

# Use Case 1 Code Analysis with C or Cpp

## Code Analysis with C/C++

⚠ Please note that these instructions may be outdated!

The static analysis of C/C++ is a bit different from the analysis of other programming languages because the preprocessor complicates the analysis process a little bit. Resolving header files and macros, used in the preprocessing phase, is essential for a complete and correct C/C++ static code analysis. Let's break down step by step how to analyze a C application with [Kiuwan Code Security](#).

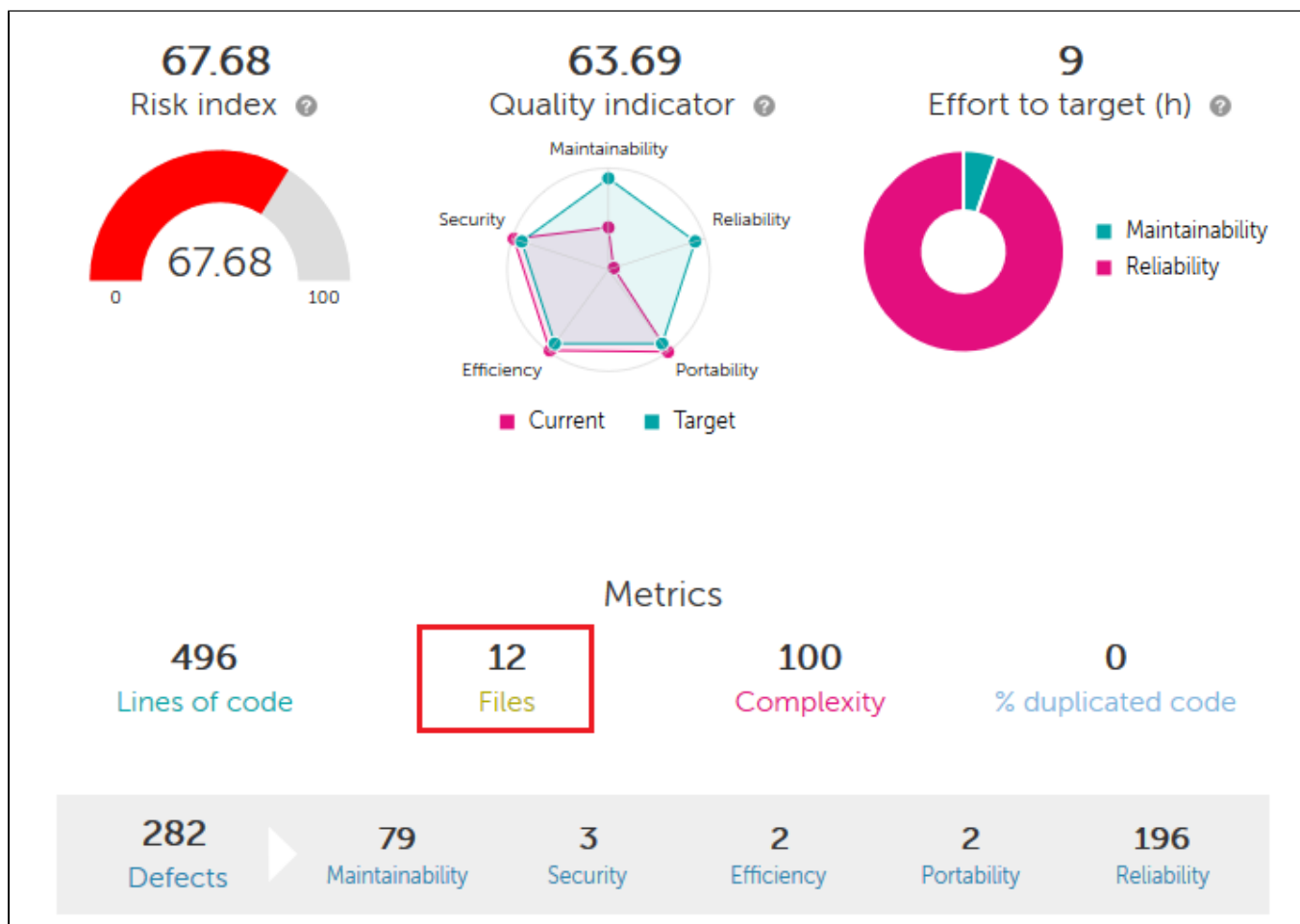
(Please note: some screenshots may be outdated)

### Analyzing in the cloud

We use the Linux FTP server (linux-ftp-0.17) as our sample application. The source code was downloaded from <https://launchpad.net/ubuntu/utopic/+source/linux-ftp>.

