of the flex struct.

- Slightly less buggy, but still…

- Be aware of **sizeof(p->fake_flex) == 0**

```
struct fake_flex_0 {
        ...
        size_t count;
        struct foo fake_flex[0];
} *p;


p = kmalloc(struct_size(p, fake_flex, count), GFP_KERNEL);
p->count = count;
```

Problems with ambiguous flexible-array variants

- The Tale of **sizeof()** & the Three Trailing Arrays.

Problems with ambiguous flexible-array variants

- The Tale of **sizeof()** & the Three Trailing Arrays.

```
sizeof(flex_struct->one_element_array) == size-of-element-type
```

Problems with ambiguous flexible-array variants

- The Tale of **sizeof()** & the Three Trailing Arrays.

```
sizeof(flex_struct->one_element_array) == size-of-element-type

sizeof(flex_struct->zero_length_array) == 0
```

Problems with ambiguous flexible-array variants

- The Tale of **sizeof()** & the Three Trailing Arrays.

```
sizeof(flex_struct->one_element_array) == size-of-element-type

sizeof(flex_struct->zero_length_array) == 0

sizeof(flex_struct->flex_array_member) == ? /* Build error */
```

Problems with ambiguous flexible-array variants

- The Tale of **sizeof()** & the Three Trailing Arrays.
  - **sizeof**() returns **different results**.
  - And that's another source of **problems**.
  - Found multiple issues in the kernel.

```
sizeof(flex_struct->one_element_array) == size-of-element-type

sizeof(flex_struct->zero_length_array) == 0

sizeof(flex_struct->flex_array_member) == ? /* Build error */
```

Problems with ambiguous flexible-array variants

Ambiguity is the enemy.

The Land of Possibilities

Undefined Behavior

The Land of Possibilities

Undefined Behavior – The bug

- – e48f129c2f20 ("[SCSI] cxgb3i: convert cdev->l2opt to use…")

```
struct l2t_data {
        unsigned int nentries;
        struct l2t_entry *rover;
        atomic_t nfree;
        rwlock_t lock;
        struct l2t_entry l2tab[0];
+       struct rcu_head rcu_head;
};
```

The Land of Possibilities

Undefined Behavior – The bug

- e48f129c2f20 ("[SCSI] cxgb3i: convert cdev->l2opt to use…")

- Compilers cannot detect dangerous code like this.

```
struct l2t_data {
        unsigned int nentries;
        struct l2t_entry *rover;
        atomic_t nfree;
        rwlock_t lock;
        struct l2t_entry l2tab[0];
+       struct rcu_head rcu_head;
};
```

The Land of Possibilities

Undefined Behavior – The fix

– 76497732932f ("cxgb3/l2t: Fix undefined behavior")

```
struct l2t_data {
        unsigned int nentries;
        struct l2t_entry *rover;
        atomic_t nfree;
        rwlock_t lock;
-       struct l2t_entry l2tab[0];
        struct rcu_head rcu_head;
+       struct l2t_entry l2tab[];
};
```

The Land of Possibilities

Undefined Behavior – The fix

– 76497732932f ("cxgb3/l2t: Fix undefined behavior")

– **Kick-off** of flexible-array transformations (FATs) in the **KSPP**.

```
struct l2t_data {
        unsigned int nentries;
        struct l2t_entry *rover;
        atomic_t nfree;
        rwlock_t lock;
-       struct l2t_entry l2tab[0];
        struct rcu_head rcu_head;
+       struct l2t_entry l2tab[];
};
```

The Land of Possibilities

Undefined Behavior – The fix

- 76497732932f ("cxgb3/l2t: Fix undefined behavior")

- **Kick-off** of flexible-array transformations (FATs) in the **KSPP**.

- Bug introduced in **2011**. Fixed in **2019**.

```
        struct l2t_data {
                unsigned int nentries;
                struct l2t_entry *rover;
                atomic_t nfree;
                rwlock_t lock;
-               struct l2t_entry l2tab[0];
                struct rcu_head rcu_head;
+               struct l2t_entry l2tab[];
        };
```

The Land of Possibilities

Undefined Behavior – The bug

– f5823fe6897c ("qed: Add ll2 option to limit the number of...")

```
#define ETH_TX_MAX_BDS_PER_NON_LSO_PACKET        18

struct qed_ll2_tx_packet {
        ...
+       /* Flexible Array of bds_set determined by max_bds_per_packet */
        struct {
                struct core_tx_bd *txq_bd;
                dma_addr_t tx_frag;
                u16 frag_len;
-       } bds_set[ETH_TX_MAX_BDS_PER_NON_LSO_PACKET];
+       } bds_set[1];
 };
```

The Land of Possibilities

Undefined Behavior – The bug

- f5823fe6897c ("qed: Add ll2 option to limit the number of...")
- **Fake** flex-array transformation ([18] → [1]).

```
#define ETH_TX_MAX_BDS_PER_NON_LSO_PACKET        18

struct qed_ll2_tx_packet {
        ...
+       /* Flexible Array of bds_set determined by max_bds_per_packet */
        struct {
                struct core_tx_bd *txq_bd;
                dma_addr_t tx_frag;
                u16 frag_len;
-       } bds_set[ETH_TX_MAX_BDS_PER_NON_LSO_PACKET];
+       } bds_set[1];
 };
```

The Land of Possibilities

Undefined Behavior – The bug

- f5823fe6897c ("qed: Add ll2 option to limit the number of...")

- Now there is a 1-element array embedded in the middle of struct **qed_ll2_tx_queue**

```
struct qed_ll2_tx_queue {
        ...
-       struct qed_ll2_tx_packet *descq_array;
+       void *descq_mem; /* memory for variable sized qed_ll2_tx_packet*/
        struct qed_ll2_tx_packet *cur_send_packet;
        struct qed_ll2_tx_packet cur_completing_packet;
        ...
        u16 cur_completing_frag_num;
        bool b_completing_packet;
};
```

The Land of Possibilities

Undefined Behavior – The fix

- a93b6a2b9f46 ("qed/red_ll2: Replace one-element array...")

```
struct qed_ll2_tx_packet {
        struct core_tx_bd *txq_bd;
        dma_addr_t tx_frag;
        u16 frag_len;
-   } bds_set[1];
+   } bds_set[];
};
```

```
struct qed_ll2_tx_queue {
        ...
-       struct qed_ll2_tx_packet cur_completing_packet;
        ...
        u16 cur_completing_frag_num;
        bool b_completing_packet;
        ...
+       struct qed_ll2_tx_packet cur_completing_packet;
};
```

