Using Clang LibASTMatchers for Compliance in Codebases

Jonah Jolley

Compliance

1. Regulatory
    a. Laws, Standards, and Regulations set by a governing body
2. Organizational
    a. Policies put forth by internal departments

Failing Compliance

1. Safety jeopardized
2. Quality suffers
3. Trust is eroded
4. Fines and disciplinary action

Noncompliance (CAPA)

- Investigation
- Correction
- Prevention

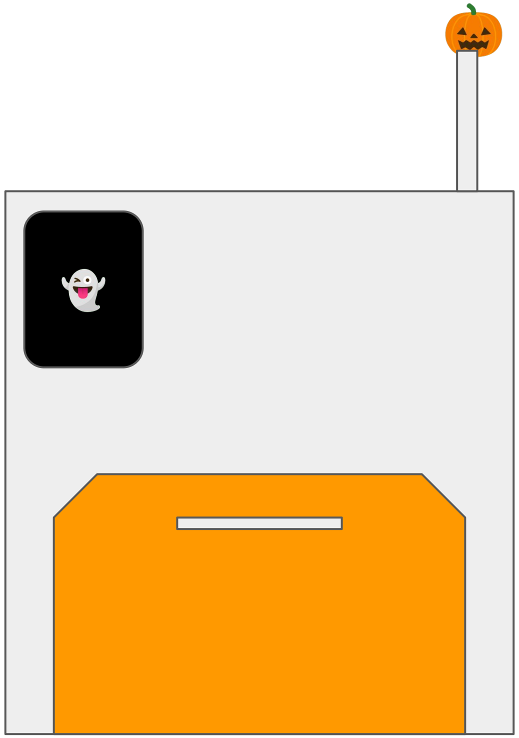
How can we comply

- Education
- Documentation
- Audit
- Automation

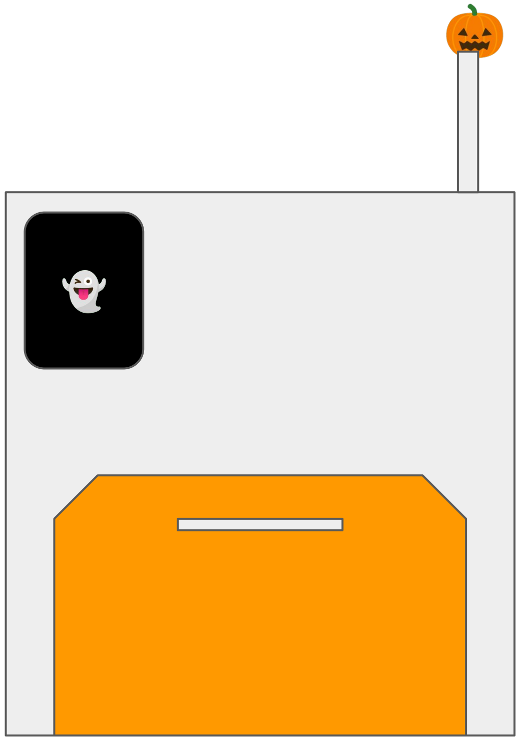
How is this relevant

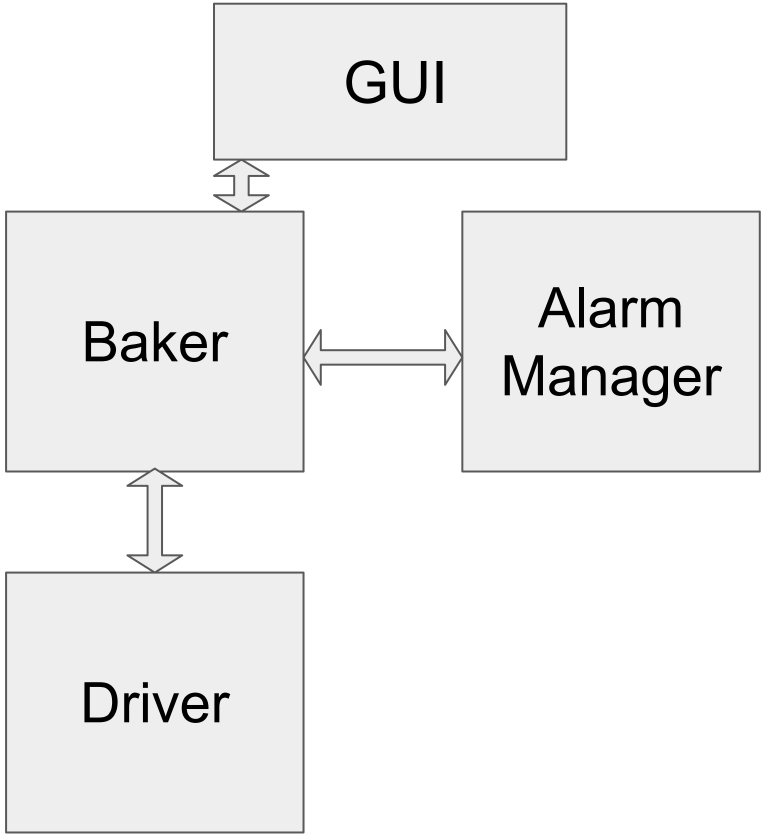
- Documentation must be submitted to a governing body
- The codebase didn’t accurately reflect what was documented
- There could be a potential unmitigated unsafe condition

Device Architecture



Device Architecture





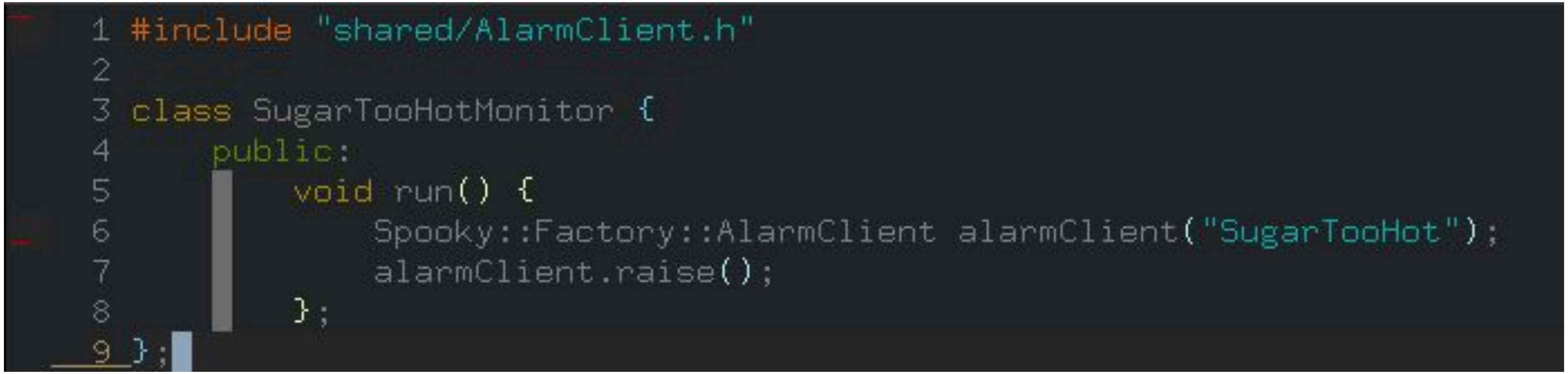
Device Architecture

- Alarms that describe an unsafe condition
- Monitors that ensure device conditions stay nominal
- Alarm Manager that will enforce corrective action

What we want

- Codebase and Documentation stay consistent
- Ensure every Alarm defined in the config file is used
- Every Alarm is raised

How do we solve this?



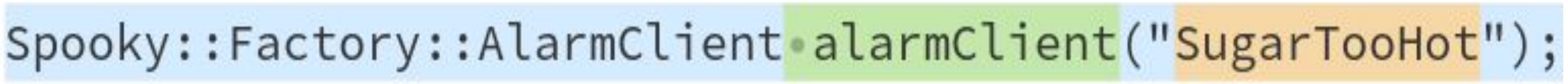
```
#include "shared/AlarmClient.h"

class SugarTooHotMonitor {
    public:
        void run() {
            Spooky::Factory::AlarmClient alarmClient("SugarTooHot");
            alarmClient.raise();
        };
};
```

Manual Process

- Takes person hours to maintain
- Error prone
- Existing tools are cumbersome

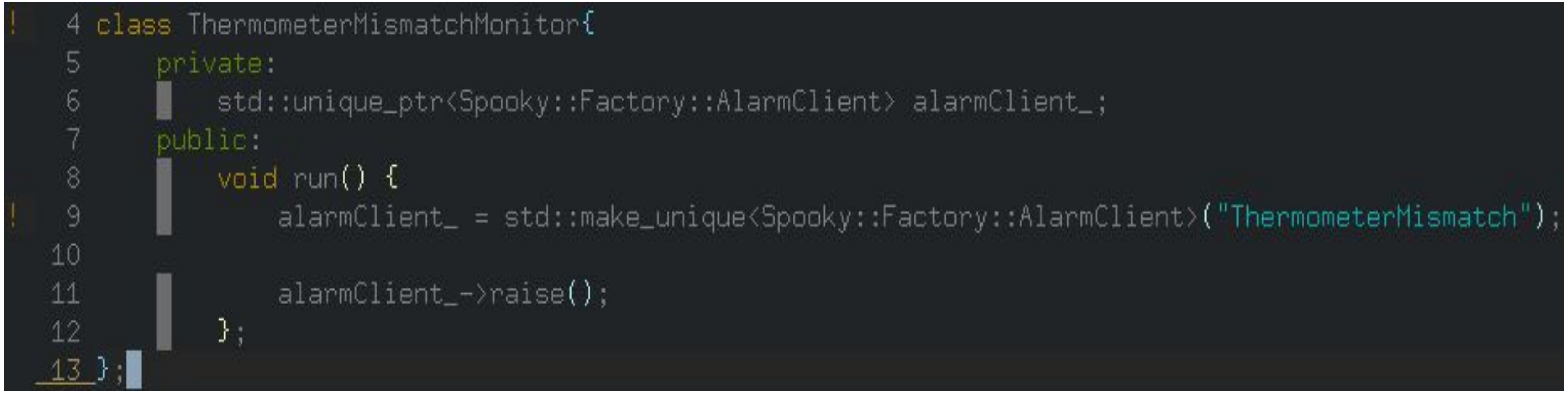
Regex

```
.*AlarmClient(.*)\(\"(.*)\"\);
```



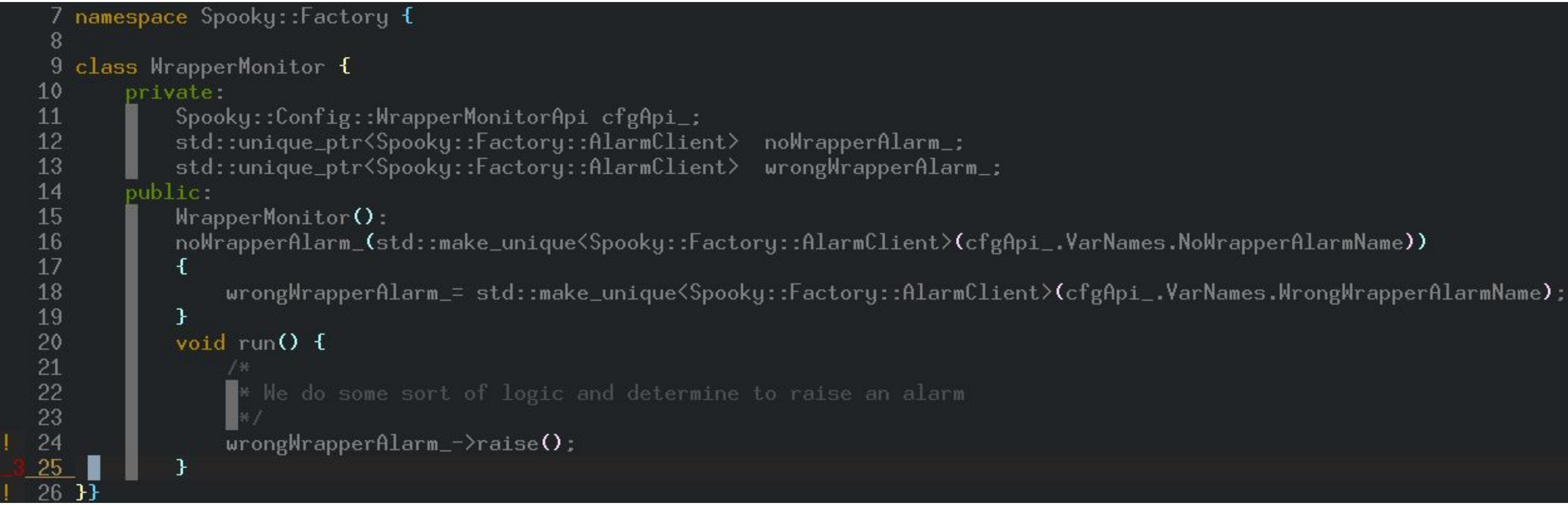
```
Spooky::Factory::AlarmClient alarmClient("SugarTooHot");
```

