Upstream Kernel Hardening: Progress on enabling -Wflex-array-member-not-at-end

Gustavo A. R. Silva
gustavoars@kernel.org
https://embeddedor.com/blog/

Supported by
The Linux Foundation & Alpha-Omega

Everything Open
January 23, 2026
Canberra, Australia

Who am I?

By @shidokou

Who am I?

- **Upstream first** – 10th year.
- Upstream Linux Kernel Engineer.
    - Kernel hardening.
    - Proactive security.

By @shidokou

Who am I?

- **Upstream first** – 10th year.
- Upstream Linux Kernel Engineer.
    - Kernel hardening.
    - Proactive security.

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

By @shidokou

Agenda

- **Introduction**
  - C99 flexible-array members (FAMs)
  - The new *-Wflex-array-member-not-at-end* compiler option
- **The challenge of -Wflex-array-member-not-at-end**
  - What’s wrong with FAMs in the middle?
  - Fixing thousands of *-Wfamnae* warnings
- **Conclusions**

Quick review of C99 flexible-array members

- Should be the last member of a struct.

```
struct flex {
   ...
   size_t  count;
   struct foo fam[];
};
```

Quick review of C99 flexible-array members

- Should be the last member of a struct.
- ***struct flex*** *may not be a member of another struct.*

```
struct flex {
    ...
    size_t  count;
    struct foo fam[];
};
```

Quick review of C99 flexible-array members

- Should be the last member of a struct.
- ***struct flex*** *may not be a member of another struct.*
- The flex struct usually contains a ***counter*** member.

```
struct flex {
   ...
   size_t  count;
   struct foo fam[];
};
```

Quick review of C99 flexible-array members

- Should be the last member of a struct.
- ***struct flex*** *may not be a member of another struct.*
- The flex struct usually contains a **counter** member.

```
struct flex {
    ...
    size_t  count;
    struct foo fam[] __counted_by(count);
};
```

Quick review of C99 flexible-array members

- Should be the last member of a struct.
- ***struct flex*** *may not be a member of another struct.*
- The flex struct usually contains a **counter** member..
- Run-time bounds-checking coverage on FAMs. (blogpost)

```
struct flex {
    ...
    size_t  count;
    struct foo fam[] __counted_by(count);
};
```

The new -Wflex-array-member-not-at-end

- Developed by Qing Zhao (2023)
- Released in GCC 14

- GCC features to help harden the kernel (LWN.net article)

The new -Wflex-array-member-not-at-end

- Warns about FAMs in the middle of composite structs.

The new -Wflex-array-member-not-at-end

- Warns about FAMs in the middle of composite structs.

```
struct flex {
    ...
    size_t count;
    struct foo fam[] __counted_by(count);
};

struct composite {
    ...

    struct flex middle;

    ... more objects ...
};
```

The new -Wflex-array-member-not-at-end

- Warns about FAMs in the middle of composite structs.

```
struct flex {
    ...
    size_t count;
    struct foo fam[] __counted_by(count);
};

struct composite {
    ...

    struct flex middle;

    ... more objects ...
};
```

The new -Wflex-array-member-not-at-end

- Warns about FAMs in the middle of composite structs.

```
struct flex {
    ...
    size_t count;
    struct foo fam[] __counted_by(count);
};

struct composite {
    ...

    struct flex middle; /* -Wfamnae warning! */

    ... more objects ...
};
```

The new -Wflex-array-member-not-at-end

- Warns about FAMs in the middle of composite structs.

```
struct composite {
    ...
-   struct flex middle; /* -Wfamnae warning! */
    ... more objects ...
+   struct flex last;
};
```

The new -Wflex-array-member-not-at-end

- Warns about FAMs in the middle of composite structs.

```
struct composite {
    ...
-   struct flex middle; /* -Wfamnae warning! */
    ... more objects ...
+   struct flex last;   /* This is fine! */
};
```

The new -Wflex-array-member-not-at-end

- Warns about FAMs in the middle of composite structs.

```
struct composite {
    ...
-   struct flex middle; /* -Wfamnae warning! */
    ... more objects ...
+   struct flex last;   /* This is fine! */
};

struct another {
    ...

    struct composite middle;

    ... more objects ...
};
```

The new -Wflex-array-member-not-at-end

- Warns about FAMs in the middle of composite structs.

```
struct composite {
    ...
-   struct flex middle; /* -Wfamnae warning! */
    ... more objects ...
+   struct flex last;   /* This is fine! */
};

struct another {
    ...

    struct composite middle; /* -Wfamnae warning! */

    ... more objects ...
};
```

The challenge of enabling

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

What’s wrong with FAMs in the middle?

What’s wrong with FAMs in the middle?

- Flex struct in a composite struct **is an extension**.

What’s wrong with FAMs in the middle?

- Flex struct in a composite struct **is an extension**.
- The flex struct can be either:
  - the last member

```
struct composite {
    ...

    struct flex last;
};
```

What’s wrong with FAMs in the middle?

- Flex struct in a composite struct **is an extension**.
- The flex struct can be either:
  - the last member
  - **not the last member**

```
struct composite {
   ...

   struct flex last;
};
```

```
struct composite {
   ...

   struct flex middle;

   ... more objects ...
};
```

What’s wrong with FAMs in the middle?

- Flex struct in a composite struct **is an extension**.
- The flex struct can be either:
  - the last member
  - **not the last member – This is deprecated now.**

```
struct composite {
   ...

   struct flex last;
};
```

```
struct composite {
   ...

   struct flex middle;

   ... more objects ...
};
```

What’s wrong with FAMs in the middle?

- Flex struct in a composite struct **is an extension**.
- The flex struct can be either:
  - the last member
  - **not the last member – This is deprecated now.**

```
struct composite {
   ...

   struct flex last;
};
```

```
struct composite {
   ...

   struct flex middle;

   ... more objects ...
};
```

What's wrong with FAMs in the middle?