The Land of Possibilities

Undefined Behavior – The fix

- a93b6a2b9f46 ("qed/red_ll2: Replace one-element array...")
- Bug introduced in **2017**. Fixed in **2020**.

```
struct qed_ll2_tx_packet {
        struct core_tx_bd *txq_bd;
        dma_addr_t tx_frag;
        u16 frag_len;
-    } bds_set[1];
+    } bds_set[];
};
```

```
struct qed_ll2_tx_queue {
        ...
-       struct qed_ll2_tx_packet cur_completing_packet;
        ...
        u16 cur_completing_frag_num;
        bool b_completing_packet;
        ...
+       struct qed_ll2_tx_packet cur_completing_packet;
};
```

Then one day in Paris…

```
$ git grep -nwW 'struct\sqed_ll2_tx_queue'
```

$ git grep -nwW '**struct**\s**qed_ll2_tx_queue**'

```
struct qed_ll2_tx_queue {
    ...
-   struct qed_ll2_tx_packet cur_completing_packet;
    ...
    u16 cur_completing_frag_num;
    bool b_completing_packet;
    ...
+   struct qed_ll2_tx_packet cur_completing_packet;
};
```

$ git grep -nwW 'struct\sqed_ll2_tx_queue'

- **struct qed_ll2_tx_packet** is a flexible structure.

```
struct qed_ll2_tx_queue {
    ...
-   struct qed_ll2_tx_packet cur_completing_packet;
    ...
    u16 cur_completing_frag_num;
    bool b_completing_packet;
    ...
+   struct qed_ll2_tx_packet cur_completing_packet;
};
```

$ git grep -nwW 'struct\sqed_ll2_tx_queue'

- **struct qed_ll2_tx_packet** is a flexible structure.
- We are looking for `struct qed_ll2_tx_queue`

```
struct qed_ll2_tx_queue {
    ...
-   struct qed_ll2_tx_packet cur_completing_packet;
    ...
    u16 cur_completing_frag_num;
    bool b_completing_packet;
    ...
+   struct qed_ll2_tx_packet cur_completing_packet;
};
```

$ git grep -nwW '**struct**\s**qed_ll2_tx_queue**'

```
struct qed_ll2_info {

  ...

  struct qed_ll2_tx_queue tx_queue;
  struct qed_ll2_cbs cbs;
};
```

$ git grep -nwW '**struct**\s**qed_ll2_tx_queue**'

Undefined Behavior – The bug

```
struct qed_ll2_info {

    ...

    struct qed_ll2_tx_queue tx_queue;
    struct qed_ll2_cbs cbs;
};
```

$ git grep -nwW '**struct**\s**qed_ll2_tx_queue**'

Undefined Behavior – The bug

```
                                            struct qed_ll2_info {
                                                ...
                                                struct qed_ll2_tx_queue {
                                                    ...

struct qed_ll2_info {                               struct qed_ll2_tx_packet {
                                                        ...
    ...                                                 struct {
                                                            struct core_tx_bd *txq_bd;
    struct qed_ll2_tx_queue tx_queue;                       dma_addr_t tx_frag;
    struct qed_ll2_cbs cbs;                                 u16 frag_len;
};                                                      } bds_set[];
                                                    };
                                                };
                                                struct qed_ll2_cbs cbs;
                                            };
```

$ git grep -nwW '**struct**\s**qed_ll2_tx_queue**'

Undefined Behavior – The bug

```
                                   struct qed_ll2_info {
                                       ...
                                       struct qed_ll2_tx_queue {
                                           ...

                                           struct qed_ll2_tx_packet {
                                               ...
struct qed_ll2_info {                           struct {
                                                    struct core_tx_bd *txq_bd;
    ...                                             dma_addr_t tx_frag;
                                                    u16 frag_len;
    struct qed_ll2_tx_queue tx_queue;           } bds_set[]; /* flex array */
    struct qed_ll2_cbs cbs;                 };
};                                     };
                                       struct qed_ll2_cbs cbs;
                                   };
```

$ git grep -nwW '**struct**\s**qed_ll2_tx_queue**'

Undefined Behavior – The bug

```
                                    struct qed_ll2_info {
                                        ...
                                        struct qed_ll2_tx_queue {
                                            ...
                                            /* flex structure */
                                            struct qed_ll2_tx_packet {
                                                ...
                                                struct {
                                                    struct core_tx_bd *txq_bd;
                                                    dma_addr_t tx_frag;
                                                    u16 frag_len;
                                                } bds_set[]; /* flex array */
                                            };
                                        };
                                        struct qed_ll2_cbs cbs;
                                    };
```

```
struct qed_ll2_info {

    ...

    struct qed_ll2_tx_queue tx_queue;
    struct qed_ll2_cbs cbs;
};
```

$ git grep -nwW '**struct**\s**qed_ll2_tx_queue**'

Undefined Behavior – The bug

```
struct qed_ll2_info {
                                            struct qed_ll2_info {
    ...                                         ...
                                                struct qed_ll2_tx_queue {
                                                    ...
                                                    /* flex structure */
                                                    struct qed_ll2_tx_packet {
                                                        ...
    struct qed_ll2_tx_queue tx_queue;                   struct {
    struct qed_ll2_cbs cbs;                                 struct core_tx_bd *txq_bd;
};                                                          dma_addr_t tx_frag;
                                                            u16 frag_len;
                                                        } bds_set[];  /* flex array */
                                                    };
                                                };
                                                struct qed_ll2_cbs cbs;  /* BUG! */
                                            };
```

$ git grep -nwW '**struct**\s**qed_ll2_tx_queue**'

Undefined Behavior – The bug

– Just out of curiosity.

```c
struct qed_ll2_cbs {
        qed_ll2_complete_rx_packet_cb rx_comp_cb;
        qed_ll2_release_rx_packet_cb rx_release_cb;
        qed_ll2_complete_tx_packet_cb tx_comp_cb;
        qed_ll2_release_tx_packet_cb tx_release_cb;
        qed_ll2_slowpath_cb slowpath_cb;
        void *cookie;
};
```

$ git grep -nwW '**struct**\s**qed_ll2_tx_queue**'

Undefined Behavior – The bug

– Just out of curiosity.