Regex



```
class ThermometerMismatchMonitor{
    private:
        std::unique_ptr<Spooky::Factory::AlarmClient> alarmClient_;
    public:
        void run() {
            alarmClient_ = std::make_unique<Spooky::Factory::AlarmClient>("ThermometerMismatch");

            alarmClient_->raise();
        };
};
```

Regex



```cpp
namespace Spooky::Factory {

class WrapperMonitor {
    private:
        Spooky::Config::WrapperMonitorApi cfgApi_;
        std::unique_ptr<Spooky::Factory::AlarmClient>  noWrapperAlarm_;
        std::unique_ptr<Spooky::Factory::AlarmClient>  wrongWrapperAlarm_;
    public:
        WrapperMonitor():
        noWrapperAlarm_(std::make_unique<Spooky::Factory::AlarmClient>(cfgApi_.VarNames.NoWrapperAlarmName))
        {
            wrongWrapperAlarm_= std::make_unique<Spooky::Factory::AlarmClient>(cfgApi_.VarNames.WrongWrapperAlarmName);
        }
        void run() {
            /*
             * We do some sort of logic and determine to raise an alarm
             */
            wrongWrapperAlarm_->raise();
        }
}}
```

Regex

- Brittle
- Complex to maintain
- Low confidence we are continually capturing every case
- Difficult to capture all the information

Clang Tooling

- Clang-Tidy
- LibTooling
    - Control the output of the program
    - Specialize it for what we need
    - Enable being able to post process with additional tooling

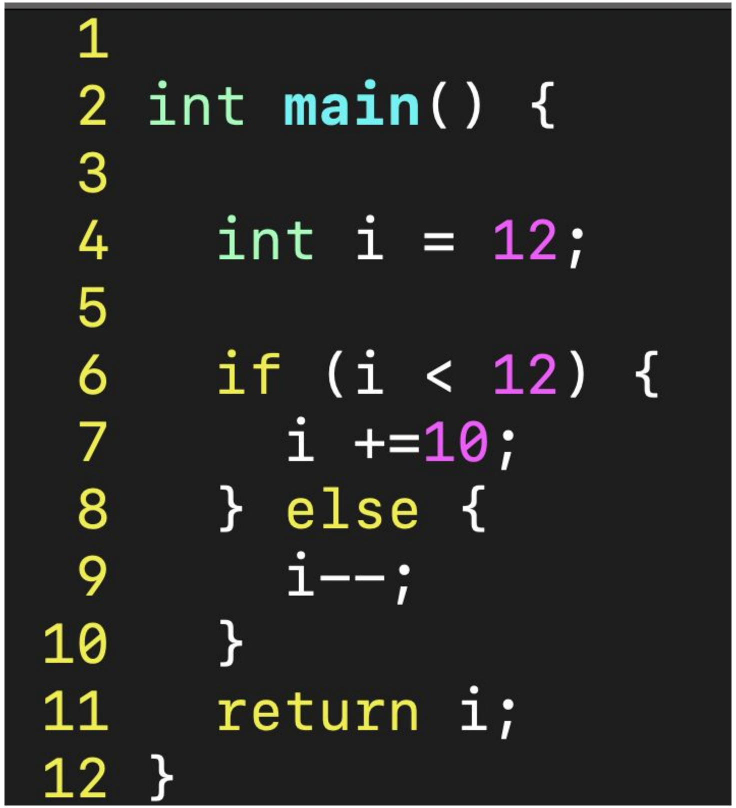
What is Clang LibTooling

- Library to support writing standalone tools
- Allows us to run tools over single files or subsets of files
- Gives us full control and access of the Clang AST
- Allows us to share code with Clang Plugins

Clang AST

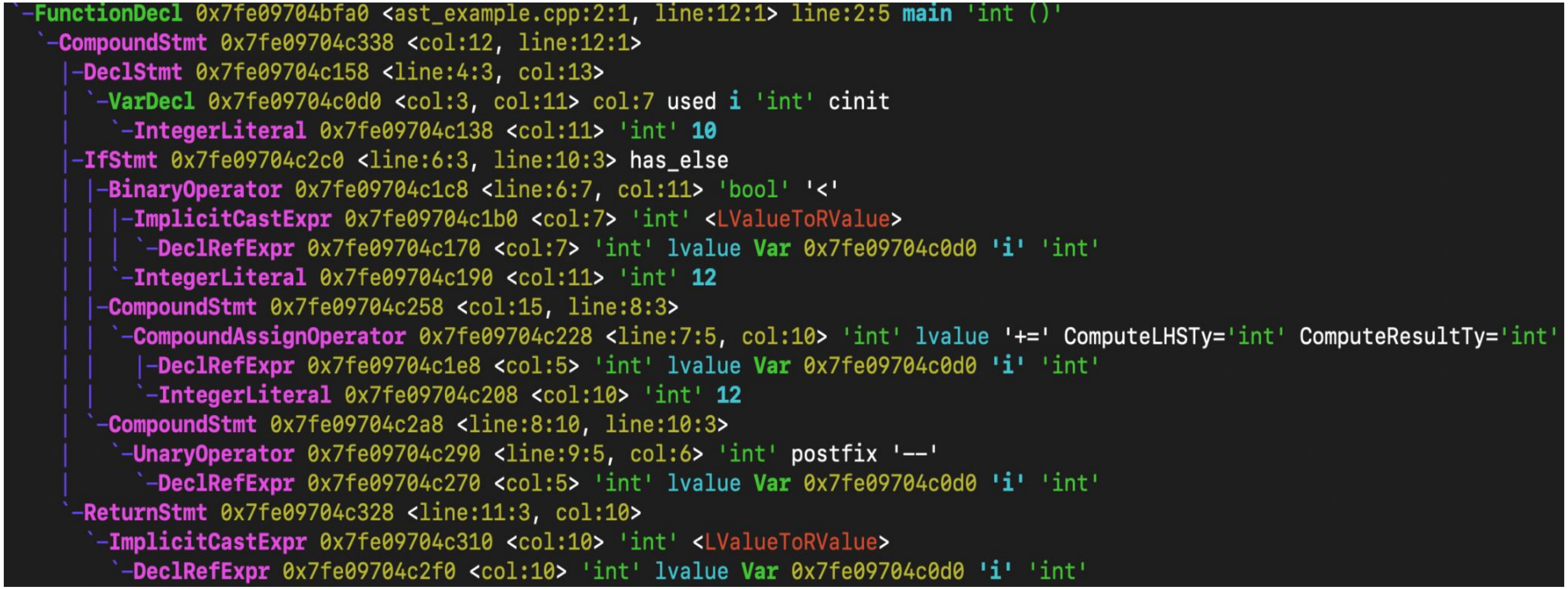
- Frontend
- ASTContext
- Core Classes
    - Decl
    - Stmt
    - Type

Clang AST



```

int main() {

  int i = 12;

  if (i < 12) {
    i +=10;
  } else {
    i--;
  }
  return i;
}
```

Clang AST



```
`-FunctionDecl 0x7fe09704bfa0 <ast_example.cpp:2:1, line:12:1> line:2:5 main 'int ()'
  `-CompoundStmt 0x7fe09704c338 <col:12, line:12:1>
    |-DeclStmt 0x7fe09704c158 <line:4:3, col:13>
    | `-VarDecl 0x7fe09704c0d0 <col:3, col:11> col:7 used i 'int' cinit
    |   `-IntegerLiteral 0x7fe09704c138 <col:11> 'int' 10
    |-IfStmt 0x7fe09704c2c0 <line:6:3, line:10:3> has_else
    | |-BinaryOperator 0x7fe09704c1c8 <line:6:7, col:11> 'bool' '<'
    | | |-ImplicitCastExpr 0x7fe09704c1b0 <col:7> 'int' <LValueToRValue>
    | | | `-DeclRefExpr 0x7fe09704c170 <col:7> 'int' lvalue Var 0x7fe09704c0d0 'i' 'int'
    | | `-IntegerLiteral 0x7fe09704c190 <col:11> 'int' 12
    | |-CompoundStmt 0x7fe09704c258 <col:15, line:8:3>
    | | `-CompoundAssignOperator 0x7fe09704c228 <line:7:5, col:10> 'int' lvalue '+=' ComputeLHSTy='int' ComputeResultTy='int'
    | |   |-DeclRefExpr 0x7fe09704c1e8 <col:5> 'int' lvalue Var 0x7fe09704c0d0 'i' 'int'
    | |   `-IntegerLiteral 0x7fe09704c208 <col:10> 'int' 12
    | `-CompoundStmt 0x7fe09704c2a8 <line:8:10, line:10:3>
    |   `-UnaryOperator 0x7fe09704c290 <line:9:5, col:6> 'int' postfix '--'
    |     `-DeclRefExpr 0x7fe09704c270 <col:5> 'int' lvalue Var 0x7fe09704c0d0 'i' 'int'
    `-ReturnStmt 0x7fe09704c328 <line:11:3, col:10>
      `-ImplicitCastExpr 0x7fe09704c310 <col:10> 'int' <LValueToRValue>
        `-DeclRefExpr 0x7fe09704c2f0 <col:10> 'int' lvalue Var 0x7fe09704c0d0 'i' 'int'
```

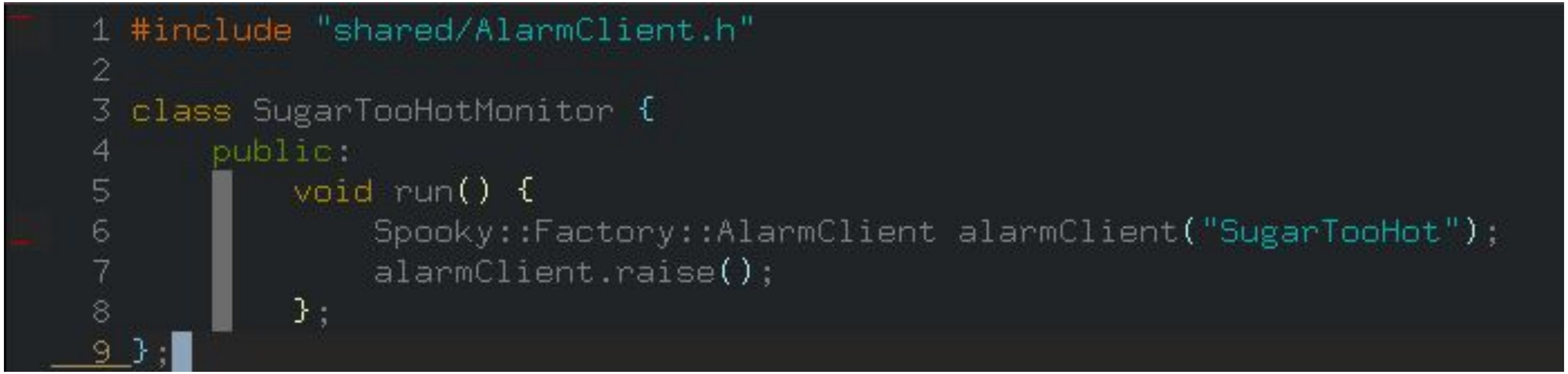
Lib ASTMatcher

- Domain Specific Language
- Use a MatchCallback to access a matched predicate
- Three basic categories
  - Node Matchers
  - Narrowing Matchers
  - Traversal Matchers

Let’s write a matcher together

- Dump the ast
- Use clang-query to write a matcher
- See how specific we can get. Bind it
- Compile
- Run it in a debugger inspect the code and extract the relevant pieces
- Repeat