- "Compilers do not handle such a case consistently. **Any code relying on this case should be modified to ensure that flexible array members only end up at the ends of structures.**" -GCC Docs.

```
struct composite {
   ...

   struct flex middle; /* TODO: Fix me! ^-^ */

   ... more objects ...
};
```

What's wrong with FAMs in the middle?

- "Compilers do not handle such a case consistently. **Any code relying on this case should be modified to ensure that flexible array members only end up at the ends of structures.**" -GCC Docs.

```
struct composite {
    ...

    struct flex middle; /* TODO: Fix me! ^-^ */

    ... more objects ...
};
```

So, we have more than **60K**

-Wfamnae warnings to address

Fixing thousands of

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

Fixing thousands of -Wfamnae warnings in Linux

```
struct flex {
    ...
    size_t  count;
    struct foo fam[] __counted_by(count);
};

struct composite {
    ...
    struct flex middle; /* -Wfamnae warning */
    ... more objects ...
};
```

Fixing thousands of -Wfamnae warnings in Linux

- **Before** Flex-Array Transformations: 8,000 warnings

```
struct flex {
    ...
    size_t  count;
    struct foo fam[] __counted_by(count);
};

struct composite {
    ...
    struct flex middle; /* -Wfamnae warning */
    ... more objects ...
};
```

Fixing thousands of -Wfamnae warnings in Linux

- **Before** Flex-Array Transformations: 8,000 warnings
- **After** years of kernel hardening: more than 60,000 warnings

```
struct flex {
    ...
    size_t  count;
    struct foo fam[] __counted_by(count);
};

struct composite {
    ...
    struct flex middle; /* -Wfamnae warning */
    ... more objects ...
};
```

Fixing thousands of -Wfamnae warnings in Linux

- **Before** Flex-Array Transformations: 8,000 warnings
- **After** years of kernel hardening: more than 60,000 warnings
- 650 unique issues

```
struct flex {
    ...
    size_t  count;
    struct foo fam[] __counted_by(count);
};

struct composite {
    ...
    struct flex middle; /* -Wfamnae warning */
    ... more objects ...
};
```

Different cases of

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

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

Case 1: **FAM never actually used.**

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

Case 1: **FAM never actually used.**

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

struct cmd_read_write_memory {
        struct wl1251_cmd_header header; /* -Wfamnae warning */

        u32 addr;
        u32 size;
        u8 value[MAX_READ_SIZE];
} __packed;
```

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

Case 1: **FAM never actually used.**

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

struct cmd_read_write_memory {
        struct wl1251_cmd_header header; /* -Wfamnae warning */

        u32 addr;
        u32 size;
        u8 value[MAX_READ_SIZE];
} __packed;
```

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

Case 1: **FAM never actually used.**

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

struct cmd_read_write_memory {
        struct wl1251_cmd_header header; /* -Wfamnae warning */

        u32 addr;
        u32 size;
        u8 value[MAX_READ_SIZE];
} __packed;
```

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

Case 1: **FAM never actually used.**

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

struct cmd_read_write_memory {
        struct wl1251_cmd_header header; /* -Wfamnae warning */

        u32 addr;
        u32 size;
        u8 value[MAX_READ_SIZE];
} __packed;
```

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

Case 1: **FAM never actually used.**

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

struct cmd_read_write_memory {
        struct wl1251_cmd_header header; /* -Wfamnae warning */

        u32 addr;
        u32 size;
        u8 value[MAX_READ_SIZE];
} __packed;
```

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

Case 1: **FAM never actually used.**

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

struct cmd_read_write_memory {
        struct wl1251_cmd_header header; /* -Wfamnae warning */

        u32 addr;
        u32 size;
        u8 value[MAX_READ_SIZE];
} __packed;
```

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

Case 1: **FAM never actually used.**

- No heap space is allocated for them anywhere.

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

struct cmd_read_write_memory {
        struct wl1251_cmd_header header; /* -Wfamnae warning */

        u32 addr;
        u32 size;
        u8 value[MAX_READ_SIZE];
} __packed;
```

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

Case 1: **FAM never actually used.**

- f4b09b29f8b4 (“wifi: ti: Avoid a hundred -Wflex-array...”)

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

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

Case 2: **FAM never accessed through the composite struct**.

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

Case 2: **FAM never accessed through the composite struct**.

```
struct flex {
    int a;
    int b;
    size_t  count;
    struct foo fam[];
};

struct composite {
    ...
    struct flex middle; /* -Wfamnae warning */
    ...
} *p;
...


do_something_with(p->middle.a, p->middle.b);
```

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

Case 2: **FAM never accessed through the composite struct**.

```
struct flex {
    int a;
    int b;
    size_t  count;
    struct foo fam[];
};

struct composite {
    ...
    struct flex middle; /* -Wfamnae warning */
    ...
} *p;
...

/* We may access the rest of the members in struct flex */
do_something_with(p->middle.a, p->middle.b);
```

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

Case 2: **FAM never accessed through the composite struct**.

```
struct flex {
    int a;
    int b;
    size_t  count;
    struct foo fam[];
};

struct composite {
    ...
    struct flex middle; /* -Wfamnae warning */
    ...
} *p;
...

/* We may access the rest of the members in struct flex */
do_something_with(p->middle.a, p->middle.b);
```

But something like this

```
... p->middle.fam ...
```

never actually occurs.

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

Case 2: **FAM never accessed through the composite struct**.

- What can we do about it?

```
struct flex {
    int a; int b;
    size_t count;
    struct foo fam[];
};

struct composite {
    ...
    struct flex middle; /* -Wfamnae warning */
    ...
};
```

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

Case 2: **FAM never accessed through the composite struct**.

```
struct flex_hdr { /* All members in struct flex except the FAM */
    int a; int b;
    size_t count;
};

struct flex { /* original struct */
    struct flex_hdr  hdr;
    struct foo fam[] __counted_by(.hdr.count);
};
```

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

Case 2: **FAM never accessed through the composite struct**.