– Structure full of function pointers.

```
struct qed_ll2_cbs {
        qed_ll2_complete_rx_packet_cb rx_comp_cb;
        qed_ll2_release_rx_packet_cb rx_release_cb;
        qed_ll2_complete_tx_packet_cb tx_comp_cb;
        qed_ll2_release_tx_packet_cb tx_release_cb;
        qed_ll2_slowpath_cb slowpath_cb;
        void *cookie;
};
```

$ git grep -nwW '**struct**\s**qed_ll2_tx_queue**'

Undefined Behavior – The fix

- https://lore.kernel.org/linux-hardening/ZQ+Nz8DfPg56pIzr@work/

```
struct qed_ll2_info {

    ...
    struct qed_ll2_tx_queue tx_queue;
    struct qed_ll2_cbs cbs;
};
```

```
  struct qed_ll2_info {
      ...
+     struct qed_ll2_cbs cbs;
      struct qed_ll2_rx_queue rx_queue;
      struct qed_ll2_tx_queue tx_queue;
-     struct qed_ll2_cbs cbs;
  };
```

$ git grep -nwW '**struct**\s**qed_ll2_tx_queue**'

Undefined Behavior – The fix

- https://lore.kernel.org/linux-hardening/ZQ+Nz8DfPg56pIzr@work/

- Bug introduced in **2017**. Fixed in **2023**.

```
  struct qed_ll2_info {
    ...
+   struct qed_ll2_cbs cbs;
    struct qed_ll2_rx_queue rx_queue;
    struct qed_ll2_tx_queue tx_queue;
-   struct qed_ll2_cbs cbs;
  };
```

$ git grep -nwW '**struct**\s**qed_ll2_tx_queue**'

Undefined Behavior – The fix

- https://lore.kernel.org/linux-hardening/ZQ+Nz8DfPg56pIzr@work/

- Bug introduced in **2017**. Fixed in **2023**. (In Paris ^.^)

```
struct qed_ll2_info {
    ...
+   struct qed_ll2_cbs cbs;
    struct qed_ll2_rx_queue rx_queue;
    struct qed_ll2_tx_queue tx_queue;
-   struct qed_ll2_cbs cbs;
};
```

$ git grep -nwW '**struct**\s**qed_ll2_tx_queue**'

Undefined Behavior – The fix

- https://lore.kernel.org/linux-hardening/ZQ+Nz8DfPg56pIzr@work/

- Bug introduced in **2017**. Fixed in **2023**. (In Paris ^.^)

- Bugfix applied to multiple stable trees.

```
struct qed_ll2_info {
    ...
+   struct qed_ll2_cbs cbs;
    struct qed_ll2_rx_queue rx_queue;
    struct qed_ll2_tx_queue tx_queue;
-   struct qed_ll2_cbs cbs;
};
```

"Nice find! Was this located with
         -Wflex-array-member-not-at-end ?"

    –Kees Cook

-Wflex-array-member-not-at-end

GCC **new compiler option** (coming soon in GCC 14).

- https://gcc.gnu.org/pipermail/gcc-patches/2023-March/614794.html

- https://gcc.gnu.org/pipermail/gcc-patches/2023-March/614793.html

- https://gcc.gnu.org/pipermail/gcc-patches/2023-March/614790.html

```
"A structure or a union with a C99 flexible array member is the
    middle field of another structure, for example:
        struct flex  { int length; char data[]; };
        struct mid_flex { int m; struct flex flex_data; int n; };
    In the above, 'mid_flex.flex_data.data[]' has undefined behavior.
    Compilers do not handle such case consistently, Any code relying on
    such case should be modified to ensure that flexible array members
    only end up at the ends of structures.
    Please use warning option '-Wflex-array-member-not-at-end' to
    identify all such cases in the source code and modify them.  This
    warning will be on by default starting from GCC 14."

    -Qing Zhao
```

-Wflex-array-member-not-at-end

GCC new compiler option (coming soon in GCC 14)

- **59,056 warnings** in Linux next-20230518.

-Wflex-array-member-not-at-end

GCC new compiler option (coming soon in GCC 14)

- **59,056 warnings** in Linux next-20230518.
- Fortunately, <u>only</u> **650 are unique**.

So I went and took a look at my build logs from that time...

-Wflex-array-member-not-at-end

GCC new compiler option (coming soon in GCC 14)

```
drivers/net/ethernet/qlogic/qed/qed_ll2.h:

100 struct qed_ll2_info {
...
114         struct qed_ll2_tx_queue tx_queue;
115         struct qed_ll2_cbs cbs;
116 };

In file included from drivers/net/ethernet/qlogic/qed/qed_dev.c:33:
drivers/net/ethernet/qlogic/qed/qed_ll2.h:114:33: warning: structure
containing a flexible array member is not at the end of another structure
[-Wflex-array-member-not-at-end]
  114 |         struct qed_ll2_tx_queue tx_queue;
      |                                 ^~~~~~~~
```

-Wflex-array-member-not-at-end

GCC new compiler option (coming soon in GCC 14)

- **It works just fine. :)**

```
drivers/net/ethernet/qlogic/qed/qed_ll2.h:

100 struct qed_ll2_info {
...
114         struct qed_ll2_tx_queue tx_queue;
115         struct qed_ll2_cbs cbs;
116 };

In file included from drivers/net/ethernet/qlogic/qed/qed_dev.c:33:
drivers/net/ethernet/qlogic/qed/qed_ll2.h:114:33: warning: structure
containing a flexible array member is not at the end of another structure
[-Wflex-array-member-not-at-end]
  114 |         struct qed_ll2_tx_queue tx_queue;
      |                                 ^~~~~~~~
```

We should probably try to enable
-Wflex-array-member-not-at-end in mainline…
(thinkingface)

Gaining bounds-checking on trailing arrays

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

A bit of history. :)

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations
- Common use of **memcpy()** and flex arrays.

```c
struct flex_struct {
        ...
        size_t count;
        struct foo flex_array[];
} *p;

...

memcpy(p->flex_array, &source, SOME_SIZE);
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

- **__builtin_object_size() was** used to determine the size of both **source** and **destination**.

- Under CONFIG_FORTIFY_SOURCE=**y**

```
__FORTIFY_INLINE void *memcpy(void *dst, const void *src, size_t size)
{
        size_t dst_size = __builtin_object_size(dst, 1);
        size_t src_size = __builtin_object_size(src, 1);

        if (__builtin_constant_p(size)) {      /* Compile-time */
                if (dst_size < size)
                        __write_overflow();
                if (src_size < size)
                        __read_overflow2();
        }
        ...
}
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

- **__builtin_object_size() was** used to determine the size of both **source** and **destination**.
- Under CONFIG_FORTIFY_SOURCE=**y**

```
__FORTIFY_INLINE void *memcpy(void *dst, const void *src, size_t size)
{
        size_t dst_size = __builtin_object_size(dst, 1);
        size_t src_size = __builtin_object_size(src, 1);

        if (__builtin_constant_p(size)) {       /* Compile-time */
                if (dst_size < size)
                        __write_overflow();
                if (src_size < size)
                        __read_overflow2();
        }
        ...
}
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations
- Common use of **memcpy()** and flex arrays.