Let’s write a matcher together

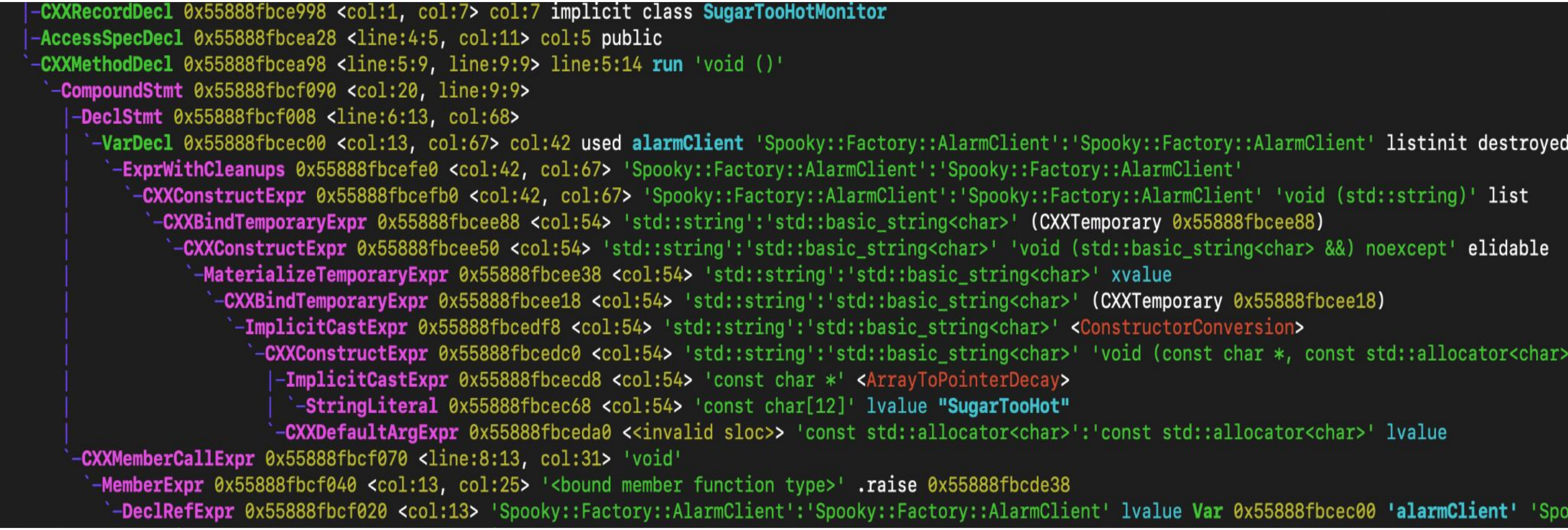


```
#include "shared/AlarmClient.h"

class SugarTooHotMonitor {
    public:
        void run() {
            Spooky::Factory::AlarmClient alarmClient("SugarTooHot");
            alarmClient.raise();
        };
};
```

Dump AST

```
clang -Xclang -ast-dump -fsyntax-only (-fno-color-diagnostics) SugarTooHotMonitor.cpp
```

Dump AST



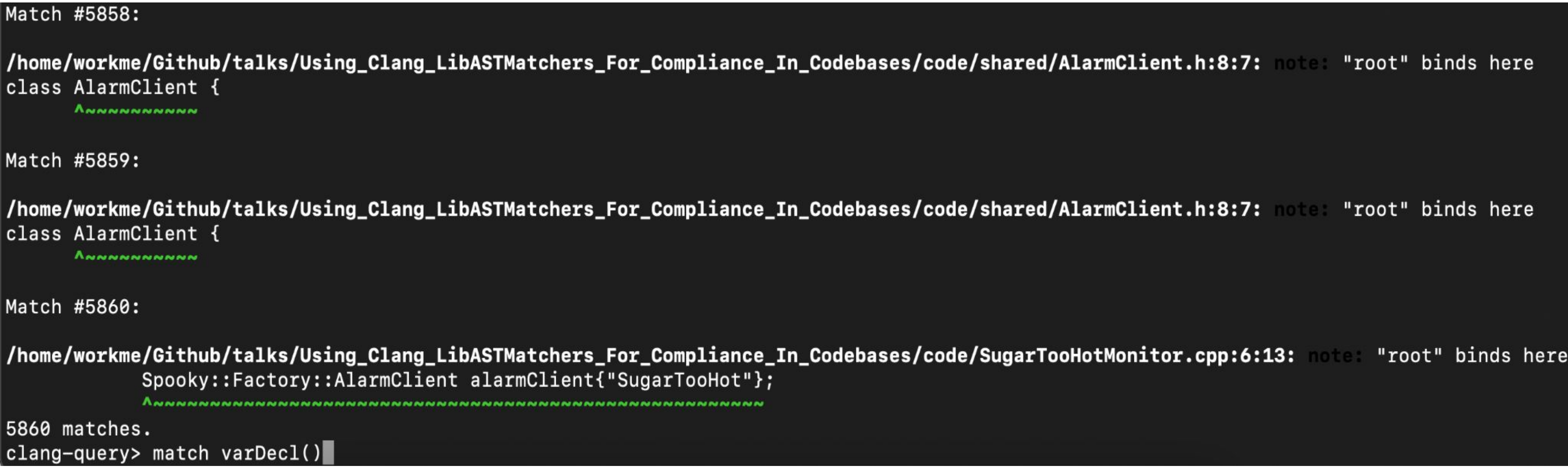
```
|-CXXRecordDecl 0x55888fbce998 <col:1, col:7> col:7 implicit class SugarTooHotMonitor
|-AccessSpecDecl 0x55888fbcea28 <line:4:5, col:11> col:5 public
`-CXXMethodDecl 0x55888fbcea98 <line:5:9, line:9:9> line:5:14 run 'void ()'
  `-CompoundStmt 0x55888fbcf090 <col:20, line:9:9>
    |-DeclStmt 0x55888fbcf008 <line:6:13, col:68>
    | `-VarDecl 0x55888fbcec00 <col:13, col:67> col:42 used alarmClient 'Spooky::Factory::AlarmClient':'Spooky::Factory::AlarmClient' listinit destroyed
    |   `-ExprWithCleanups 0x55888fbcefe0 <col:42, col:67> 'Spooky::Factory::AlarmClient':'Spooky::Factory::AlarmClient'
    |     `-CXXConstructExpr 0x55888fbcefb0 <col:42, col:67> 'Spooky::Factory::AlarmClient':'Spooky::Factory::AlarmClient' 'void (std::string)' list
    |       `-CXXBindTemporaryExpr 0x55888fbcee88 <col:54> 'std::string':'std::basic_string<char>' (CXXTemporary 0x55888fbcee88)
    |         `-CXXConstructExpr 0x55888fbcee50 <col:54> 'std::string':'std::basic_string<char>' 'void (std::basic_string<char> &&) noexcept' elidable
    |           `-MaterializeTemporaryExpr 0x55888fbcee38 <col:54> 'std::string':'std::basic_string<char>' xvalue
    |             `-CXXBindTemporaryExpr 0x55888fbcee18 <col:54> 'std::string':'std::basic_string<char>' (CXXTemporary 0x55888fbcee18)
    |               `-ImplicitCastExpr 0x55888fbcedf8 <col:54> 'std::string':'std::basic_string<char>' <ConstructorConversion>
    |                 `-CXXConstructExpr 0x55888fbcedc0 <col:54> 'std::string':'std::basic_string<char>' 'void (const char *, const std::allocator<char>
    |                   |-ImplicitCastExpr 0x55888fbcecd8 <col:54> 'const char *' <ArrayToPointerDecay>
    |                   | `-StringLiteral 0x55888fbcec68 <col:54> 'const char[12]' lvalue "SugarTooHot"
    |                   `-CXXDefaultArgExpr 0x55888fbceda0 <<invalid sloc>> 'const std::allocator<char>':'const std::allocator<char>' lvalue
    `-CXXMemberCallExpr 0x55888fbcf070 <line:8:13, col:31> 'void'
      `-MemberExpr 0x55888fbcf040 <col:13, col:25> '<bound member function type>' .raise 0x55888fbcde38
        `-DeclRefExpr 0x55888fbcf020 <col:13> 'Spooky::Factory::AlarmClient':'Spooky::Factory::AlarmClient' lvalue Var 0x55888fbcec00 'alarmClient' 'Spo
```

Clang-Query

Useful settings

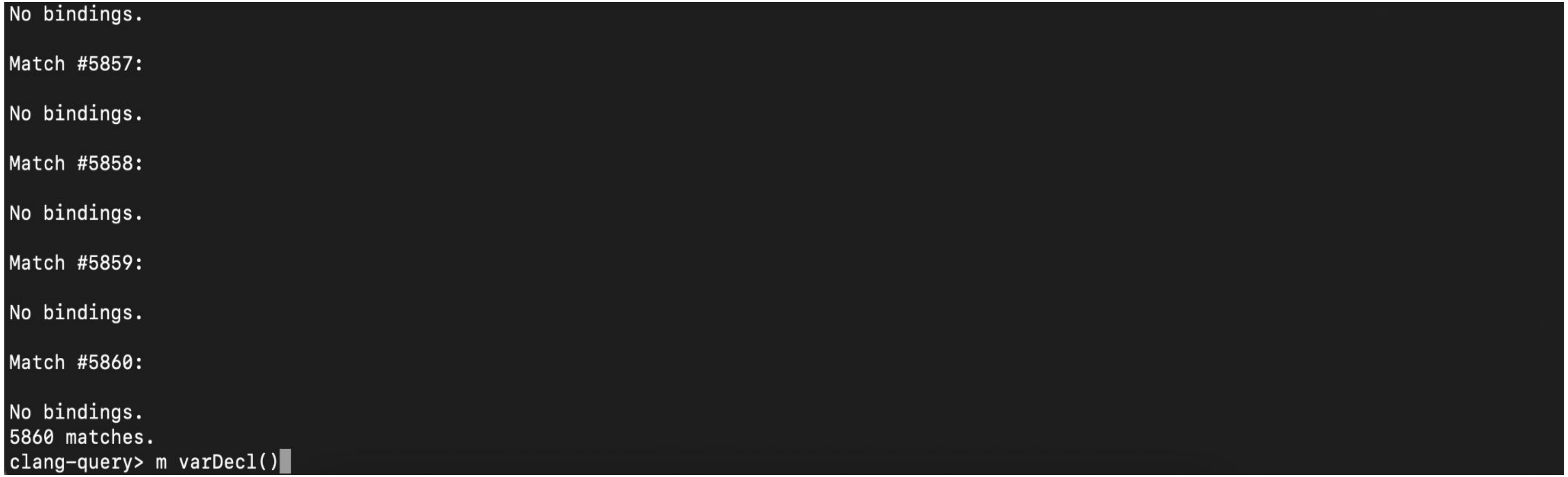
- set bind-root false
- set print-matcher true
- set output diag/dump

Clang-Query



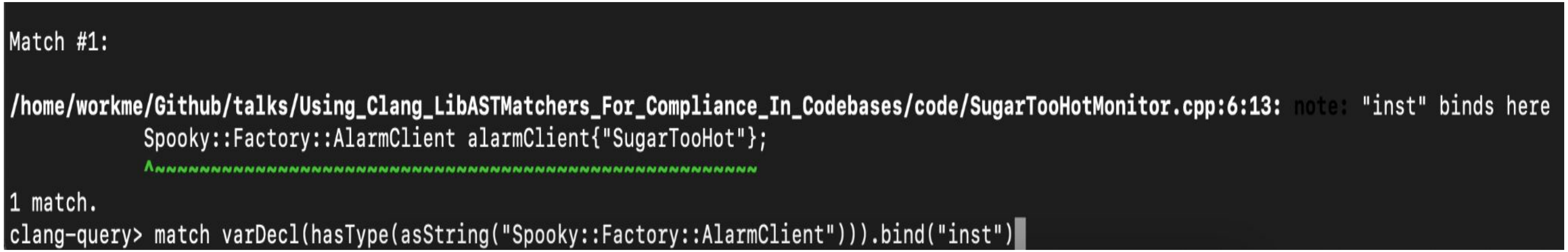
```
Match #5858:

/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/shared/AlarmClient.h:8:7: note: "root" binds here
class AlarmClient {
      ^~~~~~~~~~~


Match #5859:

/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/shared/AlarmClient.h:8:7: note: "root" binds here
class AlarmClient {
      ^~~~~~~~~~~


Match #5860:

/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitor.cpp:6:13: note: "root" binds here
            Spooky::Factory::AlarmClient alarmClient{"SugarTooHot"};
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
5860 matches.
clang-query> match varDecl()
```

