CHP comments on ISO-IECJTC1-SC22-WG23_N1413-24772-3-C-vulnerabilities-prep-for-with-editing-convenor-20240909.docx

Technical comments:

p15[1]: Table 1 – Top avoidance mechanisms in C

I have problems with the first two entries:

- 1) The use of macros when allocating memory. I've never seen problems with allocated memory – possibly because none of our customers use dynamic memory. I can see this could be useful advise, but not the most important.
- 2) The use of Annex K, is in theory good advice, but WG14 is very sceptical about the effectiveness of Annex K – both in terms of the way its defined and the quality of its implementations. Every few meetings there is a move to drop it

SM – Understand, but this was wording from the C specialists. Changing it requires a bigger committee.

p35 6.20.1 1st para: "… can result in the variable operating on an entity other…"

Variables don't operate on anything!

Suggest "… can result in the variable found not being the one expected"

SM – Thx.

P62 6.65.1 the code examples:

There is no #define example with the first bullet, so the example "my_age = my_age + 1;" is confusing, as the only 'my_age' is in the int example for bullet 2.

You could say:

```
#define your_age 42
your_age = your_age + 1;
```

and point out that it doesn't compile, but:

```
#define your_age 42
    printf("%d\n", your_age);

#define your_age 21
    printf("%d\n", your_age);
```

does compile (with a redefinition warning), and prints "42" and "21"

For the example using int declarations:

```
int const my_age = 42;
int *variable_age = &my_age;
*variable_age = 75;  //will also set my_age to 75
```

This does not compile (Visual Studio 2010) 'loss of const qualification' when address of my_age taken. If variable_age is made a const int *, then the assignment fails.

[1] p13 etc. are page numbers in the marked up Word/PDF document

For the second int example:

```
int const my_age = 42;
const int * const some_age( &my_age  );
int *variable_age = some_age;
*variable_age = 75; // sets my_age to 75.
```

This also fails to compile, for the same reason

I'm not sure is the C standard requires this behaviour, and if so, for how long. If it's a guaranteed compiler error, these discussions need to be deleted, else a caveat needs adding, like '... may unexpectedly be compiled"

SM – This is where I need help. I cribbed this material from C++ and tried to leave out the C++ specific pieces. This is brand new material. I don't want to touch it without help. If we could spend ½ hour on Zoom we could likely fix it.

Layout/Typos

p15: Table 1 – Top avoidance mechanisms in C

This section is introduced as: "5. Top avoidance mechanisms", but has been preceded by "5.General language concepts and primary avoidance mechanisms" and "5.1 General C language concepts". Should it be 5.2?

Good catch. Indeed. Thank you. Made it 5.2.

p19 para 3&4 (and multiple other places): "... can or might not.." seems an odd phrase. '...can or cannot...' or '...might or might not...' seem more natural

SM – Good catch. Changed to "It is not certain that the loop terminates ..."

p21 6.6.1 1st para: "...2024 6.46is applicable to C..." missing ' ' before 'is'

This seems to be a recurring issue, modifications highlighted in the markup introduce errors in the document when accepted. I've noted a number – but no guarantee that these are the only ones.

SM - Thanks.

p36 6.21.1 1st para: "...ISO/IEC 24772-1:2024 6.21ndoes not apply..." redundant 'n' before 'does'

SM - Thx

p37 6.24.1 2nd line of code: 'i' has been corrected to 'I' – invalidating the code

also para 3 last bullet: '...clause 6.7.9, "Initializatio"").' Missing 'n' and extra '"'

SM - thx

p59 6.60.1 line: "...C does not implement a such mechanisms.." The "a" is redundant.

SM - thx

p62 headline: "6.645 Modifying constants [UJO]" should be 6.65 M..."

Question:

p13: 4 para 2:
"*Organizations following this document meet the expectations of 4.2 of ISO/IEC 24772-1…*"
Does following this document meet **all** the requirements of 24772-1 4.2?

SM – We believe yes. The list is a direct copy of the list from 24772-1.

p13: 4 para after bulleted list:
Why no mentions of MISRA C? Its in the bibliography as [11] and isn't referenced anywhere else.

SM – Great catch. Thank you! I put a reference to MISRA in clause 4 before MITRE or CWE