```
struct composite { /* BEFORE */
    ...
    struct flex middle; /* -Wfamnae warning :( */
    ...
};

struct composite { /* AFTER */
    ...
    struct flex_hdr middle;
    ...
};
```

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

Case 2: **FAM never accessed through the composite struct**.

```
struct composite { /* BEFORE */
    ...
    struct flex middle; /* -Wfamnae warning :( */
    ...
};

struct composite { /* AFTER */
    ...
    struct flex_hdr middle;
    ...
};
```

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

Case 2: **FAM never accessed through the composite struct**.

```
struct composite { /* BEFORE */
    ...
    struct flex middle; /* -Wfamnae warning :( */
    ...
};

struct composite { /* AFTER */
    ...
    struct flex_hdr middle; /* Life's beautiful! ^.^ */
    ...
};
```

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

Case 2: **FAM never accessed through the composite struct**.

- The “counter” cannot be in a non-anonymous sub struct.

```
struct flex_hdr { /* All members in struct flex except the FAM */
    int a; int b;
    size_t count;
};

struct flex { /* original struct */
    struct flex_hdr  hdr;
    struct foo fam[] __counted_by(.hdr.count);
};
```

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

Case 2: **FAM never accessed through the composite struct**.

- The “counter” cannot be in a non-anonymous sub struct.

```
struct flex_hdr { /* All members in struct flex except the FAM */
    int a; int b;
    size_t count;
};

struct flex { /* original struct */
    struct flex_hdr  hdr;
    struct foo fam[] __counted_by(.hdr.count);
};
```

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

Case 2: **FAM never accessed through the composite struct**.

- The “counter” cannot be in a non-anonymous sub struct.
- __counted_by(.hdr.count); is not supported **yet**.

```
struct flex_hdr { /* All members in struct flex except the FAM */
    int a; int b;
    size_t count;
};

struct flex { /* original struct */
    struct flex_hdr  hdr;
    struct foo fam[] __counted_by(.hdr.count);
};
```

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

Case 2: **FAM never accessed through the composite struct**.

- Use struct_group_tagged()/__struct_group()

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

Case 2: **FAM never accessed through the composite struct**.

- Use struct_group_tagged()/__struct_group()

```
struct flex { /* BEFORE */


    int a; int b;
    size_t  count;

    struct foo fam[] __counted_by(count);
};

struct composite { /* BEFORE */
    ...
    struct flex middle; /* -Wfamnae warning */
    ...
} *p;
```

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

Case 2: **FAM never accessed through the composite struct**.

- Use struct_group_tagged()/__struct_group()

```
struct flex { /* AFTER */
    /* New members must be added within the struct_group() macro below. */
    struct_group_tagged(flex_hdr, hdr,
        int a; int b;
        size_t  count;
    );
    struct foo fam[] __counted_by(count);
};

struct composite { /* BEFORE */
    ...
    struct flex middle; /* -Wfamnae warning */
    ...
} *p;
```

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

Case 2: **FAM never accessed through the composite struct**.

- Use struct_group_tagged()/__struct_group()

```
struct flex { /* AFTER */
    /* New members must be added within the struct_group() macro below. */
    struct_group_tagged(flex_hdr, hdr,
        int a; int b;
        size_t  count;
    );
    struct foo fam[] __counted_by(count);
};

struct composite { /* AFTER */
    ...
    struct flex_hdr middle; /* FAM is gone! ^.^ */
    ...
} *p;
```

The **struct_group**() family of helpers

```
#define struct_group_tagged(TAG, NAME, MEMBERS...) \
        union {                                    \
                struct { MEMBERS };                \
                struct TAG { MEMBERS } NAME;       \
        }
```

The **struct_group**() family of helpers

```
#define struct_group_tagged(TAG, NAME, MEMBERS...) \
        union {                                    \
                struct { MEMBERS };                \
                struct TAG { MEMBERS } NAME;       \
        }
```

The **struct_group**() family of helpers

- Access each member directly or via the named struct.

```
#define struct_group_tagged(TAG, NAME, MEMBERS...) \
        union {                                    \
                struct { MEMBERS };                \
                struct TAG { MEMBERS } NAME;       \
        }
```

The **struct_group**() family of helpers

- Access each member directly or via the named struct.
- Creates a new struct type and define an identifier for the group

```
#define struct_group_tagged(TAG, NAME, MEMBERS...) \
        union {                                    \
                struct { MEMBERS };                \
                struct TAG { MEMBERS } NAME;       \
        }
```

The **struct_group**() family of helpers

- Access each member directly or via the named struct.
- Creates a new struct type and define an identifier for the group – via which we can even gain bounds-checking.

```
#define struct_group_tagged(TAG, NAME, MEMBERS...) \
        union {                                    \
                struct { MEMBERS };                \
                struct TAG { MEMBERS } NAME;       \
        }
```

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

Case 2: **FAM never accessed through the composite struct**.

- Use struct_group_tagged()/__struct_group()

```
struct flex { /* AFTER */
    /* New members must be added within the struct_group() macro below. */
    struct_group_tagged(flex_hdr, hdr,
        int a; int b;
        size_t  count;
    );
    struct foo fam[] __counted_by(count);
};

struct composite { /* AFTER */
    ...
    struct flex_hdr middle; /* FAM is gone! ^.^ */
    ...
} *p;
```

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

Case 2: **FAM never accessed through the composite struct**.

- Use struct_group_tagged()/__struct_group()

```
struct flex { /* AFTER */
    /* New members must be added within the struct_group() macro below. */
    struct_group_tagged(flex_hdr, hdr,
        int a; int b;
        size_t  count;
    );
    struct foo fam[] __counted_by(count);
};

struct composite { /* AFTER */
    ...
    struct flex_hdr middle; /* FAM is gone! ^.^ */
    ...
} *p;
```

```
{
    p->middle.a
    p->middle.b
    p->middle.count
}
        ==
{
    p->middle.hdr.a
    p->middle.hdr.b
    p->middle.hdr.count
}
```

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

Case 2: **FAM never accessed through the composite struct**.

- 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

Case 2: **FAM never accessed through the composite struct**.

- 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

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

struct composite_struct {
    ...

    struct flex_struct flex_in_the_middle;
    struct foo fixed_array[MAX_LENGTH];
    ...
} __packed;
```

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

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

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

struct composite_struct {
    ...

    struct flex_struct flex_in_the_middle;
    struct foo fixed_array[MAX_LENGTH];
    ...
} __packed;
```

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

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

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

struct composite_struct {
    ...

    struct flex_struct flex_in_the_middle;
    struct foo fixed_array[MAX_LENGTH];
    ...
} __packed;
```

- `flex_array` and `fixed_array` share the same address in memory - in the best scenario.
- Both form an implicit union.