Clang-Query



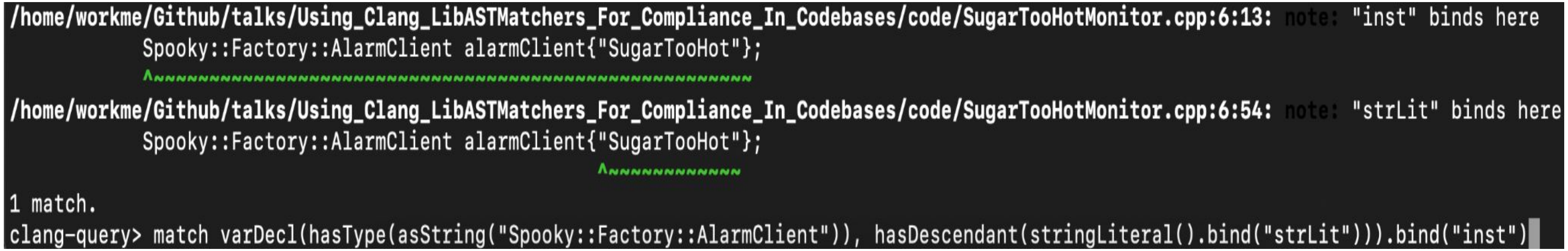
```
No bindings.

Match #5857:

No bindings.

Match #5858:

No bindings.

Match #5859:

No bindings.

Match #5860:

No bindings.
5860 matches.
clang-query> m varDecl()
```

Clang-Query



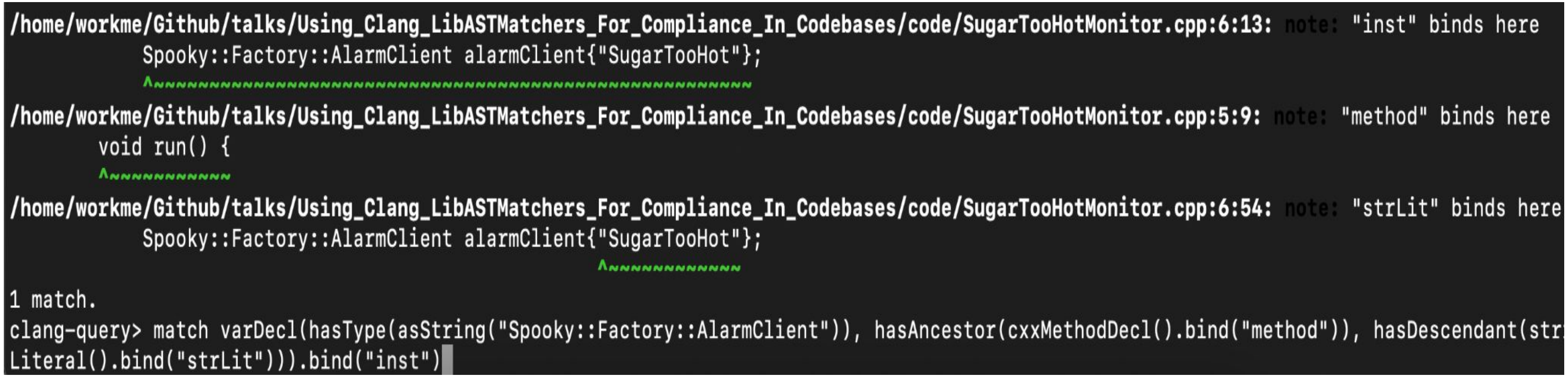
```
Match #1:

/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitor.cpp:6:13: note: "inst" binds here
            Spooky::Factory::AlarmClient alarmClient{"SugarTooHot"};
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 match.
clang-query> match varDecl(hasType(asString("Spooky::Factory::AlarmClient"))).bind("inst")
```

Clang-Query



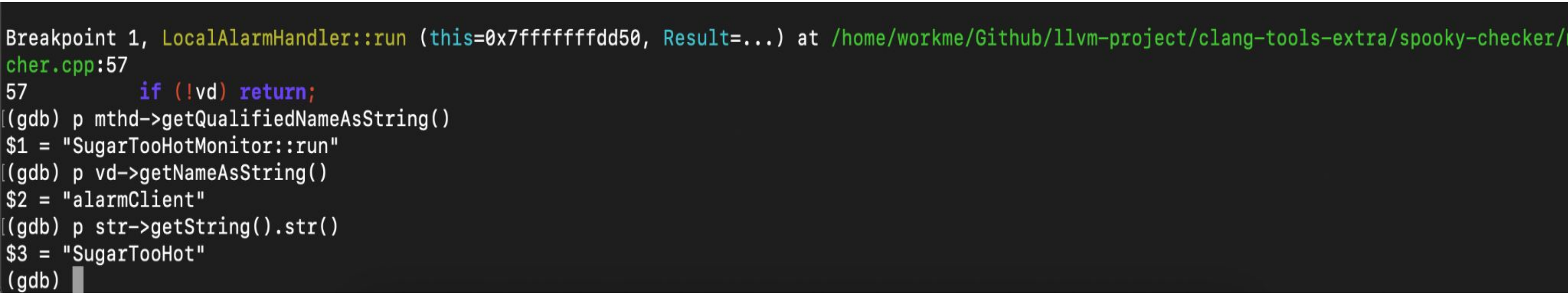
```
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitor.cpp:6:13: note: "inst" binds here
            Spooky::Factory::AlarmClient alarmClient{"SugarTooHot"};
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitor.cpp:6:54: note: "strLit" binds here
            Spooky::Factory::AlarmClient alarmClient{"SugarTooHot"};
                                                     ^~~~~~~~~~~~~

1 match.
clang-query> match varDecl(hasType(asString("Spooky::Factory::AlarmClient")), hasDescendant(stringLiteral().bind("strLit"))).bind("inst")
```

Clang-Query



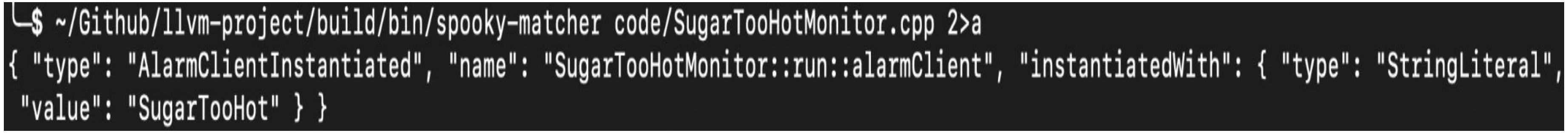
```
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitor.cpp:6:13: note: "inst" binds here
            Spooky::Factory::AlarmClient alarmClient{"SugarTooHot"};
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitor.cpp:5:9: note: "method" binds here
        void run() {
        ^~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitor.cpp:6:54: note: "strLit" binds here
            Spooky::Factory::AlarmClient alarmClient{"SugarTooHot"};
                                                     ^~~~~~~~~~~~~

1 match.
clang-query> match varDecl(hasType(asString("Spooky::Factory::AlarmClient")), hasAncestor(cxxMethodDecl().bind("method")), hasDescendant(str
Literal().bind("strLit"))).bind("inst")
```

Explore in the debugger



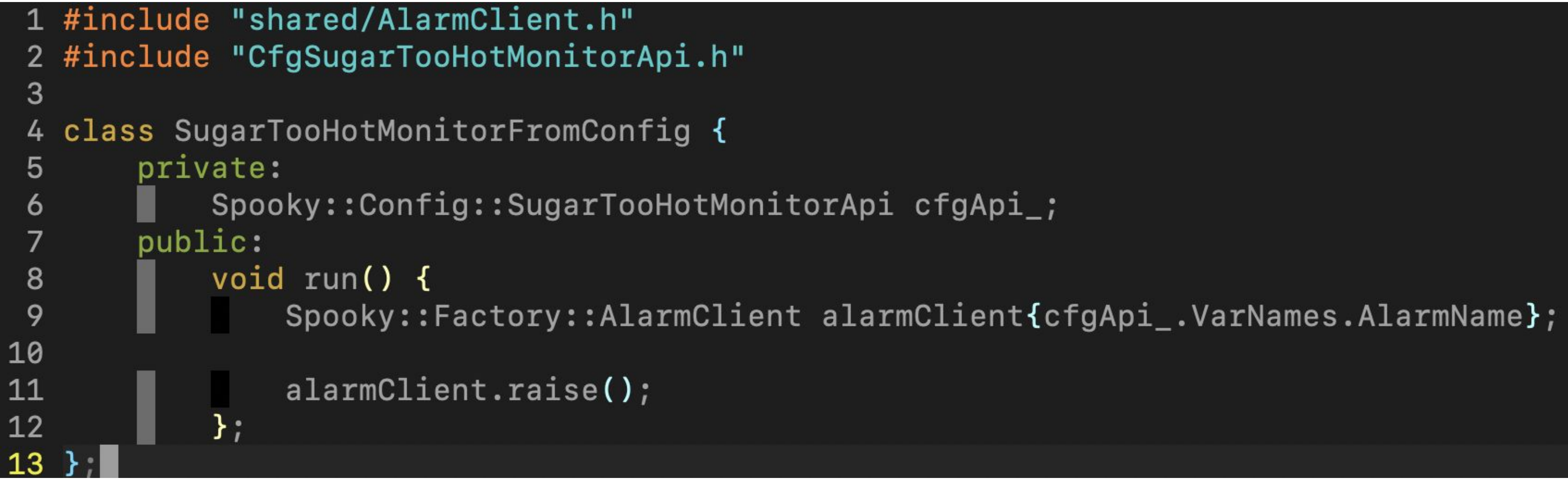
```
Breakpoint 1, LocalAlarmHandler::run (this=0x7fffffffdd50, Result=...) at /home/workme/Github/llvm-project/clang-tools-extra/spooky-checker/
cher.cpp:57
57	          if (!vd) return;
(gdb) p mthd->getQualifiedNameAsString()
$1 = "SugarTooHotMonitor::run"
(gdb) p vd->getNameAsString()
$2 = "alarmClient"
(gdb) p str->getString().str()
$3 = "SugarTooHot"
(gdb) 
```

Running it from matcher



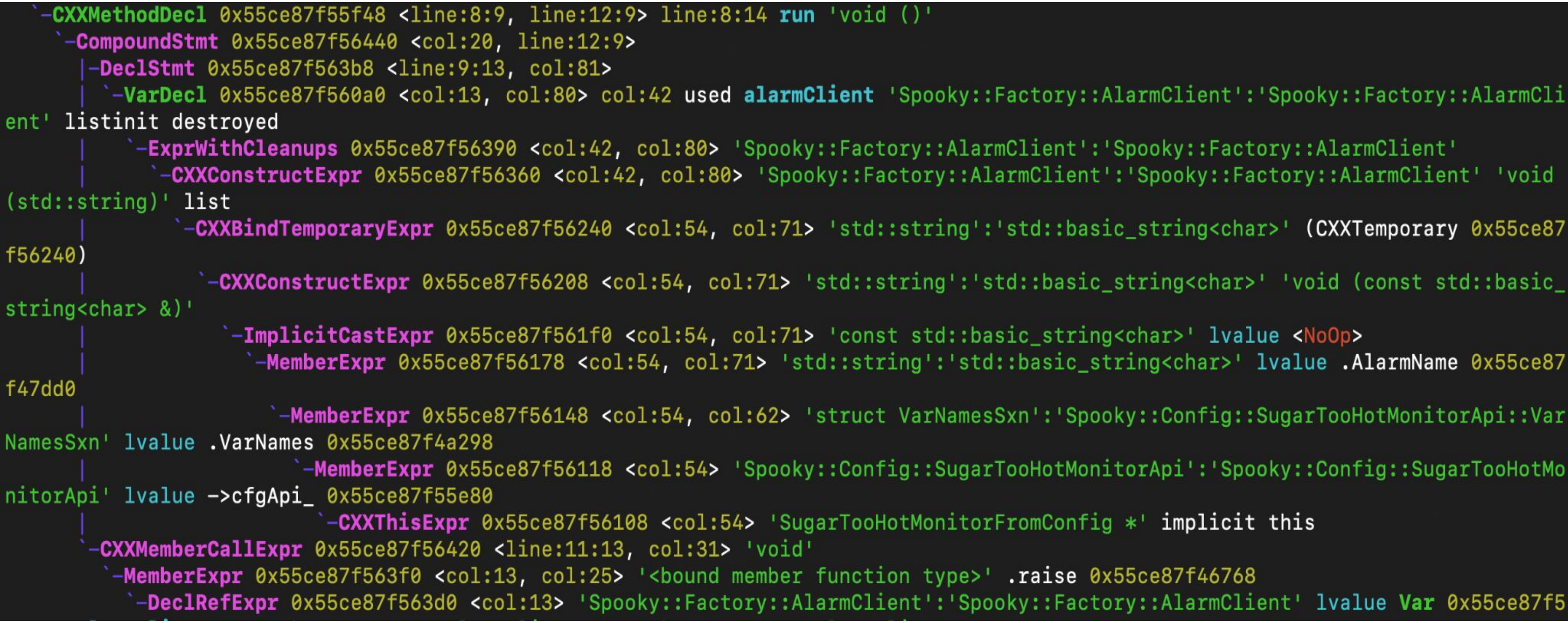
```
└─$ ~/Github/llvm-project/build/bin/spooky-matcher code/SugarTooHotMonitor.cpp 2>a
{ "type": "AlarmClientInstantiated", "name": "SugarTooHotMonitor::run::alarmClient", "instantiatedWith": { "type": "StringLiteral",
 "value": "SugarTooHot" } }
```

