

MSC39-C. Do not call `va_arg()` on a `va_list` that has an indeterminate value

Variadic functions access their variable arguments by using `va_start()` to initialize an object of type `va_list`, iteratively invoking the `va_arg()` macro, and finally calling `va_end()`. The `va_list` may be passed as an argument to another function, but calling `va_arg()` within that function causes the `va_list` to have an [indeterminate value](#) in the calling function. As a result, attempting to read variable arguments without reinitializing the `va_list` can have [unexpected behavior](#). According to the C Standard, 7.16, paragraph 3 [ISO/IEC 9899:2011],

If access to the varying arguments is desired, the called function shall declare an object (generally referred to as `ap` in this subclause) having type `va_list`. The object `ap` may be passed as an argument to another function; if that function invokes the `va_arg` macro with parameter `ap`, the value of `ap` in the calling function is indeterminate and shall be passed to the `va_end` macro prior to any further reference to `ap`.²⁵³
253) It is permitted to create a pointer to a `va_list` and pass that pointer to another function, in which case the original function may take further use of the original list after the other function returns.

Noncompliant Code Example

This noncompliant code example attempts to check that none of its variable arguments are zero by passing a `va_list` to helper function `contains_zero()`. After the call to `contains_zero()`, the value of `ap` is [indeterminate](#).

```
#include <stdarg.h>
#include <stdio.h>

int contains_zero(size_t count, va_list ap) {
    for (size_t i = 1; i < count; ++i) {
        if (va_arg(ap, double) == 0.0) {
            return 1;
        }
    }
    return 0;
}

int print_reciprocals(size_t count, ...) {
    va_list ap;
    va_start(ap, count);

    if (contains_zero(count, ap)) {
        va_end(ap);
        return 1;
    }

    for (size_t i = 0; i < count; ++i) {
        printf("%f ", 1.0 / va_arg(ap, double));
    }

    va_end(ap);
    return 0;
}
```

Compliant Solution

The compliant solution modifies `contains_zero()` to take a pointer to a `va_list`. It then uses the `va_copy` macro to make a copy of the list, traverses the copy, and cleans it up. Consequently, the `print_reciprocals()` function is free to traverse the original `va_list`.

```

#include <stdarg.h>
#include <stdio.h>

int contains_zero(size_t count, va_list *ap) {
    va_list ap1;
    va_copy(ap1, *ap);
    for (size_t i = 1; i < count; ++i) {
        if (va_arg(ap1, double) == 0.0) {
            return 1;
        }
    }
    va_end(ap1);
    return 0;
}

int print_reciprocals(size_t count, ...) {
    int status;
    va_list ap;
    va_start(ap, count);

    if (contains_zero(count, &ap)) {
        printf("0 in arguments!\n");
        status = 1;
    } else {
        for (size_t i = 0; i < count; i++) {
            printf("%f ", 1.0 / va_arg(ap, double));
        }
        printf("\n");
        status = 0;
    }

    va_end(ap);
    return status;
}

```

Risk Assessment

Reading variable arguments using a `va_list` that has an [indeterminate value](#) can have unexpected results.

| Rule | Severity | Likelihood | Remediation Cost | Priority | Level |
|---------|----------|------------|------------------|-----------|-----------|
| MSC39-C | Low | Unlikely | Low | P3 | L3 |

Automated Detection

| Tool | Version | Checker | Description |
|--------------------------------------|---------|-----------------------------|---|
| Parasoft C/C++test | 10.4.2 | CERT_C-MSC39-a | Use macros for variable arguments correctly |
| Polyspace Bug Finder | R2019b | CERT C: Rule MSC39-C | Checks for: <ul style="list-style-type: none"> Invalid <code>va_list</code> argument Too many <code>va_arg</code> calls for current argument list Rule fully covered. |
| TrustInSoft Analyzer | 1.38 | variadic | Exhaustively verified. |

Related Vulnerabilities

Search for [vulnerabilities](#) resulting from the violation of this rule on the [CERT website](#).

Bibliography

| | |
|-------------------------------------|---|
| [ISO/IEC 9899:2011] | Subclause 7.16, "Variable Arguments <code><stdarg.h></code> " |
|-------------------------------------|---|

