**Comments from the WG14 committee on WG23 part 3**

I introduced the latest version of the document, pointing out in particular the Language Concepts section (why C is like it is – the vulnerabilities are not errors in the standard – but features to be aware of), and the Section 5 ‘top 10’ issues.

There was a lot of discussion on section5:

* [line 1] The need for the cast after malloc was not liked. Would be better to wrap the allocation in a macro, so the target type and the cast type are guaranteed to be compatible
* [line 2] Martin Sebor has written a paper on the problems with Annex K functions (thought there was some support that the guidance to use Annex K was appropriate – provided there is advice on how to use it correctly)
* [line 4] has ‘Remove’ in it (we should have decided what to do with this!)

Overall, there was some support for a top 10 – for programmers not in the safety/security domain – but with the recognition that they are unlikely to be reading this document. The counter argument was that anyone in the safety/security domain ought to be willing to read the whole document and not need a short summary.

The suggestion was either that:

* the table is converted into a ‘top 10 problems’, rather than a ’top 10 solutions’
* Remove it completely (this was the general consent)

Some specific issues identified were:

**6.5.2** The final recommendation to use the volatile qualification on an enumeration type switch selector wasn’t liked as it interferes with static analysis, and that it shouldn’t be necessary, as a compliant compiler shouldn’t optimise the default clause away.

**6.8.2** It was pointed out that the advice to use strncpy is not without issues – as it may remove the string terminator (as explained in 6.8.1). There was general concern that advice like ‘use X’ should include how to use X correctly.

**6.11** The focus on casting the return from malloc was not liked, as it was argued that this is a false security. As said earlier the preferred approach ought to be to wrap the allocation and type in a macro #define allocate(T, name, N) T \*name = (T\*)malloc(N)

It was also suggested that this isn’t the main issue with allocation – but rather getting the size wrong – again may be fixed with a macro

 #define allocateArray(T, name, M) T \*name = (T\*)malloc(sizeof(T) \* M)

Observation: we don’t say anything about Variable Length Arrays.

I left a general invitation for anyone with comments to mail me

 Clive Pygott 1/11/2017