Config as input



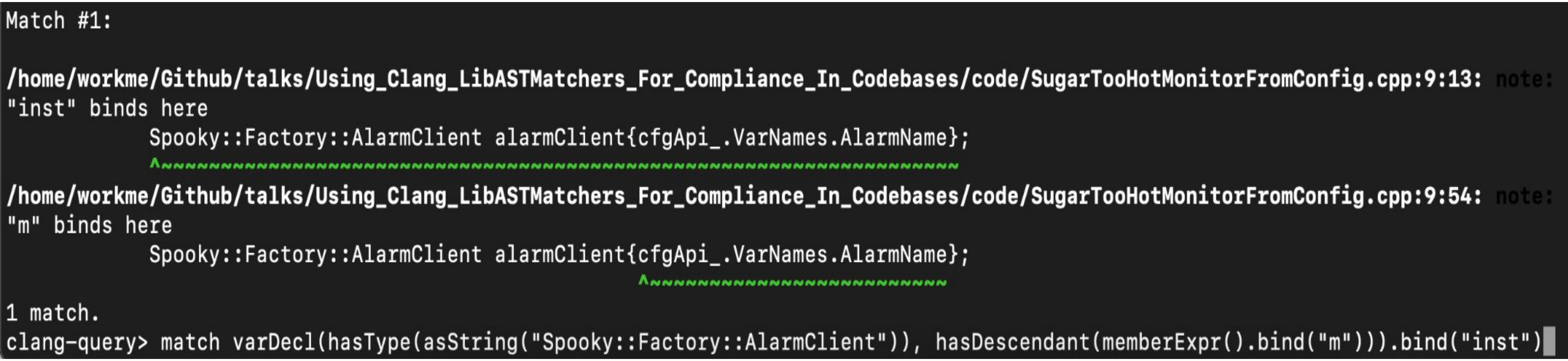
```cpp
#include "shared/AlarmClient.h"
#include "CfgSugarTooHotMonitorApi.h"

class SugarTooHotMonitorFromConfig {
    private:
        Spooky::Config::SugarTooHotMonitorApi cfgApi_;
    public:
        void run() {
            Spooky::Factory::AlarmClient alarmClient{cfgApi_.VarNames.AlarmName};

            alarmClient.raise();
        };
};
```

Config as input



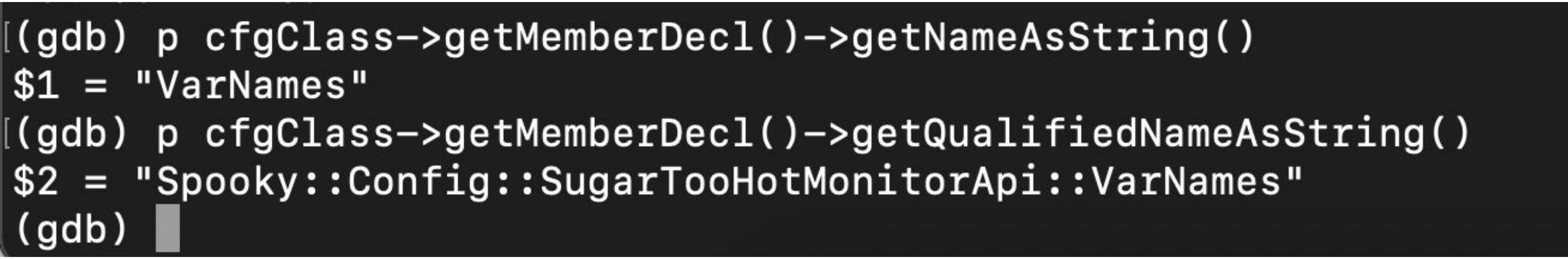
```
  `-CXXMethodDecl 0x55ce87f55f48 <line:8:9, line:12:9> line:8:14 run 'void ()'
    `-CompoundStmt 0x55ce87f56440 <col:20, line:12:9>
      |-DeclStmt 0x55ce87f563b8 <line:9:13, col:81>
      | `-VarDecl 0x55ce87f560a0 <col:13, col:80> col:42 used alarmClient 'Spooky::Factory::AlarmClient':'Spooky::Factory::AlarmCli
ent' listinit destroyed
      |   `-ExprWithCleanups 0x55ce87f56390 <col:42, col:80> 'Spooky::Factory::AlarmClient':'Spooky::Factory::AlarmClient'
      |     `-CXXConstructExpr 0x55ce87f56360 <col:42, col:80> 'Spooky::Factory::AlarmClient':'Spooky::Factory::AlarmClient' 'void
(std::string)' list
      |       `-CXXBindTemporaryExpr 0x55ce87f56240 <col:54, col:71> 'std::string':'std::basic_string<char>' (CXXTemporary 0x55ce87
f56240)
      |         `-CXXConstructExpr 0x55ce87f56208 <col:54, col:71> 'std::string':'std::basic_string<char>' 'void (const std::basic_
string<char> &)'
      |           `-ImplicitCastExpr 0x55ce87f561f0 <col:54, col:71> 'const std::basic_string<char>' lvalue <NoOp>
      |             `-MemberExpr 0x55ce87f56178 <col:54, col:71> 'std::string':'std::basic_string<char>' lvalue .AlarmName 0x55ce87
f47dd0
      |               `-MemberExpr 0x55ce87f56148 <col:54, col:62> 'struct VarNamesSxn':'Spooky::Config::SugarTooHotMonitorApi::Var
NamesSxn' lvalue .VarNames 0x55ce87f4a298
      |                 `-MemberExpr 0x55ce87f56118 <col:54> 'Spooky::Config::SugarTooHotMonitorApi':'Spooky::Config::SugarTooHotMo
nitorApi' lvalue ->cfgApi_ 0x55ce87f55e80
      |                   `-CXXThisExpr 0x55ce87f56108 <col:54> 'SugarTooHotMonitorFromConfig *' implicit this
      `-CXXMemberCallExpr 0x55ce87f56420 <line:11:13, col:31> 'void'
        `-MemberExpr 0x55ce87f563f0 <col:13, col:25> '<bound member function type>' .raise 0x55ce87f46768
          `-DeclRefExpr 0x55ce87f563d0 <col:13> 'Spooky::Factory::AlarmClient':'Spooky::Factory::AlarmClient' lvalue Var 0x55ce87f5
```

Config as input



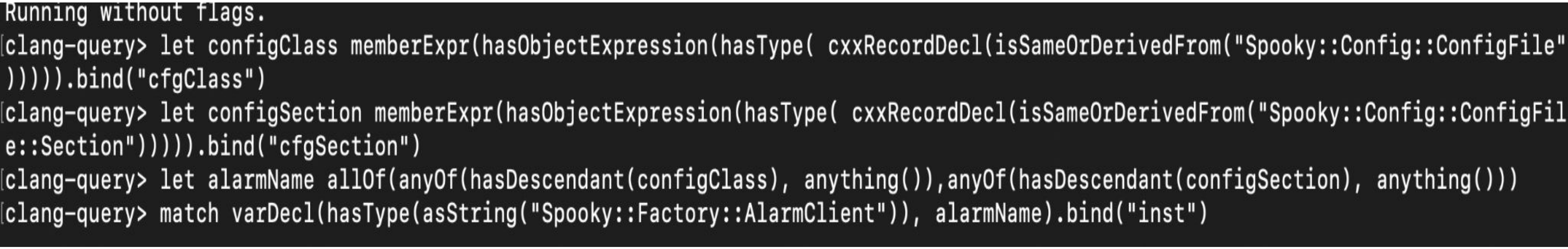
```
Match #1:

/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitorFromConfig.cpp:9:13: note:
"inst" binds here
            Spooky::Factory::AlarmClient alarmClient{cfgApi_.VarNames.AlarmName};
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitorFromConfig.cpp:9:54: note:
"m" binds here
            Spooky::Factory::AlarmClient alarmClient{cfgApi_.VarNames.AlarmName};
                                                     ^~~~~~~~~~~~~~~~~~~~~~~~~~

1 match.
clang-query> match varDecl(hasType(asString("Spooky::Factory::AlarmClient")), hasDescendant(memberExpr().bind("m"))).bind("inst")
```

Config as input



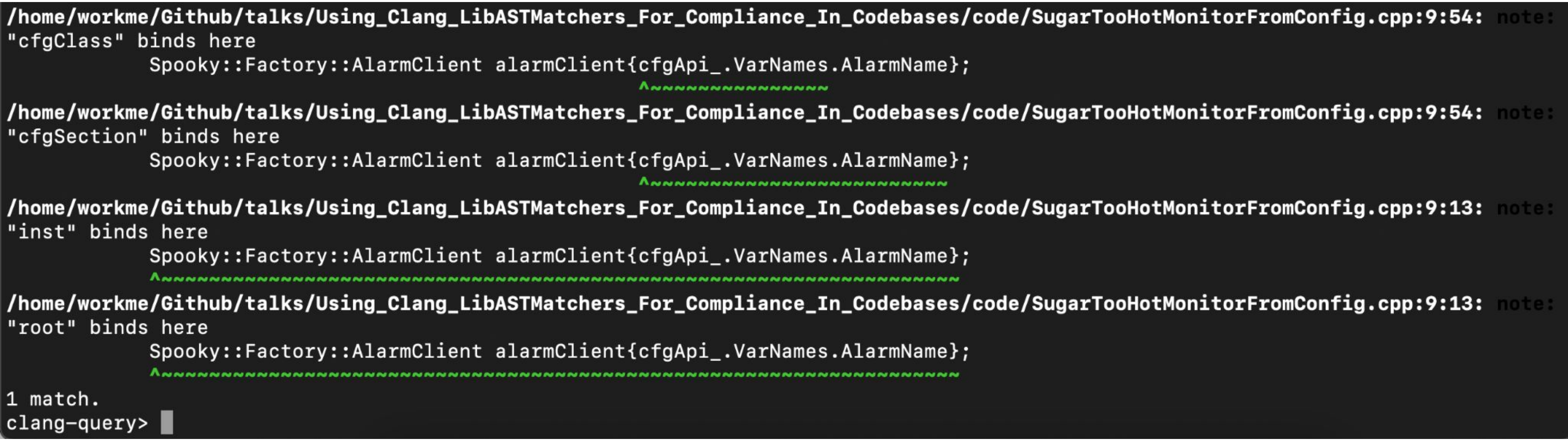
```
(gdb) p cfgClass->getMemberDecl()->getNameAsString()
$1 = "VarNames"
(gdb) p cfgClass->getMemberDecl()->getQualifiedNameAsString()
$2 = "Spooky::Config::SugarTooHotMonitorApi::VarNames"
(gdb) 
```