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

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

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

struct composite_struct {
    ...

    struct flex_struct flex_in_the_middle;
    struct foo fixed_array[MAX_LENGTH];
    ...
} __packed;
```

- `flex_array` and `fixed_array` share the same address in memory - in the best scenario.
- Both form an implicit union.
- The composite_struct's alignment may change.

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

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

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

struct composite_struct {
    ...

    struct flex_struct flex_in_the_middle;
    struct foo fixed_array[MAX_LENGTH];
    ...
} __packed;
```

- `flex_array` and `fixed_array` share the same address in memory - in the best scenario.
- Both form an implicit union.
- The composite_struct's alignment may change.
- static_assert() is necessary.

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

Trailing padding

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

Trailing padding

pahole output:

```
struct virtio_net_rss_config_trailer {                    /*  offset size*/
        __le16                       max_tx_vq;           /*     0     2 */
        __u8                         hash_key_length;     /*     2     1 */
        __u8                         hash_key_data[];     /*     3     0 */
        /* size: 4, cachelines: 1, members: 3 */
        /* padding: 1 */
};

struct virtnet_info {
…                                                          /*  offset size*/
        struct virtio_net_rss_config_trailer rss_trailer; /*    80     4 */
        /* XXX last struct has 1 byte of padding */
        u8                           rss_hash_key_data[40]; /*    84    40 */
...
};
```

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

Trailing padding

pahole output:

```
struct virtio_net_rss_config_trailer {                    /*  offset size*/
        __le16                       max_tx_vq;           /*     0     2 */
        __u8                         hash_key_length;     /*     2     1 */
        __u8                         hash_key_data[];     /*     3     0 */
        /* size: 4, cachelines: 1, members: 3 */
        /* padding: 1 */
};

struct virtnet_info {
…                                                          /*  offset size*/
        struct virtio_net_rss_config_trailer rss_trailer; /*    80     4 */
        /* XXX last struct has 1 byte of padding */
        u8                           rss_hash_key_data[40]; /*    84    40 */
...
};
```

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

Trailing padding

pahole output:

```
struct virtio_net_rss_config_trailer {                    /*  offset size*/
        __le16                   max_tx_vq;               /*     0     2 */
        __u8                     hash_key_length;         /*     2     1 */
        __u8                     hash_key_data[];         /*     3     0 */
        /* size: 4, cachelines: 1, members: 3 */
        /* padding: 1 */
};

struct virtnet_info {
…                                                         /*  offset size*/
        struct virtio_net_rss_config_trailer rss_trailer; /*    80     4 */
        /* XXX last struct has 1 byte of padding */
        u8                       rss_hash_key_data[40]; /*    84    40 */
...
};
```

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

Trailing padding

pahole output:

```
struct virtio_net_rss_config_trailer {                    /*  offset size*/
        __le16                     max_tx_vq;             /*     0     2 */
        __u8                       hash_key_length;       /*     2     1 */
        __u8                       hash_key_data[];       /*     3     0 */
        /* size: 4, cachelines: 1, members: 3 */
        /* padding: 1 */
};

struct virtnet_info {
…                                                          /*  offset size*/
        struct virtio_net_rss_config_trailer rss_trailer; /*    80     4 */
        /* XXX last struct has 1 byte of padding */
        u8                         rss_hash_key_data[40]; /*    84    40 */
...
};
```

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

Trailing padding

pahole output:

```
struct virtio_net_rss_config_trailer {                    /*  offset size*/
        __le16                       max_tx_vq;           /*     0     2 */
        __u8                         hash_key_length;     /*     2     1 */
        __u8                         hash_key_data[];     /*     3     0 */
        /* size: 4, cachelines: 1, members: 3 */
        /* padding: 1 */
};

struct virtnet_info {
…                                                         /*  offset size*/
        struct virtio_net_rss_config_trailer rss_trailer; /*    80     4 */
        /* XXX last struct has 1 byte of padding */
        u8                           rss_hash_key_data[40]; /*    84    40 */
...
};
```

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

Trailing padding

pahole output:

```
struct virtio_net_rss_config_trailer {                    /*  offset size*/
        __le16                       max_tx_vq;           /*     0     2 */
        __u8                         hash_key_length;     /*     2     1 */
        __u8                         hash_key_data[];     /*     3     0 */
        /* size: 4, cachelines: 1, members: 3 */
        /* padding: 1 */
};

struct virtnet_info {
…                                                         /*  offset size*/
        struct virtio_net_rss_config_trailer rss_trailer; /*    80     4 */
        /* XXX last struct has 1 byte of padding */
        u8                           rss_hash_key_data[40]; /*    84    40 */
...
};
```

(flex-array member offset) 83

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

Trailing padding

pahole output:

```
struct virtio_net_rss_config_trailer {                    /*  offset size*/
        __le16                       max_tx_vq;           /*     0     2 */
        __u8                         hash_key_length;     /*     2     1 */
        __u8                         hash_key_data[];     /*     3     0 */
        /* size: 4, cachelines: 1, members: 3 */
        /* padding: 1 */
};

struct virtnet_info {
…                                                         /*  offset size*/
        struct virtio_net_rss_config_trailer rss_trailer; /*    80     4 */
        /* XXX last struct has 1 byte of padding */
        u8                           rss_hash_key_data[40]; /*    84    40 */
...
};
```