```c
struct flex_struct {
        ...
        size_t count;
        struct foo flex_array[];
} *p;

...

memcpy(p->flex_array, &source, SOME_SIZE);
```

Gaining bounds-checking on trailing arrays
Hardening **memcpy()** and flexible-array transformations

```c
memcpy(p->flex_array, &source, SOME_SIZE);

__FORTIFY_INLINE void *memcpy(void *dst, const void *src, size_t size)
{
        size_t dst_size = __builtin_object_size(dst, 1);
        ...
        if (__builtin_constant_p(size)) {     /* Compile-time */
                if (dst_size < size)
                        __write_overflow();
            ...
        }
        ...
}
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

```c
memcpy(p->flex_array, &source, SOME_SIZE);

__FORTIFY_INLINE void *memcpy(void *dst, const void *src, size_t size)
{
        size_t dst_size = __builtin_object_size(dst, 1);
        ...
        if (__builtin_constant_p(size)) {     /* Compile-time */
                if (dst_size < size)
                        __write_overflow();
                ...
        }
        ...
}

__builtin_object_size(p->flex_array, 1) == -1  /* flex-array size? */
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

– **__builtin_object_size()** and flexible arrays

```
__builtin_object_size(flex_struct->flex_array_member, 1) == -1
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

– **__builtin_object_size()** and flexible arrays

- Returns **-1** if cannot determine the size of the object.

```
__builtin_object_size(flex_struct->flex_array_member, 1) == -1
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

- – __**builtin_object_size()** and flexible arrays
  - Returns **-1** if cannot determine the size of the object.

`__builtin_object_size`(flex_struct->**flex_array_member**, 1) == **-1**

The size of a flexible-array member cannot be determined

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

– **__builtin_object_size()** and flexible arrays

  • Returns **-1** if cannot determine the size of the object.

__builtin_object_size(flex_struct->**flex_array_member**, 1) == **-1**

The size of a flexible-array member cannot be determined -- **it's an object of incomplete type**.

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

- **__builtin_object_size()** and flexible arrays
  - Returns **-1** if cannot determine the size of the object.
  - The size of a flexible-array member cannot be determined (**it's an object of incomplete type**).

OK; but, what about **fake** flexible arrays?
Those do have a size.

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

- – __**builtin_object_size()** and flexible arrays

```
__builtin_object_size(flex_struct->one_element_array, 1) ==
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

– **__builtin_object_size()** and flexible arrays

```
__builtin_object_size(flex_struct->one_element_array, 1) == -1
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

– **__builtin_object_size()** and flexible arrays

```
__builtin_object_size(flex_struct->one_element_array, 1) == -1

__builtin_object_size(flex_struct->zero_length_array, 1) ==
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

– **__builtin_object_size()** and flexible arrays

```
__builtin_object_size(flex_struct->one_element_array, 1) == -1

__builtin_object_size(flex_struct->zero_length_array, 1) == -1
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

– __**builtin_object_size()** and flexible arrays

```
__builtin_object_size(flex_struct->one_element_array, 1) == -1

__builtin_object_size(flex_struct->zero_length_array, 1) == -1

__builtin_object_size(flex_struct->flex_array_member, 1) == -1
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

– **__builtin_object_size()** and flexible arrays

```
__builtin_object_size(flex_struct->one_element_array, 1) == -1

__builtin_object_size(flex_struct->zero_length_array, 1) == -1

__builtin_object_size(flex_struct->flex_array_member, 1) == -1
```

It's not able to reason about the size of the **fake** flex arrays either. Returns **-1** for **all three** cases.

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

- **__builtin_object_size()** and flexible arrays
  - Returns **-1** for **all** three cases.
  - It **doesn't know** the size of the **fake** flex arrays either.

```
__builtin_object_size(flex_struct->one_element_array, 1) == -1
__builtin_object_size(flex_struct->zero_length_array, 1) == -1
__builtin_object_size(flex_struct->flex_array_member, 1) == -1

sizeof(flex_struct->one_element_array) == size-of-element-type
sizeof(flex_struct->zero_length_array) == 0
sizeof(flex_struct->flex_array_member) == ? /* Error */
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

– **__builtin_object_size()** and flexible arrays

- Returns **-1** for **all** three cases.
- It **doesn't know** the size of the **fake** flex arrays either.
- A bit **confusing**, isn't it?

```
__builtin_object_size(flex_struct->one_element_array, 1) == -1
__builtin_object_size(flex_struct->zero_length_array, 1) == -1
__builtin_object_size(flex_struct->flex_array_member, 1) == -1

sizeof(flex_struct->one_element_array) == size-of-element-type
sizeof(flex_struct->zero_length_array) == 0
sizeof(flex_struct->flex_array_member) == ? /* Error */
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

- – __**builtin_object_size()** and flexible arrays
    - Returns **-1** for **all** three cases.
    - It **doesn't know** the size of the **fake** flex arrays either.
    - A bit **confusing**, isn't it?

```
__builtin_object_size(flex_struct->one_element_array, 1) == -1
__builtin_object_size(flex_struct->zero_length_array, 1) == -1
__builtin_object_size(flex_struct->flex_array_member, 1) == -1

sizeof(flex_struct->one_element_array) == size-of-element-type
sizeof(flex_struct->zero_length_array) == 0
sizeof(flex_struct->flex_array_member) == ? /* Error */

__builtin_object_size(any_struct->any_trailing_array, 1) ==
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

– **__builtin_object_size()** and flexible arrays

- Returns **-1** for **all** three cases.
- It **doesn't know** the size of the **fake** flex arrays either.
- A bit **confusing**, isn't it?

```
__builtin_object_size(flex_struct->one_element_array, 1) == -1
__builtin_object_size(flex_struct->zero_length_array, 1) == -1
__builtin_object_size(flex_struct->flex_array_member, 1) == -1

sizeof(flex_struct->one_element_array) == size-of-element-type
sizeof(flex_struct->zero_length_array) == 0
sizeof(flex_struct->flex_array_member) == ? /* Error */

__builtin_object_size(any_struct->any_trailing_array, 1) == -1
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

- __**builtin_object_size()** and flexible arrays

What is going on?!

```
__builtin_object_size(any_struct->any_trailing_array, 1) == -1
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

– **__builtin_object_size()** and flexible arrays

In this scenario *memcpy*() is not able to sanity-check trailing arrays at all.