Config as input



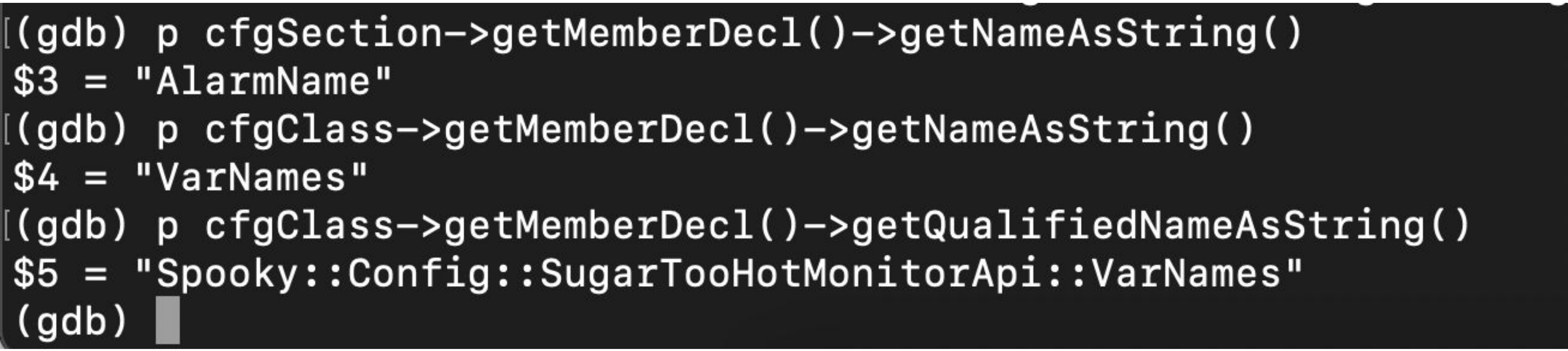
```
Running without flags.
[clang-query> let configClass memberExpr(hasObjectExpression(hasType( cxxRecordDecl(isSameOrDerivedFrom("Spooky::Config::ConfigFile"
))))).bind("cfgClass")
[clang-query> let configSection memberExpr(hasObjectExpression(hasType( cxxRecordDecl(isSameOrDerivedFrom("Spooky::Config::ConfigFil
e::Section"))))).bind("cfgSection")
[clang-query> let alarmName allOf(anyOf(hasDescendant(configClass), anything()),anyOf(hasDescendant(configSection), anything()))
[clang-query> match varDecl(hasType(asString("Spooky::Factory::AlarmClient")), alarmName).bind("inst")
```

Config as input



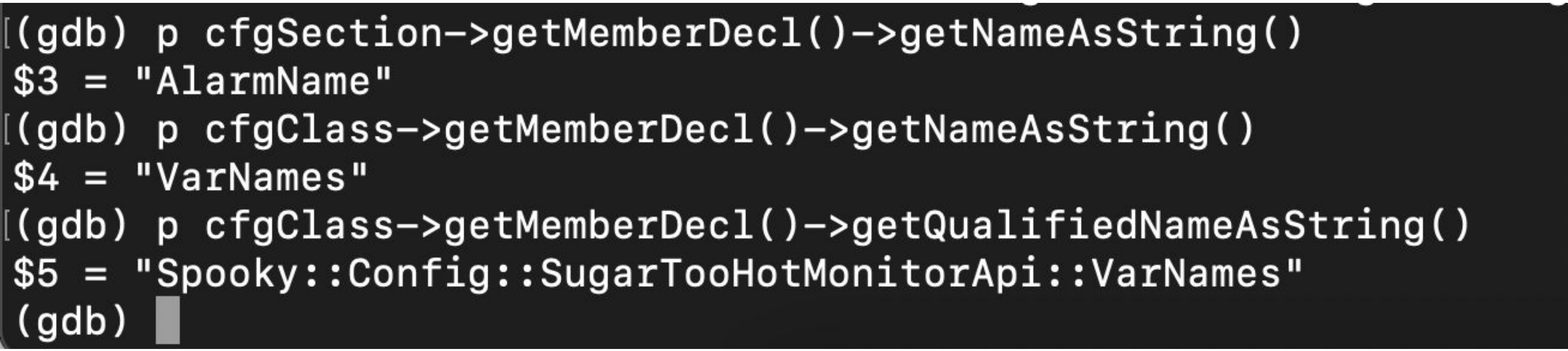
```
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitorFromConfig.cpp:9:54: note:
"cfgClass" binds here
            Spooky::Factory::AlarmClient alarmClient{cfgApi_.VarNames.AlarmName};
                                                     ^~~~~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitorFromConfig.cpp:9:54: note:
"cfgSection" binds here
            Spooky::Factory::AlarmClient alarmClient{cfgApi_.VarNames.AlarmName};
                                                     ^~~~~~~~~~~~~~~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitorFromConfig.cpp:9:13: note:
"inst" binds here
            Spooky::Factory::AlarmClient alarmClient{cfgApi_.VarNames.AlarmName};
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitorFromConfig.cpp:9:13: note:
"root" binds here
            Spooky::Factory::AlarmClient alarmClient{cfgApi_.VarNames.AlarmName};
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 match.
clang-query>
```

Config as input



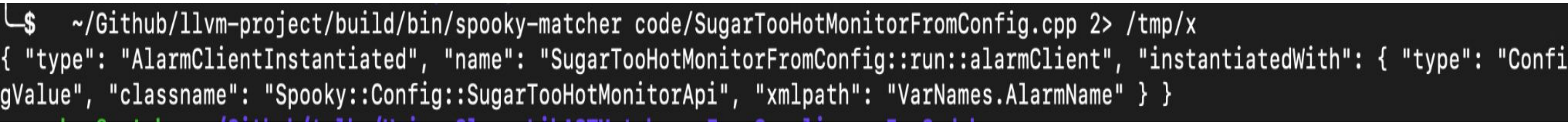
```
(gdb) p cfgSection->getMemberDecl()->getNameAsString()
$3 = "AlarmName"
(gdb) p cfgClass->getMemberDecl()->getNameAsString()
$4 = "VarNames"
(gdb) p cfgClass->getMemberDecl()->getQualifiedNameAsString()
$5 = "Spooky::Config::SugarTooHotMonitorApi::VarNames"
(gdb) 
```

Config as input



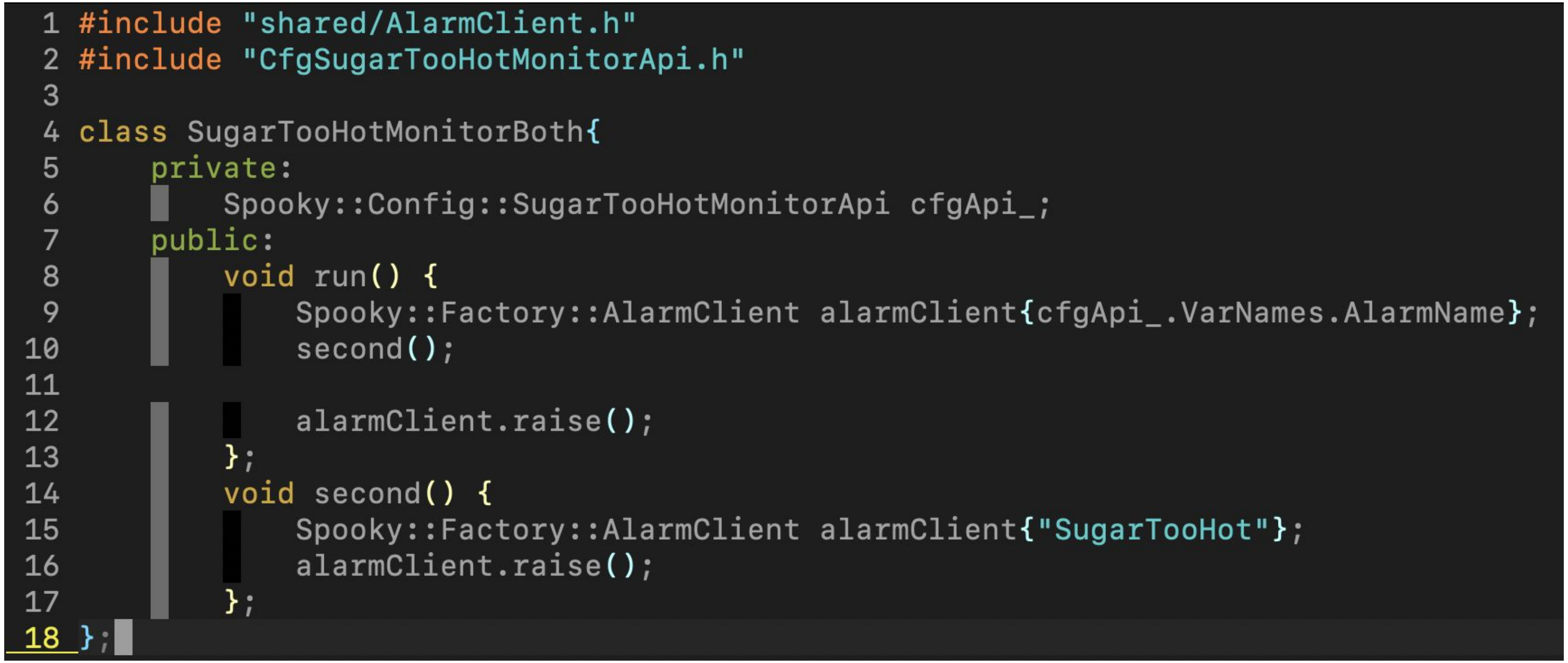
```
[(gdb) p cfgSection->getMemberDecl()->getNameAsString()
$3 = "AlarmName"
[(gdb) p cfgClass->getMemberDecl()->getNameAsString()
$4 = "VarNames"
[(gdb) p cfgClass->getMemberDecl()->getQualifiedNameAsString()
$5 = "Spooky::Config::SugarTooHotMonitorApi::VarNames"
(gdb) 
```

Config as input



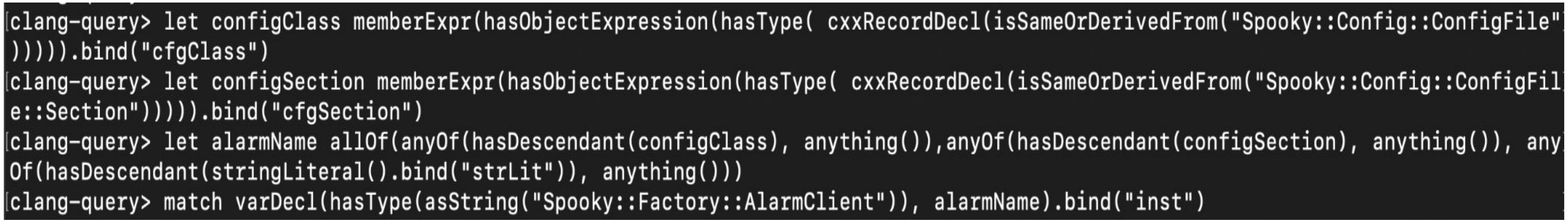
```
└$   ~/Github/llvm-project/build/bin/spooky-matcher code/SugarTooHotMonitorFromConfig.cpp 2> /tmp/x
{ "type": "AlarmClientInstantiated", "name": "SugarTooHotMonitorFromConfig::run::alarmClient", "instantiatedWith": { "type": "Confi
gValue", "classname": "Spooky::Config::SugarTooHotMonitorApi", "xmlpath": "VarNames.AlarmName" } }
```