(flex-array member offset) 83

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

Trailing padding

pahole output:

```
struct virtio_net_rss_config_trailer {                    /*  offset size*/
        __le16                       max_tx_vq;           /*     0     2 */
        __u8                         hash_key_length;     /*     2     1 */
        __u8                         hash_key_data[];     /*     3     0 */
        /* size: 4, cachelines: 1, members: 3 */
        /* padding: 1 */
};

struct virtnet_info {
…                                                         /*  offset size*/
        struct virtio_net_rss_config_trailer rss_trailer; /*    80     4 */
        /* XXX last struct has 1 byte of padding */
        u8                           rss_hash_key_data[40]; /*    84    40 */
...
};
```

(flex-array member offset) 83 != 84 (fixed-size array offset)

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

Trailing padding - 4156c3745f06 ("virtio_net: Fix alignment...")

pahole output:

```
struct virtio_net_rss_config_trailer {                    /*  offset size*/
        __le16                     max_tx_vq;             /*     0     2 */
        __u8                       hash_key_length;       /*     2     1 */
        __u8                       hash_key_data[];       /*     3     0 */
        /* size: 4, cachelines: 1, members: 3 */
        /* padding: 1 */
};

struct virtnet_info {
...                                                       /*  offset size*/
        struct virtio_net_rss_config_trailer rss_trailer; /*    80     4 */
        /* XXX last struct has 1 byte of padding */
        u8                         rss_hash_key_data[40]; /*    84    40 */
...
};
```

(flex-array member offset) 83 != 84 (fixed-size array offset)

-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;
        ...
+   );
    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.

```
 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;
        ...
+   );
    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.

- However, **FAM digest** is accessed at run-time.

```
 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;
        ...
+   );
    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.

- However, **FAM digest** is accessed at run-time.

```
 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;
        ...
+   );
    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.

- However, **FAM digest** is 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;
```

```
struct ima_max_digest_data hash;
...
/* read data from the FAM digest */
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.

- However, **FAM digest** is 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;
```

```
struct ima_max_digest_data hash;
...
/* read data from the FAM digest */
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.

- However, **FAM digest** is 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;
```

```
 struct ima_max_digest_data hash;
 ...
 /* read data from the FAM digest */
 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.

- However, **FAM digest** is 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;
```

```
struct ima_max_digest_data hash;
...
/* read data from the FAM digest */
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.

-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.

```
    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.

```
    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.

```
    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.

```
 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;
        ...
+   );
    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;
```

```
    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);
```

-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;
        ...
+   );
    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;
```

```
    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);
```

-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;
        ...
+   );
    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;
```

```
    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);
```

-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;
        ...
+   );
    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;
```

```
    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);
```

-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;
        ...
+   );
    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;
```

```
    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);
```

-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;
        ...
+   );
    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;
```

```
    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);
```

-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;
        ...
+   );
    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;
```

```
    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);
```

-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;
        ...
+   );
    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;
```

```
    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);
```

-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 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 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

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


int some_function(...)
{
    struct {
        struct flex_struct flex; /* on-stack -Wfamnae warning */
        struct foo fixed_array[10];
    } obj = ...
    ...
}
```

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

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


int some_function(...)
{
    struct {
        struct flex_struct flex; /* on-stack -Wfamnae warning */
        struct foo fixed_array[10];
    } obj = ...
    ...
}
```

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
struct flex_struct {
    ...
    size_t  count;
    struct foo flex_array[]; /* flex-array member */
};


int some_function(...)
{
    struct {
        struct flex_struct flex; /* on-stack -Wfamnae warning */
        struct foo fixed_array[10]; /* fixed-size array */
    } obj = ...
    ...
}
```

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
struct fun_admin_bind_req {
    struct fun_admin_req_common common;
    struct fun_admin_bind_entry entry[];
};

int fun_bind(...)
{
    struct {
        struct fun_admin_bind_req req;/* on-stack -Wfamnae warning */
        struct fun_admin_bind_entry entry[2];
    } cmd = ...
    ...
}
```

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
struct fun_admin_bind_req {
    struct fun_admin_req_common common;
    struct fun_admin_bind_entry entry[];
};

int fun_bind(...)
{
    struct {
        struct fun_admin_bind_req req;/* on-stack -Wfamnae warning */
        struct fun_admin_bind_entry entry[2];
    } cmd = ...
    ...
}
```

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
struct fun_admin_bind_req {
    struct fun_admin_req_common common;
    struct fun_admin_bind_entry entry[];      /* flex-array member */
};

int fun_bind(...)
{
    struct {
        struct fun_admin_bind_req req;/* on-stack -Wfamnae warning */
        struct fun_admin_bind_entry entry[2]; /* fixed-size array */
    } cmd = ...
    ...
}
```

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-   struct {
-       struct fun_admin_bind_req req;
-       struct fun_admin_bind_entry entry[2];
-   } cmd = {
-       .req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-                                                sizeof(cmd)),
-       .entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-       .entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-   };
+   DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+   cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+                                            __struct_size(cmd));
+   cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+   cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-   struct {
-       struct fun_admin_bind_req req;
-       struct fun_admin_bind_entry entry[2];
-   } cmd = {
-       .req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-                                                sizeof(cmd)),
-       .entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-       .entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-   };
+   DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+   cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+                                            __struct_size(cmd));
+   cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+   cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-   struct {
-       struct fun_admin_bind_req req;
-       struct fun_admin_bind_entry entry[2];
-   } cmd = {
-       .req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-                                                sizeof(cmd)),
-       .entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-       .entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-   };
+   DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+   cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+                                            __struct_size(cmd));
+   cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+   cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-   struct {
-       struct fun_admin_bind_req req;
-       struct fun_admin_bind_entry entry[2];
-   } cmd = {
-       .req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-                                                sizeof(cmd)),
-       .entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-       .entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-   };
+   DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+   cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+                                            __struct_size(cmd));
+   cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+   cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-   struct {
-       struct fun_admin_bind_req req;
-       struct fun_admin_bind_entry entry[2];
-   } cmd = {
-       .req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-                                                sizeof(cmd)),
-       .entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-       .entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-   };
+   DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+   cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+                                            __struct_size(cmd));
+   cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+   cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-   struct {
-       struct fun_admin_bind_req req;
-       struct fun_admin_bind_entry entry[2];
-   } cmd = {
-       .req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-                                                sizeof(cmd)),
-       .entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-       .entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-   };
+   DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+   cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+                                            __struct_size(cmd));
+   cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+   cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-   struct {
-       struct fun_admin_bind_req req;
-       struct fun_admin_bind_entry entry[2];
-   } cmd = {
-       .req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-                                                sizeof(cmd)),
-       .entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-       .entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-   };
+   DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+   cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+                                            __struct_size(cmd));
+   cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+   cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-   struct {
-       struct fun_admin_bind_req req;
-       struct fun_admin_bind_entry entry[2];
-   } cmd = {
-       .req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-                                                sizeof(cmd)),
-       .entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-       .entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-   };
+   DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+   cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+                                            __struct_size(cmd));
+   cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+   cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