```
__builtin_object_size(any_struct->any_trailing_array, 1) == -1
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

– __**builtin_object_size()** and flexible arrays

A case for:

"Go fix the compiler!"

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

– __**builtin_object_size()** and flexible arrays

But why, exactly?

```
__builtin_object_size(any_struct->any_trailing_array, 1) == -1
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

- **__builtin_object_size()** and flexible arrays
- BSD **sockaddr** (sys/socket.h)
  - char sa_data[**14**]
  - #define SOCK_MAXADDRLEN **255**

```c
/*
 * Structure used by kernel to store most
 * addresses.
 */
struct sockaddr {
        unsigned char   sa_len;        /* total length */
        sa_family_t     sa_family;     /* address family */
        char            sa_data[14];   /* actually longer; address value */
};
#define SOCK_MAXADDRLEN    255         /* longest possible addresses */
```

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

- **__builtin_object_size()** and flexible arrays

- https://reviews.llvm.org/D126864

"Some code consider that **trailing** arrays are **flexible**, **whatever** their **size**.
Support for these **legacy** code has been introduced in
f8f632498307d22e10fab0704548b270b15f1e1e but **it prevents evaluation of
builtin_object_size and builtin_dynamic_object_size in some legit cases**."

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

– **__builtin_object_size()** and flexible arrays.

So, what do we do?

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

- **Kernel**: Make flexible-array declarations **unambiguous**.

  - Get rid of **fake** flexible arrays.

  - Only C99 **flexible-array members** should be used as flexible arrays.

Gaining bounds-checking on trailing arrays

Hardening **memcpy()** and flexible-array transformations

- **Kernel**: Make flexible-array declarations **unambiguous**.
  - Get rid of **fake** flexible arrays.
  - Only C99 **flexible-array members** should be used as flexible arrays.
- **Compiler**: Fix it.
  - Fix __**builtin_object_size()**
  - Add new option **-fstrict-flex-arrays**[=n]

Gaining bounds-checking on trailing arrays

-fstrict-flex-arrays[=n] – Released in **GCC-13** and **Clang-16**.

Gaining bounds-checking on trailing arrays

-fstrict-flex-arrays[=n] – Released in **GCC-13** and **Clang-16.**

- -fstrict-flex-arrays**=0 (default)**

Gaining bounds-checking on trailing arrays

-fstrict-flex-arrays[=n] – Released in **GCC-13** and **Clang-16.**

- -fstrict-flex-arrays**=0 (default)**
  - **All** trailing arrays are treated as flex arrays.

```
__builtin_object_size(any_struct->any_trailing_array, 1) == -1
```

Gaining bounds-checking on trailing arrays

-fstrict-flex-arrays[=n] – Released in **GCC-13** and **Clang-16.**

- -fstrict-flex-arrays**=0 (default)**
  - **All** trailing arrays are treated as flex arrays.

```
__builtin_object_size(any_struct->any_trailing_array, 1) == -1
```

Everything remains the **same**.

Gaining bounds-checking on trailing arrays

-fstrict-flex-arrays[=n] – Released in **GCC-13** and **Clang-16.**

- -fstrict-flex-arrays**=1**

Gaining bounds-checking on trailing arrays

-fstrict-flex-arrays[=n] – Released in **GCC-13** and **Clang-16.**

- -fstrict-flex-arrays**=1**

  - Only **[1]**, **[0]** and **[ ]** are treated as flex arrays.

```
__builtin_object_size(flex_struct->one_element_array, 1) == -1
__builtin_object_size(flex_struct->zero_length_array, 1) == -1
__builtin_object_size(flex_struct->flex_array_member, 1) == -1
```

Gaining bounds-checking on trailing arrays

-fstrict-flex-arrays[=n] – Released in **GCC-13** and **Clang-16.**

– -fstrict-flex-arrays**=1**

- Only **[1]**, **[0]** and **[ ]** are treated as flex arrays.

```
__builtin_object_size(flex_struct->one_element_array, 1) == -1
__builtin_object_size(flex_struct->zero_length_array, 1) == -1
__builtin_object_size(flex_struct->flex_array_member, 1) == -1
```

Now **fixed-size** trailing arrays (except [1] & [0], of course) **gain** bounds-checking. :)

Gaining bounds-checking on trailing arrays

-fstrict-flex-arrays[=n] – Released in **GCC-13** and **Clang-16.**

- -fstrict-flex-arrays**=2**

Gaining bounds-checking on trailing arrays

-fstrict-flex-arrays[=n] – Released in **GCC-13** and **Clang-16.**

- -fstrict-flex-arrays**=2**
  - Only **[0]** and **[ ]** are treated as flex arrays.

```
__builtin_object_size(flex_struct->zero_length_array, 1) == -1
__builtin_object_size(flex_struct->flex_array_member, 1) == -1
```

Gaining bounds-checking on trailing arrays

-fstrict-flex-arrays[=n] – Released in **GCC-13** and **Clang-16.**

- -fstrict-flex-arrays**=2**

  - Only **[0]** and **[ ]** are treated as flex arrays.

```
__builtin_object_size(flex_struct->zero_length_array, 1) == -1
__builtin_object_size(flex_struct->flex_array_member, 1) == -1
```

Now **fixed-size** trailing arrays (except [0], of course) **gain** bounds-checking. :)

Gaining bounds-checking on trailing arrays

-fstrict-flex-arrays[=n] – Released in **GCC-13** and **Clang-16.**

Now what's left to be resolved is the case for **zero-length arrays**.

Gaining bounds-checking on trailing arrays

-fstrict-flex-arrays[=n] – Released in **GCC-13** and **Clang-16.**

Now what's left to be resolved is the case for **zero-length arrays**.

Could that probably be resolved with

-fstrict-flex-arrays=3 ? Maybe?

The case of **Clang** vs -fstrict-flex-arrays**=3**

The case of **Clang** vs -fstrict-flex-arrays**=3**

- **-Wzero-length-array** (tons of warnings, as usual)

The case of **Clang** vs -fstrict-flex-arrays**=3**

- **-Wzero-length-array** (tons of warnings, as usual)
- 0-length arrays are not only used as fake flex-arrays.
- They are used as markers in structures.

The case of **Clang** vs -fstrict-flex-arrays**=3**

- **-Wzero-length-array** (tons of warnings, as usual)
- 0-length arrays are not only used as fake flex-arrays.
- They are used as markers in structures.

