

Cif - Information flow in C++

github.com/Teemperor/Cif

Raphael Isemann - Group 26 - 21.05.2018

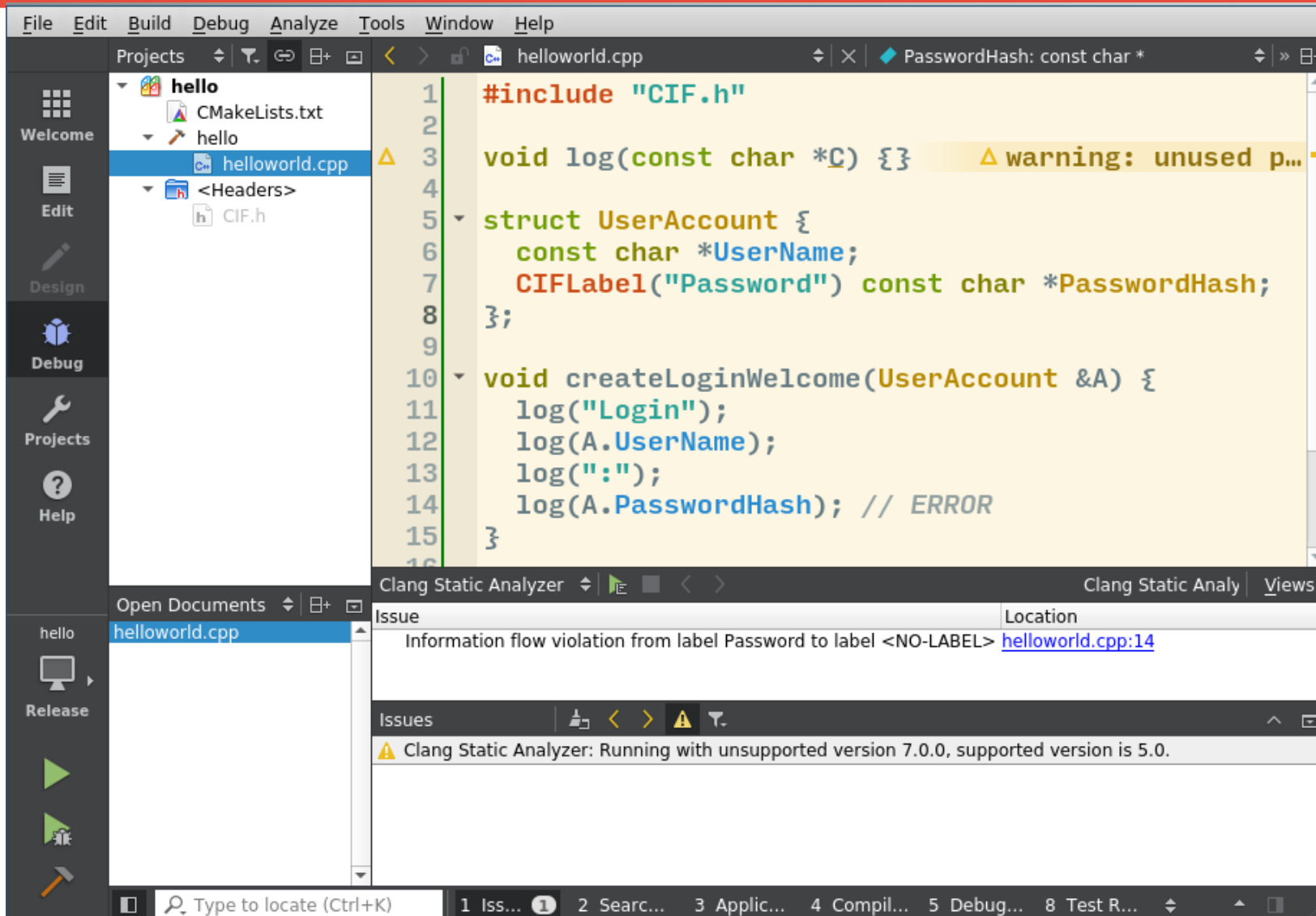
Cif - “Jif for C++”

- **brings information flow security to C++ and other C-languages.**
- **decentral label model + program verifier.**
- **Cif is inspired by Jif for Java, but**
 - is only enforcing information flow in static code, but not during runtime.
 - works with real modern code bases.
 - is actually normal C++, not a dialect like Jif/Java.
 - can be integrated into existing tooling (Qtcreator, Xcode, scan-build reports).
 - Doesn't separate integrity/confidentiality concerns.