```
-   struct {
-       struct fun_admin_bind_req req;
-       struct fun_admin_bind_entry entry[2];
-   } cmd = {
-       .req.common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
-                                                sizeof(cmd)),
-       .entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0),
-       .entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1),
-   };
+   DEFINE_RAW_FLEX(struct fun_admin_bind_req, cmd, entry, 2);
+
+   cmd->common = FUN_ADMIN_REQ_COMMON_INIT2(FUN_ADMIN_OP_BIND,
+                                            __struct_size(cmd));
+   cmd->entry[0] = FUN_ADMIN_BIND_ENTRY_INIT(type0, id0);
+   cmd->entry[1] = FUN_ADMIN_BIND_ENTRY_INIT(type1, id1);
```

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

Case 4a: **Implicit unions** between FAMs and fixed-size arrays of the same element type – **on stack**.

- 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

Case 4b: **Implicit unions** between FAMs and objects of any type – **on stack**.

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

Case 4b: **Implicit unions** between FAMs and objects of any type – **on stack**.

```
struct cros_ec_command {
        ...
        uint8_t data[];
};

static size_t cros_ec_pdinfo_read(...)
{
...
    struct {
        struct cros_ec_command msg;
        union {
            struct ec_response_usb_pd_control_v1 resp;
            struct ec_params_usb_pd_control params;
        };
    } __packed ec_buf;
...
}
```

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

Case 4b: **Implicit unions** between FAMs and objects of any type – **on stack**.

```
struct cros_ec_command {
        ...
        uint8_t data[];
};

static size_t cros_ec_pdinfo_read(...)
{
...
    struct {
        struct cros_ec_command msg;
        union {
            struct ec_response_usb_pd_control_v1 resp;
            struct ec_params_usb_pd_control params;
        };
    } __packed ec_buf;
...
}
```

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

Case 4b: **Implicit unions** between FAMs and objects of any type – **on stack**.

```
struct cros_ec_command {
        ...
        uint8_t data[];
};

static size_t cros_ec_pdinfo_read(...)
{
...
    struct {
        struct cros_ec_command msg;
        union {
            struct ec_response_usb_pd_control_v1 resp;
            struct ec_params_usb_pd_control params;
        };
    } __packed ec_buf;
...
}
```

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

Case 4b: **Implicit unions** between FAMs and objects of any type – **on stack**.

```
-   struct {
-       struct cros_ec_command msg;
-       union {
-           struct ec_response_usb_pd_control_v1 resp;
-           struct ec_params_usb_pd_control params;
-       };
-   } __packed ec_buf;
+   DEFINE_RAW_FLEX(struct cros_ec_command, msg, data,
+                   MAX(sizeof(struct ec_response_usb_pd_control_v1),
+                       sizeof(struct ec_params_usb_pd_control)));
```

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

Case 4b: **Implicit unions** between FAMs and objects of any type – **on stack**.

```
-   struct {
-       struct cros_ec_command msg;
-       union {
-           struct ec_response_usb_pd_control_v1 resp;
-           struct ec_params_usb_pd_control params;
-       };
-   } __packed ec_buf;
+   DEFINE_RAW_FLEX(struct cros_ec_command, msg, data,
+                   MAX(sizeof(struct ec_response_usb_pd_control_v1),
+                       sizeof(struct ec_params_usb_pd_control));
```

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

Case 4b: **Implicit unions** between FAMs and objects of any type – **on stack**.

- da90147bf52b (“platform/chrome: cros_ec_debugfs:…”)

```
-   struct {
-       struct cros_ec_command msg;
-       union {
-           struct ec_response_usb_pd_control_v1 resp;
-           struct ec_params_usb_pd_control params;
-       };
-   } __packed ec_buf;
+   DEFINE_RAW_FLEX(struct cros_ec_command, msg, data,
+                   MAX(sizeof(struct ec_response_usb_pd_control_v1),
+                       sizeof(struct ec_params_usb_pd_control)));
```

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

Case 4b: **Implicit unions** between FAMs and objects of any type – **on stack**.

```
static int cros_ec_sleep_event(...)
{
...
    struct {
        struct cros_ec_command msg;
        union {
            struct ec_params_host_sleep_event req0;
            struct ec_params_host_sleep_event_v1 req1;
            struct ec_response_host_sleep_event_v1 resp1;
        } u;
    } __packed buf;
...
}
```

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

Case 4b: **Implicit unions** between FAMs and objects of any type – **on stack**.

