Variable Length Array's in Structs (VLAIS) and why they don't belong in your C code

Mark Charlebois
Qualcomm Innovation Center (QuIC)

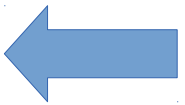
Linux Plumbers Conference 2013

VLAIS – What is it?

```
void foo( int n ) {
    struct {
        int x;
        char y[n];
        int z;
    } bar;
}
```

VLAIS – What is it?

```
void foo( int n ) {
    struct {
        int x;
        char y[n];
        int z;
    } bar;
}
```



This is expressly forbidden in C99 and C11

Issues Caused by VLAIS

- Kernel is not C compliant
- Tools break when analyzing the kernel code
- VLAIS is not a feature of C99 or C11 so there is no roadmap for support in the standard
- Feature is undocumented
- Since there is no warning about this unless you use -std99 and -pedantic, developers do not realize this is illegal C code - and - those flags won't work with the kernel
- Major blocker for other compilers (like clang) to build the kernel

How and When Did This Happen?

- Discussions in 2004 about the previous issues
- GCC devs felt that since VLAIS support was added for ADA, it would be easy to also support in C
  - http://gcc.gnu.org/ml/gcc/2004-06/msg01322.html
- Developers may not be aware VLAIS is explicitly disallowed in the C standards

What are the Alternatives

- Do not place the VLA in a struct
- If possible use a flexible array member and move the array to the end of the struct
- Allocate an equivalent block of memory and calculate the pointer offsets

Offset Calculation via Macros

- Macros make the code more readable
- Recent compilers can show errors in nested macros so debugging is still easy
- Macros themselves can still be ugly

Calculating Index Offset

```
#define vla_struct(structname) size_t structname##__##next = 0
#define vla_struct_size(structname) structname##__##next

#define vla_item(structname, type, name, n) \
	type * structname##_##name; \
	size_t structname##_##name##__##pad = \
		((-structname##__##next) & (__alignof__(type)-1)); \
\
	size_t structname##_##name##__##offset = \
		structname##__##next + structname##_##name##__##pad; \
\
	size_t structname##_##name##__##sz = n * sizeof(type); \
\
	structname##__##next = \
		structname##__##next + structname##_##name##__##pad + \
		structname##_##name##__##sz;

#define vla_ptr(ptr,structname,name) structname##_##name = \
	(__typeof__(structname##_##name))&ptr[structname##_##name##__##offset]
```

Sample Code Using Macros

```
..

void test(size_t a, size_t b) {
      vla_struct(foo);
            vla_item(foo, struct bar, vara, a);
            vla_item(foo, int, varb, b);

      char *buffer = malloc(vla_struct_size(foo));

      vla_ptr(buffer, foo, vara);
      vla_ptr(buffer, foo, varb);

      for (int x=0; x<b; x+=1)
            foo_varb[x] = x;
      ...
}
```

Kernel Maintainer Feedback

- Macros are evil
- Each file needs to be patched individually, based on needs, not based on generic macros
- We can just remove VLAIS usage (USB Gadget)
- The kernel was only ever meant to support GCC so go away
- Go ask LLVM to support VLAIS

Proposed Patches

- dm-crypt.c, hmac.c, libcrc32.c, testmgr.c
  - convert to array on the stack
  - add pointer offsets
- netfilter/xt_repldata.c
  - Remove VLAIS members from struct
  - Overalloc the struct
  - Calculate pointer offsets
- gadget.c
  - rewrite without VLAIS

Suggestions?

- Comments on the patches?
- Required testing for crypto and netfilter patches?
- Is the kernel community interested in being more closely aligned with C99, C11?
- Several kernel maintainers and developers have expressed interest in compiling the kernel with clang