Analyzing the code with Kiuwan Code Security is easy:

1. Create an application.
2. Upload the source code:
3. Start the analysis,
4. View the results:



We found **15 files** to analyze in the uploaded ZIP file, but **only 12 were analyzed**. Why this difference? Let's look at the logs window (open the pop-up menu to the right of 'new analysis' button) to see the cause:

Unparsed files (3)	
linux-ftp-0.17/ftpd/ftpd.c	
linux-ftp-0.17/ftpd/extern.h	Parse error at line 240, column 1. Encountered: static
linux-ftp-0.17/support/vis.h	

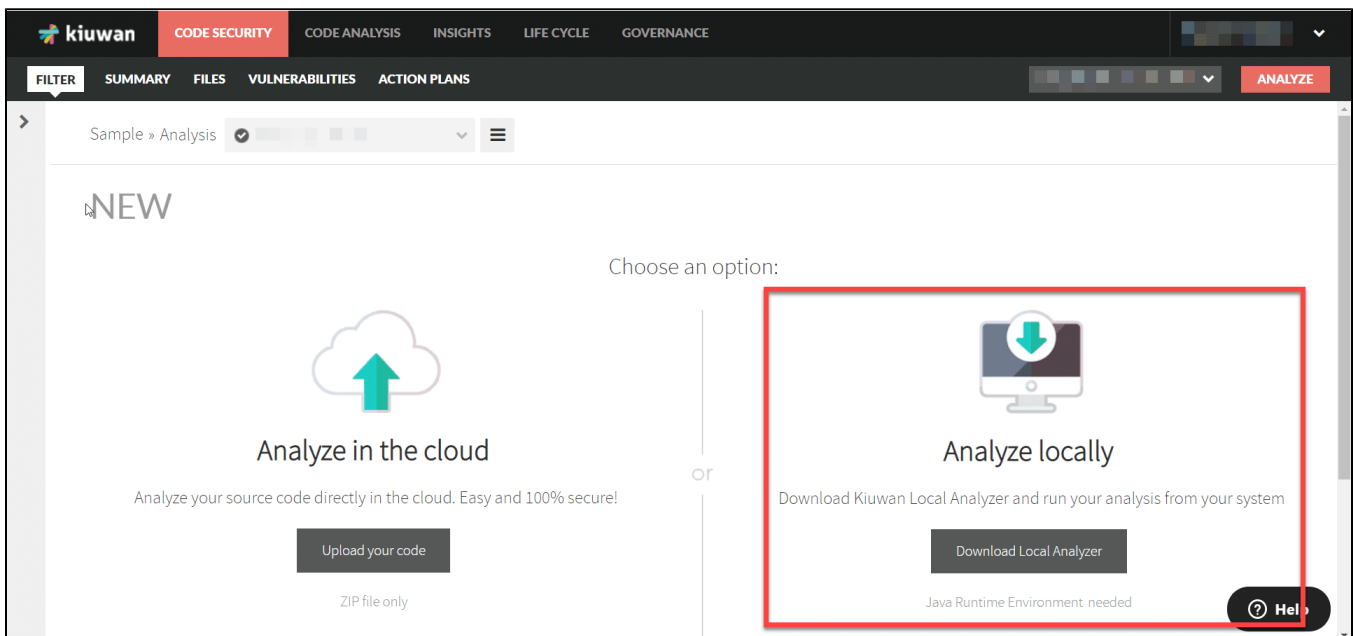
There are **three unparsed files**. Place the cursor over each row and the tooltip will display the parser error.

Normally, these errors are due to badly constructed files (they do not compile) or a specific statement that is not supported by our analyzers.