Multiple versions in same file



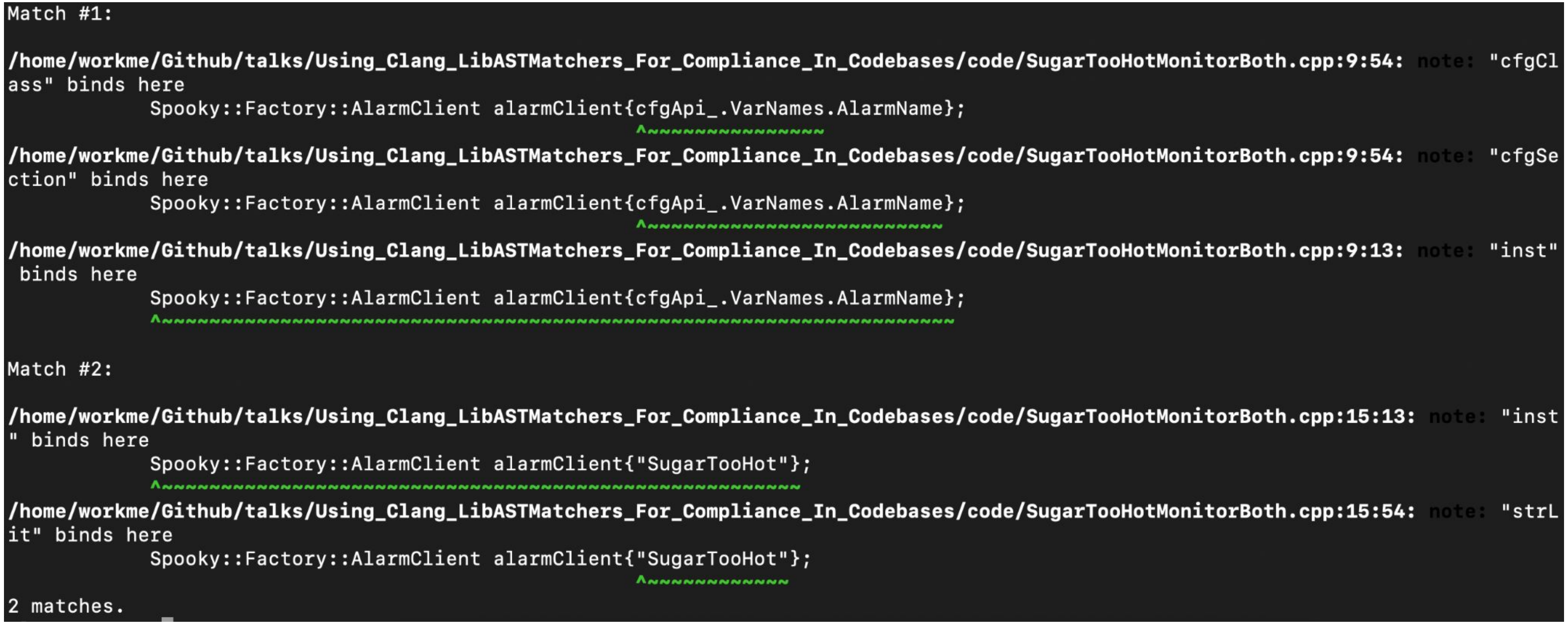
```
#include "shared/AlarmClient.h"
#include "CfgSugarTooHotMonitorApi.h"

class SugarTooHotMonitorBoth{
    private:
        Spooky::Config::SugarTooHotMonitorApi cfgApi_;
    public:
        void run() {
            Spooky::Factory::AlarmClient alarmClient{cfgApi_.VarNames.AlarmName};
            second();

            alarmClient.raise();
        };
        void second() {
            Spooky::Factory::AlarmClient alarmClient{"SugarTooHot"};
            alarmClient.raise();
        };
};
```

Multiple versions in same file



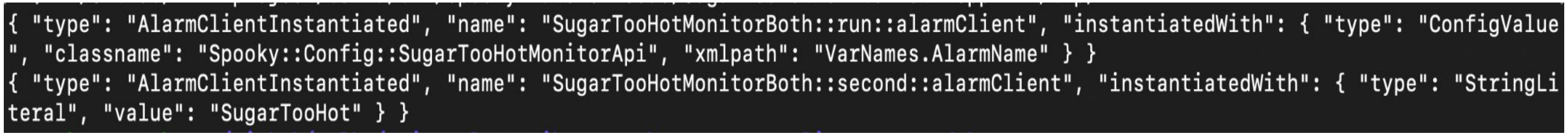
```
[clang-query> let configClass memberExpr(hasObjectExpression(hasType( cxxRecordDecl(isSameOrDerivedFrom("Spooky::Config::ConfigFile"
))))).bind("cfgClass")
[clang-query> let configSection memberExpr(hasObjectExpression(hasType( cxxRecordDecl(isSameOrDerivedFrom("Spooky::Config::ConfigFil
e::Section"))))).bind("cfgSection")
[clang-query> let alarmName allOf(anyOf(hasDescendant(configClass), anything()),anyOf(hasDescendant(configSection), anything()), any
Of(hasDescendant(stringLiteral().bind("strLit")), anything()))
[clang-query> match varDecl(hasType(asString("Spooky::Factory::AlarmClient")), alarmName).bind("inst")
```

Multiple versions in same file



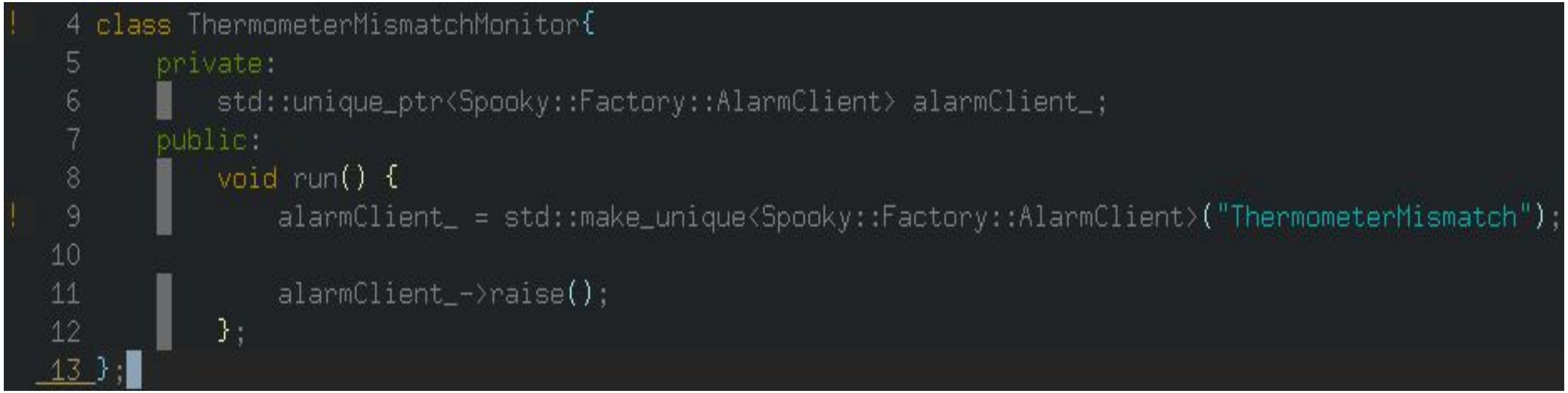
```
Match #1:

/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitorBoth.cpp:9:54: note: "cfgCl
ass" binds here
            Spooky::Factory::AlarmClient alarmClient{cfgApi_.VarNames.AlarmName};
                                                     ^~~~~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitorBoth.cpp:9:54: note: "cfgSe
ction" binds here
            Spooky::Factory::AlarmClient alarmClient{cfgApi_.VarNames.AlarmName};
                                                     ^~~~~~~~~~~~~~~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitorBoth.cpp:9:13: note: "inst"
 binds here
            Spooky::Factory::AlarmClient alarmClient{cfgApi_.VarNames.AlarmName};
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


Match #2:

/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitorBoth.cpp:15:13: note: "inst
" binds here
            Spooky::Factory::AlarmClient alarmClient{"SugarTooHot"};
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/SugarTooHotMonitorBoth.cpp:15:54: note: "strL
it" binds here
            Spooky::Factory::AlarmClient alarmClient{"SugarTooHot"};
                                                     ^~~~~~~~~~~~
2 matches.
```

Multiple versions in same file



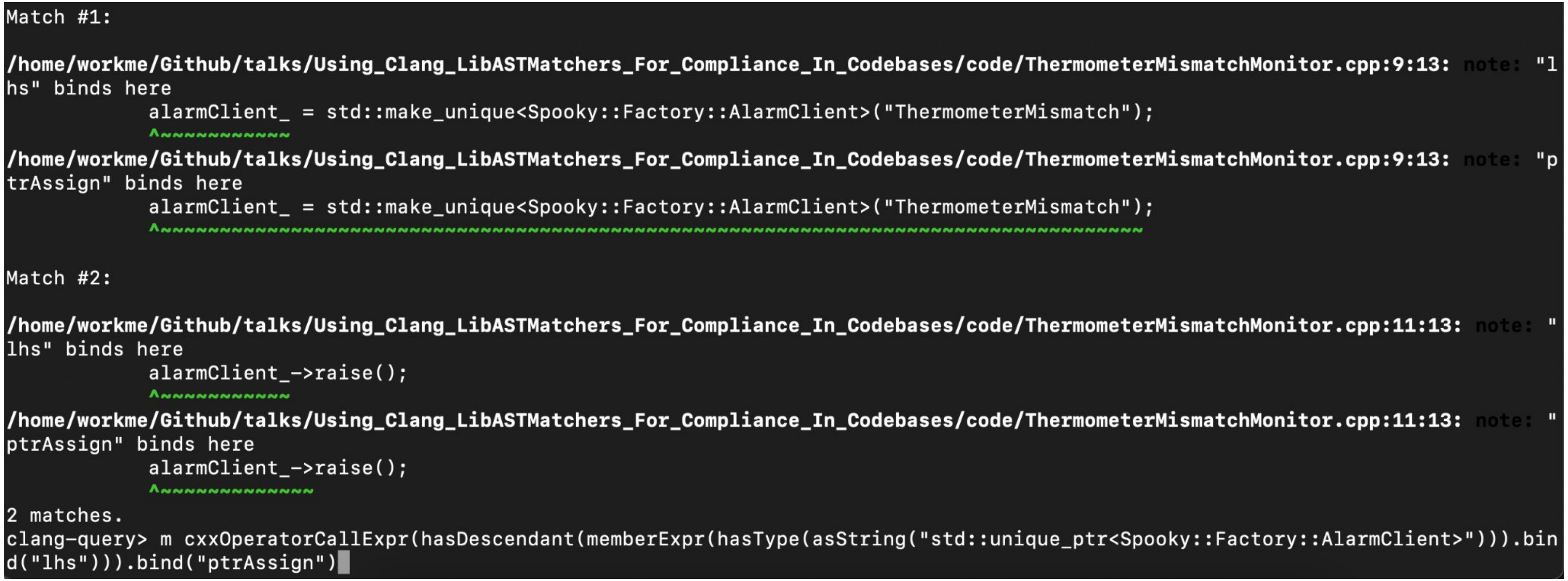
```
{ "type": "AlarmClientInstantiated", "name": "SugarTooHotMonitorBoth::run::alarmClient", "instantiatedWith": { "type": "ConfigValue
", "classname": "Spooky::Config::SugarTooHotMonitorApi", "xmlpath": "VarNames.AlarmName" } }
{ "type": "AlarmClientInstantiated", "name": "SugarTooHotMonitorBoth::second::alarmClient", "instantiatedWith": { "type": "StringLi
teral", "value": "SugarTooHot" } }
```

More Complicated



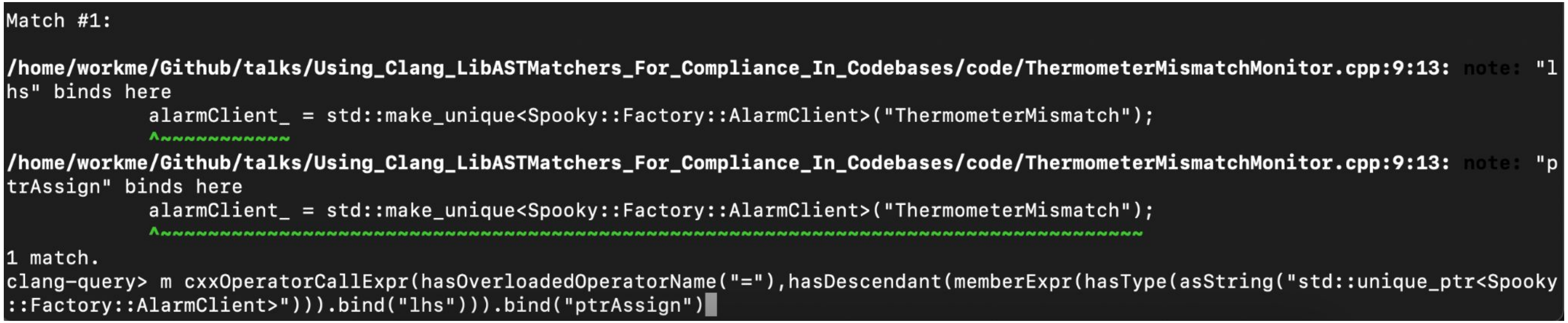
```cpp
class ThermometerMismatchMonitor{
    private:
        std::unique_ptr<Spooky::Factory::AlarmClient> alarmClient_;
    public:
        void run() {
            alarmClient_ = std::make_unique<Spooky::Factory::AlarmClient>("ThermometerMismatch");

            alarmClient_->raise();
        };
};
```