```
struct marked {
    ...
    uint8_t hdr_start[0];

    ... some other members

    uint8_t hdr_end[0];
    ...
 } *p;
...
memcpy(p->start, &source,
       offsetof(struct marked, hdr_end) – offsetof(struct marked, hdr_start));
```

The case of **Clang** vs -fstrict-flex-arrays**=3**

- **-Wzero-length-array** (tons of warnings, as usual)
- 0-length arrays are not only used as fake flex-arrays.
- They are used as markers in structures.

```
struct marked {
    ...
-   uint8_t hdr_start[0];
+   struct_group(hdr,

    ... some other members

-   uint8_t hdr_end[0];
+   );
    ...
 } *p;
...
            memcpy(p->start, &source, sizeof(p->hdr));
```

The case of **Clang** vs -fstrict-flex-arrays**=3**

- **-Wzero-length-array** (tons of warnings, as usual)
- 0-length arrays are not only used as fake flex-arrays.
- They are used as markers in structures.
- Under certain configurations some arrays end up having a size zero.

The case of **Clang** vs -fstrict-flex-arrays**=3**

- **-Wzero-length-array** (tons of warnings, as usual)
- 0-length arrays are not only used as fake flex-arrays.
- They are used as markers in structures.
- Under certain configurations some arrays end up having a size zero.
- **So, 0-length arrays are here to stay, but not as VLOs.**

The case of **Clang** vs -fstrict-flex-arrays**=3**

- **-Wzero-length-array** (tons of warnings, as usual)
- 0-length arrays are not only used as fake flex-arrays.
- They are used as markers in structures.
- Under certain configurations some arrays end up having a size zero.
- **So, 0-length arrays are here to stay, but not as VLOs.**

Fortunately, that issue is now resolved. :)

Gaining bounds-checking on trailing arrays

- **-fstrict-flex-arrays[=n]** – Released in **GCC-13** and **Clang-16.**

Gaining bounds-checking on trailing arrays

- **-fstrict-flex-arrays[=n]** – Released in **GCC-13** and **Clang-16.**
  - -fstrict-flex-arrays**=3**
    - Only C99 flexible-array members (**[ ]**) are treated VLOs.

```
__builtin_object_size(flex_struct->flex_array_member, 1) == -1
```

Gaining bounds-checking on trailing arrays

- **-fstrict-flex-arrays[=n]** – Released in **GCC-13** and **Clang-16.**

  - -fstrict-flex-arrays**=3**

    - Only C99 flexible-array members (**[ ]**) are treated VLOs.

```
__builtin_object_size(flex_struct->flex_array_member, 1) == -1
```

Now **ALL** trailing arrays of **fixed-size gain** bounds-checking. :D

Gaining bounds-checking on trailing arrays

- **-fstrict-flex-arrays[=n]** – Released in **GCC-13** and **Clang-16.**
  - -fstrict-flex-arrays**=3**
    - Only C99 flexible-array members (**[ ]**) are treated VLOs.

```
__builtin_object_size(flex_struct->flex_array_member, 1) == -1
```

Now **ALL** trailing arrays of **fixed-size gain** bounds-checking. :D

This is what we **need**!

Gaining bounds-checking on trailing arrays

Fortified **memcpy()** and **-fstrict-flex-arrays=3**

- Globally enabled in **Linux 6.5**. Yeeiii!!

Gaining bounds-checking on trailing arrays

Fortified **memcpy()** and **-fstrict-flex-arrays=3**

- Globally enabled in **Linux 6.5**. Yeeiii!!

- CONFIG_UBSAN_BOUNDS and CONFIG_FORTIFY_SOURCE benefit from this.

- Only C99 flexible-array members are considered to be dynamically sized.

Gaining bounds-checking on trailing arrays

Fortified **memcpy()** and **-fstrict-flex-arrays=3**

- Globally enabled in **Linux 6.5**. Yeeiii!!

- CONFIG_UBSAN_BOUNDS and CONFIG_FORTIFY_SOURCE benefit from this.

- Only C99 flexible-array members are considered to be dynamically sized.

- **Therefore, we've gained bounds-checking on trailing arrays of fixed-size.**

Gaining bounds-checking on trailing arrays

Great, but what about bounds-checking

on **flexible-array members**?

Gaining bounds-checking on trailing arrays

We need a new attribute

Gaining bounds-checking on trailing arrays

We need a new attribute

- How about __attribute__((__**counted_by**__(member))) ?

```c
struct bounded_flex_struct {
    ...
    size_t elements;
    struct foo array[] __attribute__((counted_by(elements)));
};
```

Gaining bounds-checking on trailing arrays

We need a new attribute

- How about __attribute__((__**counted_by**__(member))) ?
- Coming soon in **GCC-14**

```
#if __has_attribute(__counted_by__)
# define __counted_by(member) __attribute__((__counted_by__(member)))
#else
# define __counted_by(member)
#endif
```

Gaining bounds-checking on trailing arrays

We need a new attribute

- How about __attribute__((__**counted_by**__(member))) ?

- Coming soon in **GCC-14**

- Just released in **Clang-18 !!!**

Gaining bounds-checking on trailing arrays

We need a new attribute

- How about __attribute__((__**counted_by**__(member))) ?

- Coming soon in **GCC-14**

- Just released in **Clang-18 !!!**

- https://releases.llvm.org/18.1.0/tools/clang/docs/ReleaseNotes.html

"Clang now supports the **C-only attribute counted_by**. When applied to a struct's flexible array member, it points to the struct field that holds **the number of elements in the flexible array** member. This information can improve the results of the array bound sanitizer and the **__builtin_dynamic_object_size** builtin."

Gaining bounds-checking on trailing arrays

We need a new attribute

- How about __attribute__((__**counted_by**__(member))) ?

- Coming soon in **GCC-14**

- Just released in **Clang-18 !!!**