```
+union cros_ec_sleep_data {
+       struct ec_params_host_sleep_event req0;
+       struct ec_params_host_sleep_event_v1 req1;
+       struct ec_response_host_sleep_event_v1 resp1;
+} __packed;
...
-   struct {
-       struct cros_ec_command msg;
-       union {
-           struct ec_params_host_sleep_event req0;
-           struct ec_params_host_sleep_event_v1 req1;
-           struct ec_response_host_sleep_event_v1 resp1;
-       } u;
-   } __packed ec_buf;
+   DEFINE_RAW_FLEX(struct cros_ec_command, msg, data,
+                   sizeof(union cros_ec_sleep_data));
```

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

Case 4b: **Implicit unions** between FAMs and objects of any type – **on stack**.

- Re: [PATCH][next] platform/chrome: cros_ec: Avoid...

```
“> +    DEFINE_RAW_FLEX(struct cros_ec_command, msg, data,
 > +                    sizeof(union cros_ec_sleep_data));

Is it possible to use something similar to:

MAX(MAX(sizeof(A), sizeof(B)),
    sizeof(C))

so that the union doesn't need to be defined?”
```

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

Case 4b: **Implicit unions** between FAMs and objects of any type – **on stack**.

- (ab)using DEFINE_RAW_FLEX() (blogpost in progress...)

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

Case 4b: **Implicit unions** between FAMs and objects of any type – **on stack**.

- (ab)using DEFINE_RAW_FLEX() (blogpost in progress...)
- DEFINE_RAW_FLEX() was not created for this scenario

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

Case 4b: **Implicit unions** between FAMs and objects of any type – **on stack**.

- (ab)using DEFINE_RAW_FLEX() (blogpost in progress...)
- DEFINE_RAW_FLEX() was not created for this scenario
- We probably need a new helper… 🤔

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

TRAILING_OVERLAP() - A new helper

```
#define TRAILING_OVERLAP(TYPE, NAME, FAM, MEMBERS)                      \
   union {                                                              \
      TYPE NAME;                                                        \
      struct {                                                          \
         unsigned char __offset_to_FAM[offsetof(TYPE, FAM)];            \
         MEMBERS;                                                       \
      };                                                                \
   }
```

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

TRAILING_OVERLAP() - A new helper

- Creates a union between a FAM and a set of MEMBERS that would otherwise follow it.

```
#define TRAILING_OVERLAP(TYPE, NAME, FAM, MEMBERS)                \
   union {                                                        \
      TYPE NAME;                                                  \
      struct {                                                    \
         unsigned char __offset_to_FAM[offsetof(TYPE, FAM)];      \
         MEMBERS;                                                 \
      };                                                          \
   }
```

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

TRAILING_OVERLAP() - A new helper

- Creates a union between a FAM and a set of MEMBERS that would otherwise follow it.
- Overlays the trailing MEMBERS onto the FAM while preserving the original memory layout.

```
#define TRAILING_OVERLAP(TYPE, NAME, FAM, MEMBERS)                    \
    union {                                                           \
        TYPE NAME;                                                    \
        struct {                                                      \
            unsigned char __offset_to_FAM[offsetof(TYPE, FAM)];       \
            MEMBERS;                                                  \
        };                                                            \
    }
```

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

TRAILING_OVERLAP() - A new helper

```
struct {
    struct cros_ec_command msg;
    union {
        struct ec_params_host_sleep_event req0;
        struct ec_params_host_sleep_event_v1 req1;
        struct ec_response_host_sleep_event_v1 resp1;
    } u;
} __packed buf;
```

```
struct cros_ec_command {
        ...
        uint8_t data[];
};
```

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

TRAILING_OVERLAP() - A new helper

```
struct {
    struct cros_ec_command msg;
    union {
        struct ec_params_host_sleep_event req0;
        struct ec_params_host_sleep_event_v1 req1;
        struct ec_response_host_sleep_event_v1 resp1;
    } u;
} __packed buf;
```

```
struct cros_ec_command {
        ...
        uint8_t data[];
};
```

```
union {                                 /* Open-coded TRAILING_OVERLAP() */
    struct cros_ec_command msg;
    struct {
        unsigned char __offset_to_data[offsetof(struct cros_ec_command, data)];
        union {
            struct ec_params_host_sleep_event req0;
            struct ec_params_host_sleep_event_v1 req1;
            struct ec_response_host_sleep_event_v1 resp1;
        } u;
    }
} __packed buf;
```

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

TRAILING_OVERLAP() - A new helper

```
struct {
     struct cros_ec_command msg;
     union {
          struct ec_params_host_sleep_event req0;
          struct ec_params_host_sleep_event_v1 req1;
          struct ec_response_host_sleep_event_v1 resp1;
     } u;
} __packed buf;
```

```
struct cros_ec_command {
          ...
          uint8_t data[];
};
```

```
union {                                   /* Open-coded TRAILING_OVERLAP() */
     struct cros_ec_command msg;
     struct {
          unsigned char __offset_to_data[offsetof(struct cros_ec_command, data)];
          union {
               struct ec_params_host_sleep_event req0;
               struct ec_params_host_sleep_event_v1 req1;
               struct ec_response_host_sleep_event_v1 resp1;
          } u;
     }
} __packed buf;
```

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

TRAILING_OVERLAP() - A new helper

```
struct {
     struct cros_ec_command msg;
     union {
          struct ec_params_host_sleep_event req0;
          struct ec_params_host_sleep_event_v1 req1;
          struct ec_response_host_sleep_event_v1 resp1;
     } u;
} __packed buf;
```

```
struct cros_ec_command {
        ...
        uint8_t data[];
};
```

```
union {                                  /* Open-coded TRAILING_OVERLAP() */
     struct cros_ec_command msg;
     struct {
          unsigned char __offset_to_data[offsetof(struct cros_ec_command, data)];
          union {
               struct ec_params_host_sleep_event req0;
               struct ec_params_host_sleep_event_v1 req1;
               struct ec_response_host_sleep_event_v1 resp1;
          } u;
     }
} __packed buf;
```

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

TRAILING_OVERLAP() - A new helper

```
struct {
    struct cros_ec_command msg;
    union {
        struct ec_params_host_sleep_event req0;
        struct ec_params_host_sleep_event_v1 req1;
        struct ec_response_host_sleep_event_v1 resp1;
    } u;
} __packed buf;
```

