Call to alloca in a loop

Name: Call to alloca in a loop

Description: Using alloca in a loop can lead to a stack overflow

ID: cpp/alloca-in-loop

Kind: problem

Severity: warning

Precision: high

The `alloca` macro allocates memory by expanding the current stack frame. Invoking `alloca` within a loop may lead to a stack overflow because the memory is not released until the function returns.

Recommendation

Consider invoking `alloca` once outside the loop, or using `malloc` or `new` to allocate memory on the heap if the allocation must be done inside the loop.

Example

The variable `path` is allocated inside a loop with `alloca`. Consequently, storage for all copies of the path is present in the stack frame until the end of the function.

```
1 char *dir_path;
2 char **dir_entries;
3 int count;
4
5 for (int i = 0; i < count; i++) {
6   char *path = (char*)alloca(strlen(dir_path) + strlen(dir_entry[i]) + 2);
7   // use path
8 }
```

In the revised example, `path` is allocated with `malloc` and freed at the end of the loop.

```
1 char *dir_path;
2 char **dir_entries;
3 int count;
4
5 for (int i = 0; i < count; i++) {
6   char *path = (char*)malloc(strlen(dir_path) + strlen(dir_entry[i]) + 2);
7   // use path
8   free(path);
9 }
```

References

- Linux Programmer's Manual: ALLOCA(3).
- Common Weakness Enumeration: CWE-770.