```c
struct bounded_flex_struct {
    ...
    size_t elements;
    struct foo array[] __counted_by(elements);
};
```

Gaining bounds-checking on trailing arrays

A quick recap

Gaining bounds-checking on trailing arrays

A quick recap

- flexible-array transformations mostly done.

- -fstrict-flex-arrays=3 enabled.

Gaining bounds-checking on trailing arrays

A quick recap

- – flexible-array transformations mostly done.

- – -fstrict-flex-arrays=3 enabled.

    - • bounds-checking on fixed-size arrays.

Gaining bounds-checking on trailing arrays

A quick recap

- – flexible-array transformations mostly done.
- – -fstrict-flex-arrays=3 enabled.
  - • bounds-checking on fixed-size arrays.
- – __counted_by() just released in Clang and soon in GCC.
  - • Improves the results of __builtin_dynamic_object_size().

So, the next sensible thing to do is...

Gaining bounds-checking on trailing arrays

Fortified **memcpy()** and **__builtin_dynamic_object_size**()

Gaining bounds-checking on trailing arrays

Fortified **memcpy()** and **__builtin_dynamic_object_size**()

- __bdos() **replaced** __builtin_object_size()

- __bdos() adds **run-time coverage** whereas __bos() only covers **compile-time**.

- It **gets hints from** __alloc_size__ and from __counted_by()

- Greater fortification for memcpy().

The case of UAPI

The case of UAPI

One-element arrays in UAPI – First attempts.
- Duplicate the original struct within a union.
- Flexible-array will be used by **kernel-space**.
- One-element array will be used by **user-space**.

```
 struct ip_msfilter {
-          __be32          imsf_multiaddr;
-          __be32          imsf_interface;
-          __u32           imsf_fmode;
-          __u32           imsf_numsrc;
-          __be32          imsf_slist[1];
+          union {
+                  struct {
+                          __be32          imsf_multiaddr_aux;
+                          __be32          imsf_interface_aux;
+                          __u32           imsf_fmode_aux;
+                          __u32           imsf_numsrc_aux;
+                          __be32          imsf_slist[1];
+                  };
+                  struct {
+                          __be32          imsf_multiaddr;
+                          __be32          imsf_interface;
+                          __u32           imsf_fmode;
+                          __u32           imsf_numsrc;
+                          __be32          imsf_slist_flex[];
+                  };
+          };
 };
```

The case of UAPI

One-element arrays in UAPI – Better code.

– Just use the **__DECLARE_FLEX_ARRAY**() helper in a union.

```c
struct ip_msfilter {
        __be32          imsf_multiaddr;
        __be32          imsf_interface;
        __u32           imsf_fmode;
        __u32           imsf_numsrc;
        union {
                __be32          imsf_slist[1];
                __DECLARE_FLEX_ARRAY(__be32, imsf_slist_flex);
        };
};
```

The case of UAPI

One-element arrays in UAPI – Better code.

- Just use the **__DECLARE_FLEX_ARRAY**() helper in a union.
- The bad news is that the **sizeof**(flex_struct) will remain the same.

```
struct ip_msfilter {
        __be32          imsf_multiaddr;
        __be32          imsf_interface;
        __u32           imsf_fmode;
        __u32           imsf_numsrc;
        union {
                __be32          imsf_slist[1];
                __DECLARE_FLEX_ARRAY(__be32, imsf_slist_flex);
        };
};
```

The case of UAPI

One-element arrays in UAPI – Even better code.
- Soon, __**DECLARE_FLEX_ARRAY**() won't be needed anymore.

The case of UAPI

One-element arrays in UAPI – Even better code.

- Soon, __**DECLARE_FLEX_ARRAY**() won't be needed anymore.
- FAMs will be allowed in unions and alone in structs in **Clang 19**.
- https://github.com/llvm/llvm-project/pull/84428

```c
struct ip_msfilter {
        __be32              imsf_multiaddr;
        __be32              imsf_interface;
        __u32               imsf_fmode;
        __u32               imsf_numsrc;
        union {
                __be32 imsf_slist[1];
                __be32 imsf_slist_flex[];
        };
};
```

Bleeding-edge upstream kernel hardening

Bleeding-edge upstream kernel hardening

-Wflex-array-member-not-at-end

Bleeding-edge upstream kernel hardening

-Wflex-array-member-not-at-end

```c
struct flex_struct {
    ...
    size_t count;
    struct something flex_array[] __counted_by(count);
};

struct foo {
    ...

    struct flex_struct flex_in_the_middle; /* suspicious ¬.¬ */

    ...
};
```

Bleeding-edge upstream kernel hardening

-Wflex-array-member-not-at-end

- We had ~60K warnings in total. Only **650** unique.

```
struct flex_struct {
    ...
    size_t count;
    struct something flex_array[] __counted_by(count);
};

struct foo {
    ...

    struct flex_struct flex_in_the_middle; /* suspicious ¬.¬ */

    ...
};
```

-Wflex-array-member-not-at-end

Case 1: Some FAMs not actually being used.

-Wflex-array-member-not-at-end

Case 1: Some FAMs not actually being used.

- – f4b09b29f8b4 ("wifi: ti: Avoid a hundred -Wflex-array-member-not-at-end warnings")

```
struct wl1251_cmd_header {
    u16 id;
    u16 status;
-   /* payload */
-   u8 data[];
  } __packed;
```

-Wflex-array-member-not-at-end

Case 2: FAMs never accessed. Only the rest of the members in the flex struct are used.

-Wflex-array-member-not-at-end

Case 2: FAMs never accessed. Only the rest of the members in the flex struct are used.

– Two separate structs: original struct & header struct

```c
struct flex_struct { /* original struct */
    ...
    size_t count;
    struct something flex_array[] __counted_by(count);
};
```

-Wflex-array-member-not-at-end

Case 2: FAMs never accessed. Only the rest of the members in the flex struct are used.

– Two separate structs: original struct & header struct

```c
struct flex_struct { /* original struct */
    ...
    size_t count;
    struct something flex_array[] __counted_by(count);
};

struct flex_struct_hdr { /* header struct */
    ...
    size_t count;
};
```

-Wflex-array-member-not-at-end

Case 2: FAMs never accessed. Only the rest of the members in the flex struct are used.

– Two separate structs: original struct & header struct

```
struct foo { /* before */
    ...
    struct flex_struct middle; /* suspicious ¬.¬ */
    ...
};

struct foo { /* after */
    ...
    struct flex_struct_hdr middle; /* OK! */
    ...
};
```

-Wflex-array-member-not-at-end

Case 2: FAMs never accessed. Only the rest of the members in the flex struct are used.

- – Use struct_group_tagged()/__struct_group()

-Wflex-array-member-not-at-end

Case 2: FAMs never accessed. Only the rest of the members in the flex struct are used.

– Use struct_group_tagged()/__struct_group()

```c
struct flex_struct {


    ...
    size_t count;

    struct something flex_array[] __counted_by(count);
};

struct foo { /* before */
    ...
    struct flex_struct middle; /* suspicious ¬.¬ */
    ...
};
```

-Wflex-array-member-not-at-end

Case 2: FAMs never accessed. Only the rest of the members in the flex struct are used.

- Use struct_group_tagged()/__struct_group()