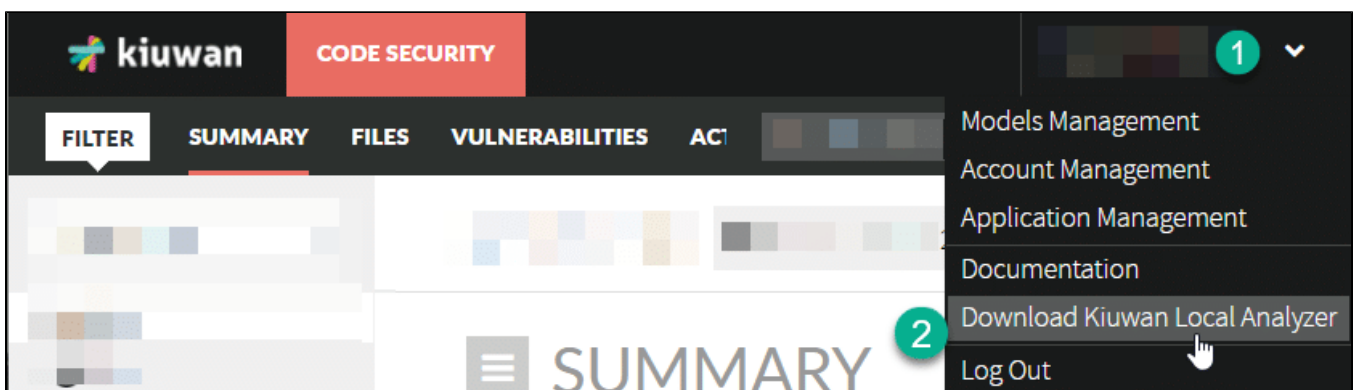
But when we analyze C or C ++, we must **ensure** that we have **fully declared macros** and **directories where header files could be found** before declaring that these files are wrong.

## Analyzing locally

This extra configuration cannot be done when analyzing in the cloud, so you **have to use the Kiuwan Local Analyzer**, which you can download from the new analysis screen:



Or from your management drop-down menu:



Once the local analysis is complete, you can check the temp folder under Kiuwan Local Analyzer **installation directory**. You can find a new directory for this analysis. In our case: `%KiuwanLocalAnalyzer%\temp\linux-ftp-82232984`

Here are the **log files** (certainly not very user friendly) that help us to find the causes of parsing errors.

The file `.unresolved.headers.log` has the list of **header files** that were not found during the analysis. These files are not mandatory for a successful analysis, but they **can help you to know where you have declared some macros** that are subsequently used.

Our first parsing error.

```
Cannot parse C:\_dev\c\linux-ftp-0.17\ftp\extern.h, due to: Parse error at line 38, column 1. Encountered: void
line[38]: void blkfree __P((char **));
```