Simple Cif example

```
void log(const char *C);  
struct UserAccount {  
    const char *UserName;  
    CIFLabel("Password") const char *PasswordHash;  
};  
void logLogin(UserAccount &A) {  
    log("Login"); log(A.UserName);  
    Log(":" ); log(A.PasswordHash); // ERROR  
}
```

Simple Cif example (QtCreator)



The screenshot shows the Qt Creator IDE with the following content:

```
File Edit Build Debug Analyze Tools Window Help
Projects helloworld.cpp PasswordHash: const char *
hello
  CMakeLists.txt
  hello
  helloworld.cpp
  <Headers>
  CIF.h
1 #include "CIF.h"
2
3 void log(const char *C) {}  Δ warning: unused p...
4
5 struct UserAccount {
6     const char *UserName;
7     CIFLabel("Password") const char *PasswordHash;
8 };
9
10 void createLoginWelcome(UserAccount &A) {
11     log("Login");
12     log(A.UserName);
13     log(":");
14     log(A.PasswordHash); // ERROR
15 }
16
```

Clang Static Analyzer Clang Static Analy Views

Issue	Location
Information flow violation from label Password to label <NO-LABEL>	helloworld.cpp:14

Issues

Δ Clang Static Analyzer: Running with unsupported version 7.0.0, supported version is 5.0.

Type to locate (Ctrl+K) 1 Iss... 2 Search... 3 Applic... 4 Compil... 5 Debug... 8 Test R...

Using Clang for C-language tooling

- **The Cif verifier is implemented as a Clang static analyzer checker.**
- **Clang is the C-language frontend for LLVM.**
 - Widely used for compiling and analyzing code.
- **Clang supports “C-languages” like C, C++, Obj-C, Obj-C++, OpenCL and their dialects.**
 - Cif could support all of them!
 - But so far C/C++ language features are modelled.
- **Many tools support clang integration, so they also support Cif.**

The Cif verifier

- **Searches for information flow in the AST.**
- **Found flows are verified to follow information flow policy.**
- **Enforces direct flow in all kind of assignments**
 - In assignment operators, function calls, etc.
- **Enforces implicit flow**
 - Branching on a label: `if(high) {...}`
- **Only checks one translation unit at once.**
- **Analyzes a project while building it.**

Cif rules

- **Declarations can be labelled multiple labels (e.g. “A,B,Foo”)**
- **Flow only allowed if target is classified stronger or equal to the source.**
- **Classes inherit their labels to fields/subclasses.**
 - `Class CIFLabel(“Pass”) PasswordManager { ... };`
- **Structs and “pure” classes can have dynamic labels.**
 - `CIFLabel(“A”) std::vector<int> a;`
- **Pure functions return combined labels of the passed arguments.**

Cif syntax

- **CIFLabel("A,B") int x;**

- Assigns label to constructs like a variable declarations, parameter, function return parameter, class, struct, etc.

- **CIFDeclassify("A→B", x)**

- Declassifies information in a controlled fashion.

- **CIFPure / CIFPureList**

- Marks functions as pure.
- Marks classes as containers with dynamic labels.

- **CIFOut**

- Designates 'out' parameter for functions.

- **Implemented as macros, invisible for other CCs.**

Cif in the real world

- Cif was tested on real projects like OpenSSH, Apache HTTPD, LigHTTPD, nginx.
- Didn't found any vulnerabilities.
- But it did find suspicious code like this:

```
ngx_log_debug3(NGX_LOG_DEBUG_HTTP, r->connection->log, 0,  
              "rc: %i user: \"%V\" salt: \"%s\"",  
              rc, &r->headers_in.user, passwd->data);
```

- **Code:** *github.com/Teemperor/CifExamples*