```c
struct flex_struct {
    /* New members must be added within the struct_group() macro below. */
    struct_group_tagged(flex_struct_hdr, hdr,
        ...
        size_t count;
    );
    struct something flex_array[] __counted_by(count);
};

struct foo { /* after */
    ...
    struct flex_struct_hdr middle; /* OK! */
    ...
};
```

-Wflex-array-member-not-at-end

Case 2: FAMs never accessed. Only the rest of the members in the flex struct are used.

– 5c4250092fad ("wifi: mwl8k: Avoid -Wflex-array-…")

```
  struct mwl8k_cmd_pkt {
-     __le16  code;
-     __le16  length;
-     __u8    seq_num;
-     __u8    macid;
-     __le16  result;
+     __struct_group(mwl8k_cmd_pkt_hdr, hdr, __packed,
+         __le16  code;
+         __le16  length;
+         __u8    seq_num;
+         __u8    macid;
+         __le16  result;
+     );
      char payload[];
  } __packed;
```

-Wflex-array-member-not-at-end

- 5c4250092fad ("wifi: mwl8k: Avoid -Wflex-array-...")
- Replace *mwl8k_cmd_pkt* with *mwl8k_cmd_pkt_hdr*

```
  struct mwl8k_cmd_rf_antenna {
-   struct mwl8k_cmd_pkt header;
+   struct mwl8k_cmd_pkt_hdr header;
    __le16 antenna;
    __le16 mode;
  } __packed;
```

-Wflex-array-member-not-at-end

Case 3: Implicit unions between FAMs and fixed-size arrays of the same element type.

-Wflex-array-member-not-at-end

Case 3: Implicit unions between FAMs and fixed-size arrays of the same element type.

```
 struct ima_digest_data { /* flexible struct */
+   /* New members must be added within the __struct_group() macro below. */
+   __struct_group(ima_digest_data_hdr, hdr, __packed,
    u8 algo;
    u8 length;
    union {
@@ -57,6 +62,7 @@ struct ima_digest_data {
        } ng;
        u8 data[2];
    } xattr;
+   );
    u8 digest[];
 } __packed;
```

```
                                    /* implicit union: FAM & fixed-size array*/
                                   struct ima_max_digest_data {
-                                      struct ima_digest_data hdr;
+                                      struct ima_digest_data_hdr hdr;
                                       u8 digest[HASH_MAX_DIGESTSIZE];
                                   } __packed;
```

-Wflex-array-member-not-at-end

Case 3: Implicit unions between FAMs and fixed-size arrays of the same element type.

- Implicit union: **FAM digest** and **fixed-size** array **digest share the same address**.

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-   struct ima_digest_data hdr;
+   struct ima_digest_data_hdr hdr;
    u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

-Wflex-array-member-not-at-end

Case 3: Implicit unions between FAMs and fixed-size arrays of the same element type.

- – Implicit union: **FAM digest** and **fixed-size** array **digest share the same address**.

- – However, **FAM digest** should be accessed at run-time.

```
/* implicit union: FAM & fixed-size array*/
 struct ima_max_digest_data {
-   struct ima_digest_data hdr;
+   struct ima_digest_data_hdr hdr;
    u8 digest[HASH_MAX_DIGESTSIZE];
 } __packed;
```

-Wflex-array-member-not-at-end

Case 3: Implicit unions between FAMs and fixed-size arrays of the same element type.

- Use **container_of()** to get a pointer to the flex struct.

- Access FAM through that pointer.

-Wflex-array-member-not-at-end

Case 3: Implicit unions between FAMs and fixed-size arrays of the same element type.

– Use **container_of()** to get a pointer to the flex struct.

– Access FAM through that pointer.

```
    struct ima_max_digest_data hash; /* struct with implicit union */
+   struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                       struct ima_digest_data, hdr);

... hash_hdr is now a pointer to flex struct ima_digest_data

        /* read data from the FAM digest */
-       memcpy(digest_hash, hash.hdr.digest, digest_hash_len);
+       memcpy(digest_hash, hash_hdr->digest, digest_hash_len);
```

-Wflex-array-member-not-at-end

Case 3: Implicit unions between FAMs and fixed-size arrays of the same element type.

– Use **container_of()** to get a pointer to the flex struct.

– Access FAM through that pointer.

– 38aa3f5ac6d2 ("integrity: Avoid -Wflex-array-member...")

```
    struct ima_max_digest_data hash; /* struct with implicit union */
+   struct ima_digest_data *hash_hdr = container_of(&hash.hdr,
+                     struct ima_digest_data, hdr);

... hash_hdr is now a pointer to flex struct ima_digest_data

        /* read data from the FAM digest */
-       memcpy(digest_hash, hash.hdr.digest, digest_hash_len);
+       memcpy(digest_hash, hash_hdr->digest, digest_hash_len);
```

-Wflex-array-member-not-at-end

Case 4: The same as case 3 but on-stack.

- For those we use DECLARE_FLEX() and DECLARE_RAW_FLEX() helpers.

- Some examples:
  - 6c85a13b133f ("platform/chrome: cros_ec_proto:…")
  - 4d69c58ef2e4 ("fsnotify: Avoid -Wflex-array-mem…")
  - 215c4704208b ("Bluetooth: L2CAP: Avoid -Wflex-…")

-Wflex-array-member-not-at-end

Patches in linux-next

- -Wflex-array-member-not-at-end patches in linux-next
- From 650 to **less than 400 warnings now!** :D (this is with a few more patches pending to send out).

Conclusions

Conclusions

- **-fstrict-flex-arrays=3** enabled in Linux 6.5

- **__counted_by()** attribute is a reality now.

- **__builtin_dynamic_object_size()** increased bounds-checking coverage.

- Gaining **bounds-checking** on **FAMs** is closer than ever!

- FORTIFY_SOURCE and UBSAN bounds-checking better every time.

Conclusions

- Vulnerabilities discovered over the last years could've been prevented with the most recent **memcpy()** and **FORTIFY_SOURCE** updates.

- We've been finding and fixing bugs in both **kernel-space** and **user-space**.

- We have a clear strategy to fix **-Wflex-array-member-not-at-end** warning and enable the option in mainline, soon.

- **The security of the kernel is being significantly improved. :)**

Thank you, Gladstone!

Gustavo A. R. Silva
gustavoars@kernel.org
fosstodon.org/@gustavoars