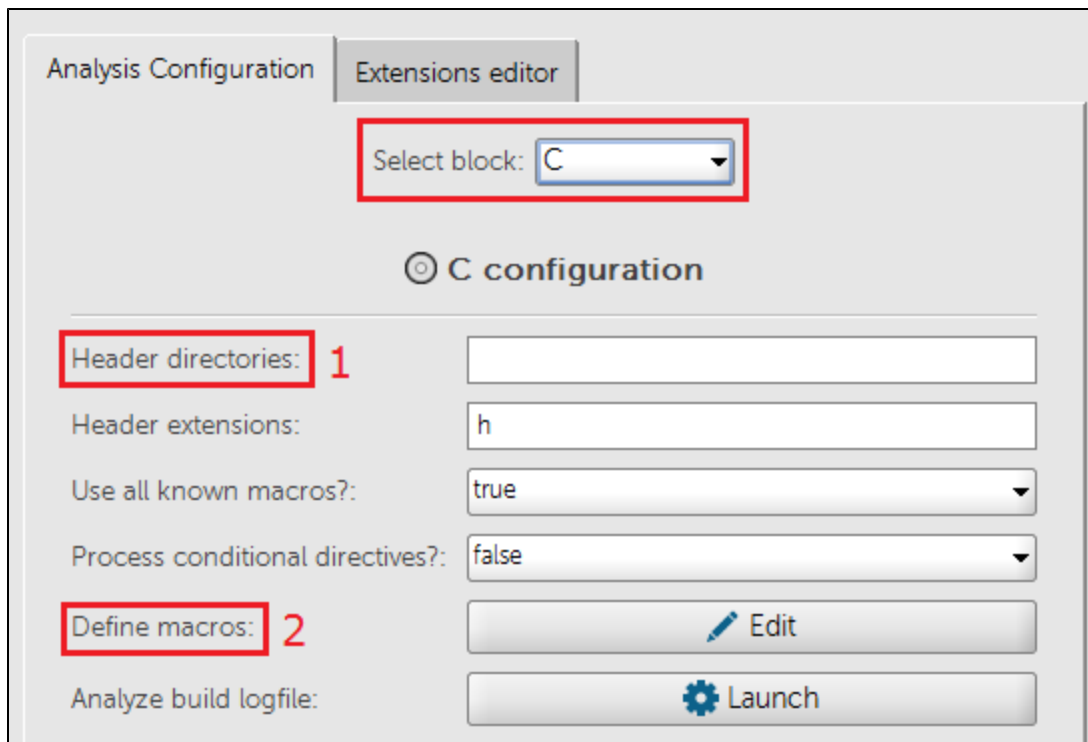
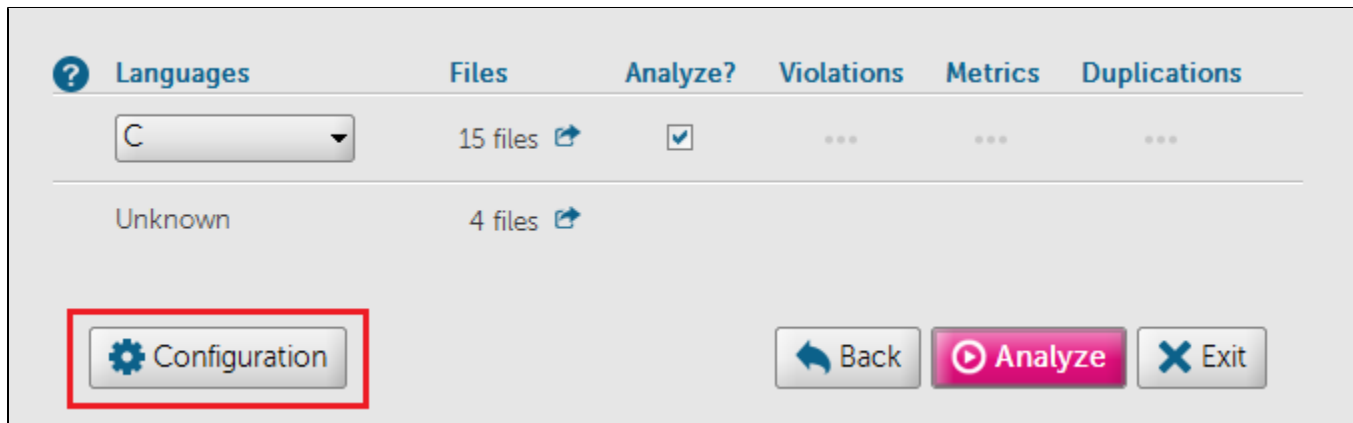
This error is due to the macro `__P`, which was not found during the analysis. This symbol is known as a parameter wrapper. It is a kind of macro, often used in sources that are **meant to be compatible** with pre-ANSI compilers to protect parameter declarations in function prototypes.

This macro, in our system, is located in `/usr/include/x86_64-linux-gnu/sys/cdefs.h` file, and is defined as:

```
#define __P(args) args
```