```
struct cros_ec_command {
        ...
        uint8_t data[];
};
```

```
TRAILING_OVERLAP(struct cros_ec_command, msg, data,
    union {
        struct ec_params_host_sleep_event req0;
        struct ec_params_host_sleep_event_v1 req1;
        struct ec_response_host_sleep_event_v1 resp1;
    } u;
) __packed buf;
```

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

TRAILING_OVERLAP() - A new helper

```
+union cros_ec_sleep_data {
+       struct ec_params_host_sleep_event req0;
+       struct ec_params_host_sleep_event_v1 req1;
+       struct ec_response_host_sleep_event_v1 resp1;
+} __packed;
...
-   struct {
-       struct cros_ec_command msg;
-       union {
-           struct ec_params_host_sleep_event req0;
-           struct ec_params_host_sleep_event_v1 req1;
-           struct ec_response_host_sleep_event_v1 resp1;
-       } u;
-   } __packed ec_buf;
+   DEFINE_RAW_FLEX(struct cros_ec_command, msg, data,
+                   sizeof(union cros_ec_sleep_data));
```

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

TRAILING_OVERLAP() - A new helper

```
-   struct {
-       struct cros_ec_command msg;
+   TRAILING_OVERLAP(struct cros_ec_command, msg, data,
        union {
            struct ec_params_host_sleep_event req0;
            struct ec_params_host_sleep_event_v1 req1;
            struct ec_response_host_sleep_event_v1 resp1;
        } u;
-   } __packed ec_buf;
+   ) __packed ec_buf;
```

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

TRAILING_OVERLAP() - A new helper

- [PATCH v2] platform/chrome: cros_ec: Avoid...

```
-   struct {
-       struct cros_ec_command msg;
+   TRAILING_OVERLAP(struct cros_ec_command, msg, data,
        union {
            struct ec_params_host_sleep_event req0;
            struct ec_params_host_sleep_event_v1 req1;
            struct ec_response_host_sleep_event_v1 resp1;
        } u;
-   } __packed ec_buf;
+   ) __packed ec_buf;
```

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

- TRAILING_OVERLAP() - A new helper
  - Superior to __struct_group()/container_of() in many cases

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

TRAILING_OVERLAP() - A new helper

- Superior to __struct_group()/container_of() in many cases

```
 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;
        ...
+   );
    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;
```

```
    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);
```

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

TRAILING_OVERLAP() - A new helper

- Superior to __struct_group()/container_of() in many cases

```
 struct ima_max_digest_data {
-  struct ima_digest_data hdr;
-  u8 digest[HASH_MAX_DIGESTSIZE];
+  TRAILING_OVERLAP(struct ima_digest_data, hdr, digest,
+     u8 digest[HASH_MAX_DIGESTSIZE];
+  );
 } __packed;
```

The evolution of -Wfamnae approaches

The evolution of -Wfamnae approaches

- Separate struct flex_hdr { … }; and struct flex { … };
- __struct_group()/container_of()
- DEFINE_FLEX()/DEFINE_RAW_FLEX()
- (ab)using DEFINE_RAW_FLEX()
- The new TRAILING_OVERLAP() helper.

The evolution of -Wfamnae approaches

[PATCH v5][for-next/hardening] acpi: nfit: Avoid multiple...

- Changes in v5:
  - Fix union initialization.
  - Leave trailing object indentation unchanged.
- Changes in v4:
  - Use the new TRAILING_OVERLAP() helper.
- Changes in v3:
  - Use union instead of DEFINE_RAW_FLEX().
- Changes in v2:
  - Use DEFINE_RAW_FLEX() instead of __struct_group().
- V1: Use __struct_group()

Conclusions

Conclusions

A three-step approach for the complex case:

Conclusions

A three-step approach for the complex case:

- Use **struct_group_tagged()** to create a new tagged struct.
  - This groups together all members in the flex struct **except the FAM**.
- **Change the type** of the conflicting object to the newly created tagged struct.
- Use **container_of()** to retrieve a pointer to the flex struct when needed.
  - Access the **FAM** via this pointer if necessary.

Conclusions

A three-step approach for the complex case:

- Use **struct_group_tagged()** to create a new tagged struct.
  - This groups together all members in the flex struct **except the FAM**.
- **Change the type** of the conflicting object to the newly created tagged struct.
- Use **container_of()** to retrieve a pointer to the flex struct when needed.
  - Access the **FAM** via this pointer if necessary.

- However, in some cases we can use **TRAILING_OVERLAP(),** instead.

Conclusions

For implicit unions on the stack (FAM & objects of the same element type):

- Use **DECLARE_FLEX()** when the FAM is annotated with __counted_by().
- We can use **DECLARE_RAW_FLEX()** in any other case.

Conclusions

For implicit unions anywhere (FAM & a group of objects of any type):

- Use **TRAILING_OVERLAP()** when FAM and the group of objects can be moved to the end.

Conclusions

- Clear strategy to enable -Wflex-array-member-not-at-end in mainline, soon.
- Dozens of patches are already in mainline.

Conclusions

- Clear strategy to enable -Wflex-array-member-not-at-end in mainline, soon.
- Dozens of patches are already in mainline.
- Down to **~110** (from ~650) unique warnings in linux-next.
- **~83%** of unique warnings addressed so far.

Thank you, Canberra! 🇦🇺

Gustavo A. R. Silva
gustavoars@kernel.org
fosstodon.org/@gustavoars
https://embeddedor.com/blog/

By @shidokou

More on upstream Linux kernel hardening

- **Safer flexible arrays for the kernel (LWN article)**
- **How to use the new counted_by attribute in C (and Linux) (Blogpost)**
- **Enhancing spatial safety: Better array-bounds checking in C (and Linux) (Presentation)**
- **Linux Kernel Hardening: Ten Years Deep (YouTube video)**
- **GCC features to help harden the kernel (LWN article)**
- **https://best.openssf.org/Compiler-Hardening-Guides/Compiler-Options-Hardening-Guide-for-C-and-C++.html**