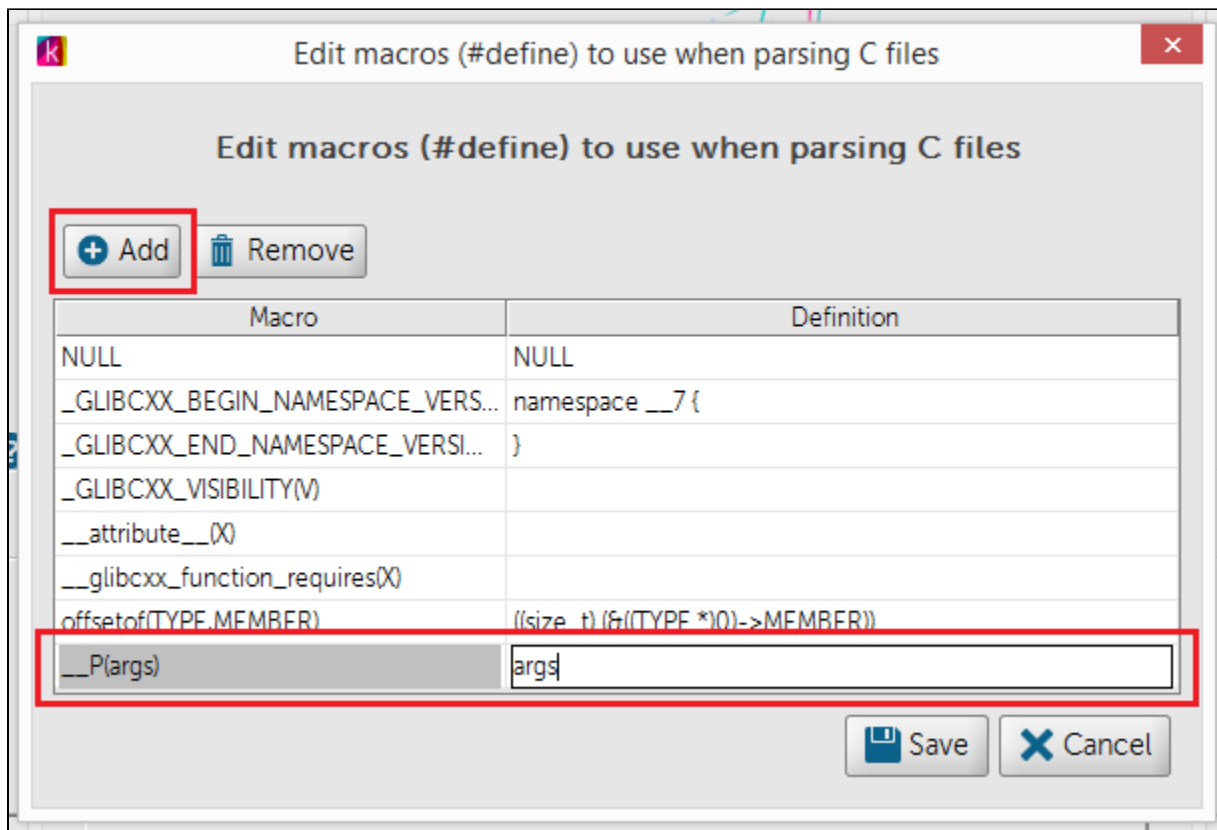
The `sys/cdefs.h` file was one of the listed in the log file `c.unresolved.headers.log`.

To solve this problem, we can edit the configuration for this application. On Kiuwan Local Analyzer's 'Analyze screen', click on **Configuration**:



We have 2 options:

1. **Edit the 'Header directories' entry**, where you can set a comma-separated directories list (absolute or relative to source directories), which includes files that could be found. This is a good option if you are analyzing in the **same machine** where the code is compiled and you have access to all source code dependencies.
2. Go to the **'Macro definition section'**, and click on **Edit**. On the new screen, you can define this new macro:



In both cases, this configuration is saved for subsequent analyses, so the configuration is a 'one-time' action.

Let's go to the second error.

Once one error is fixed, we need to **analyze it again**, since **some errors are hidden or caused by another one**. In the new log file, after the second analysis, we get:


Cannot parse C:\\_dev\c\linux-ftp-0.17\ftp\ftpd.c, due to: Parse error at line 1644, column 1. Encountered: reply  
line[1644]: reply(int n, char \*fmt, va\_dcl va\_alist)

Seeing the code, around line 1644, we find:

```
#ifdef __STDC__
reply(int n, const char *fmt, ...)
#else
reply(int n, char *fmt, va_dcl va_alist)
#endif
```

Our analyzer does not support the LEGACY mode to handle variable argument lists used in va\_dcl va\_alist.

To skip this definition, we can **define the macro \_\_STDC\_\_, with value 1**, as seen before, and ask KIUWAN to process the preprocessor conditional directives.

 **C configuration**

Header directories:

Header extensions:

h


Use all known macros?:

true


Process conditional directives?:

true

Define macros:

 Edit

Analyze build logfile:

 Launch

After this last change all files were processed, so we have finished our work.

## Conclusion

**In short**, when we have problems in our C analysis:

1. **Analyze locally**: Local Kiuwan Analyzer
2. **Resolve parsing errors** whose origin is due to configuration problems:
  - Review unresolved.headers.log
  - Include Header directories
  - Define macros
  - Whether we need to process conditional directives