More Complicated



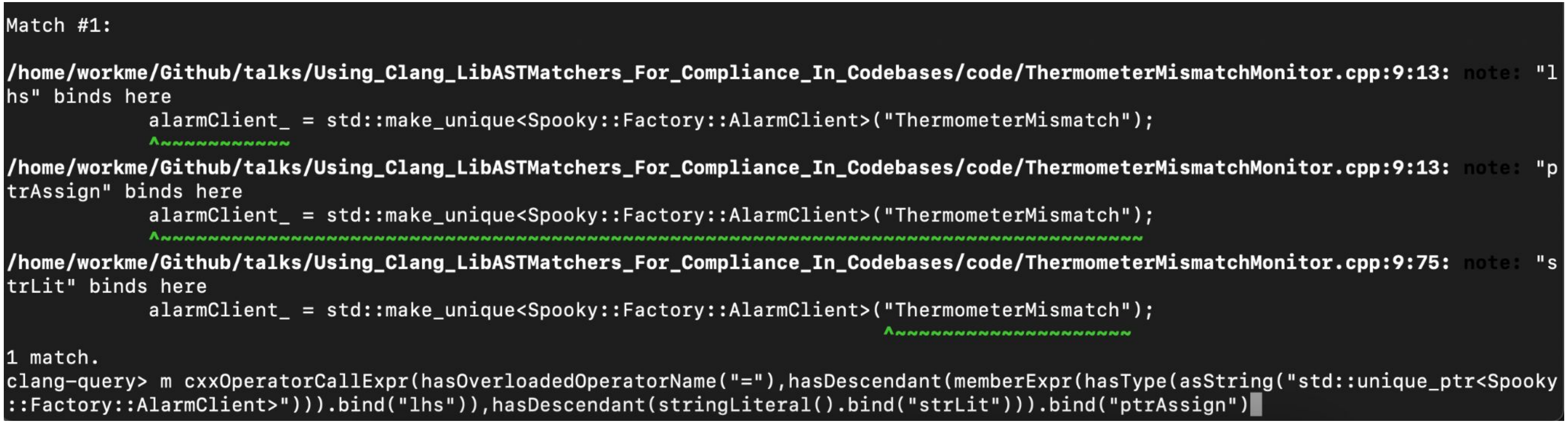
```
Match #1:

/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/ThermometerMismatchMonitor.cpp:9:13: note: "l
hs" binds here
            alarmClient_ = std::make_unique<Spooky::Factory::AlarmClient>("ThermometerMismatch");
            ^~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/ThermometerMismatchMonitor.cpp:9:13: note: "p
trAssign" binds here
            alarmClient_ = std::make_unique<Spooky::Factory::AlarmClient>("ThermometerMismatch");
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


Match #2:

/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/ThermometerMismatchMonitor.cpp:11:13: note: "
lhs" binds here
            alarmClient_->raise();
            ^~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/ThermometerMismatchMonitor.cpp:11:13: note: "
ptrAssign" binds here
            alarmClient_->raise();
            ^~~~~~~~~~~~~~
2 matches.
clang-query> m cxxOperatorCallExpr(hasDescendant(memberExpr(hasType(asString("std::unique_ptr<Spooky::Factory::AlarmClient>"))).bin
d("lhs"))).bind("ptrAssign")
```

More Complicated



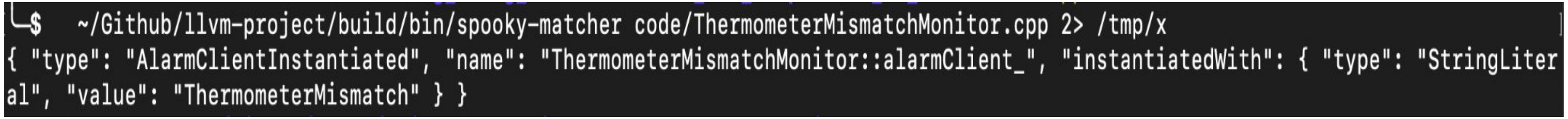
```
Match #1:

/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/ThermometerMismatchMonitor.cpp:9:13: note: "l
hs" binds here
            alarmClient_ = std::make_unique<Spooky::Factory::AlarmClient>("ThermometerMismatch");
            ^~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/ThermometerMismatchMonitor.cpp:9:13: note: "p
trAssign" binds here
            alarmClient_ = std::make_unique<Spooky::Factory::AlarmClient>("ThermometerMismatch");
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 match.
clang-query> m cxxOperatorCallExpr(hasOverloadedOperatorName("="),hasDescendant(memberExpr(hasType(asString("std::unique_ptr<Spooky
::Factory::AlarmClient>"))).bind("lhs"))).bind("ptrAssign")
```

More Complicated



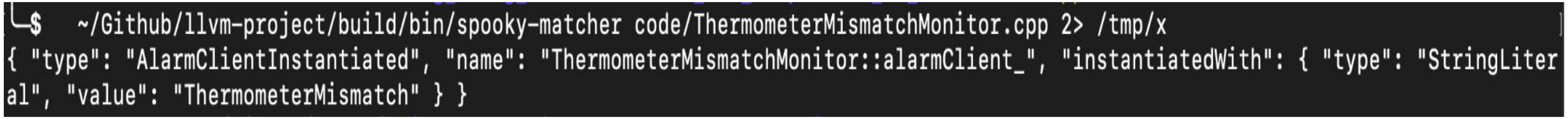
```
Match #1:

/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/ThermometerMismatchMonitor.cpp:9:13: note: "l
hs" binds here
            alarmClient_ = std::make_unique<Spooky::Factory::AlarmClient>("ThermometerMismatch");
            ^~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/ThermometerMismatchMonitor.cpp:9:13: note: "p
trAssign" binds here
            alarmClient_ = std::make_unique<Spooky::Factory::AlarmClient>("ThermometerMismatch");
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/ThermometerMismatchMonitor.cpp:9:75: note: "s
trLit" binds here
            alarmClient_ = std::make_unique<Spooky::Factory::AlarmClient>("ThermometerMismatch");
                                                                          ^~~~~~~~~~~~~~~~~~~~

1 match.
clang-query> m cxxOperatorCallExpr(hasOverloadedOperatorName("="),hasDescendant(memberExpr(hasType(asString("std::unique_ptr<Spooky
::Factory::AlarmClient>"))).bind("lhs")),hasDescendant(stringLiteral().bind("strLit"))).bind("ptrAssign")
```

More Complicated



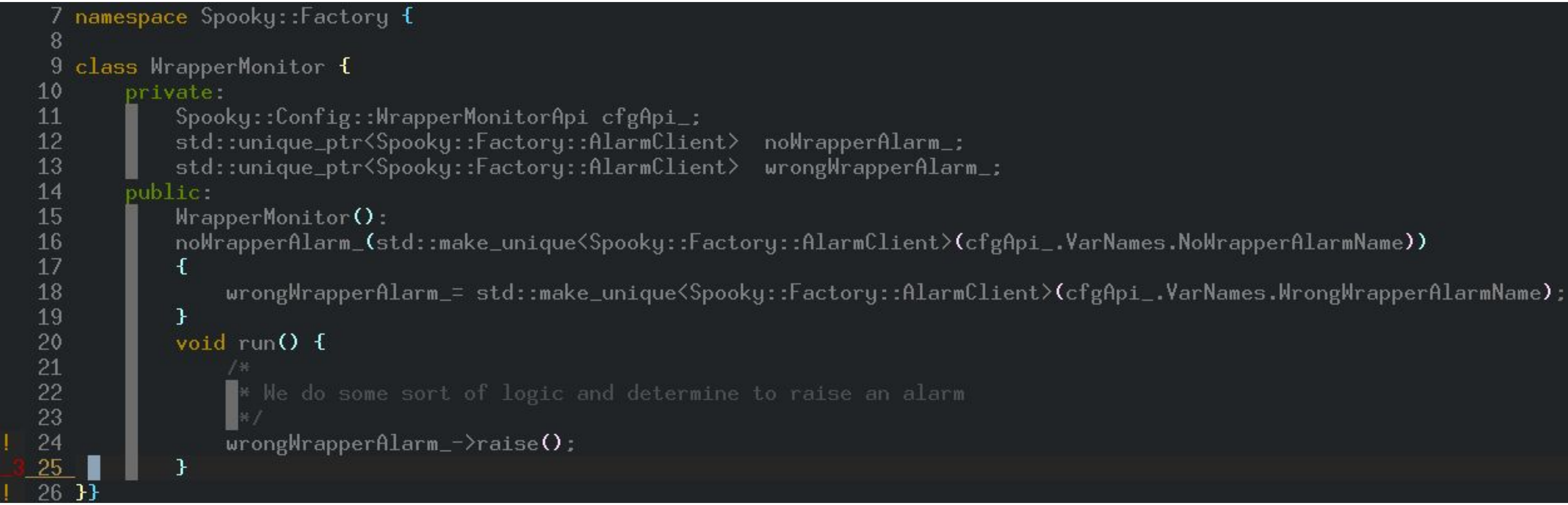
```
└─$   ~/Github/llvm-project/build/bin/spooky-matcher code/ThermometerMismatchMonitor.cpp 2> /tmp/x
{ "type": "AlarmClientInstantiated", "name": "ThermometerMismatchMonitor::alarmClient_", "instantiatedWith": { "type": "StringLiter
al", "value": "ThermometerMismatch" } }
```

More Complicated



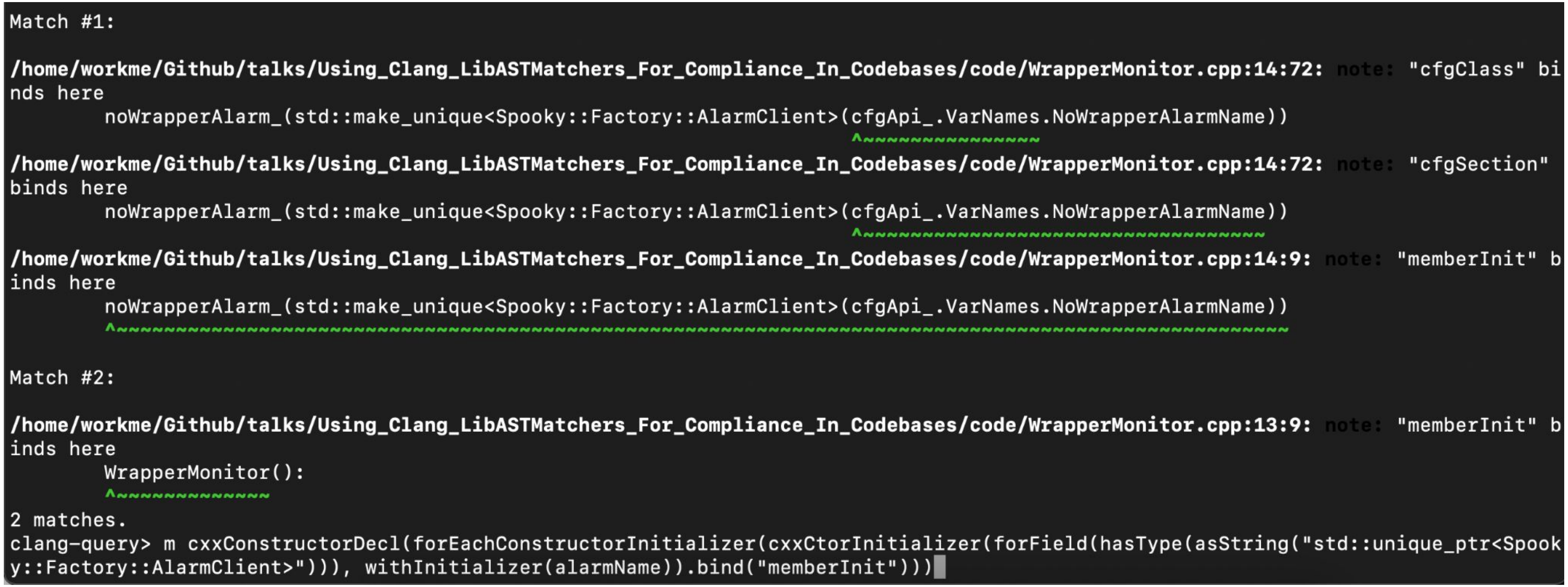
```
└─$   ~/Github/llvm-project/build/bin/spooky-matcher code/ThermometerMismatchMonitor.cpp 2> /tmp/x
{ "type": "AlarmClientInstantiated", "name": "ThermometerMismatchMonitor::alarmClient_", "instantiatedWith": { "type": "StringLiter
al", "value": "ThermometerMismatch" } }
```

More Complicated



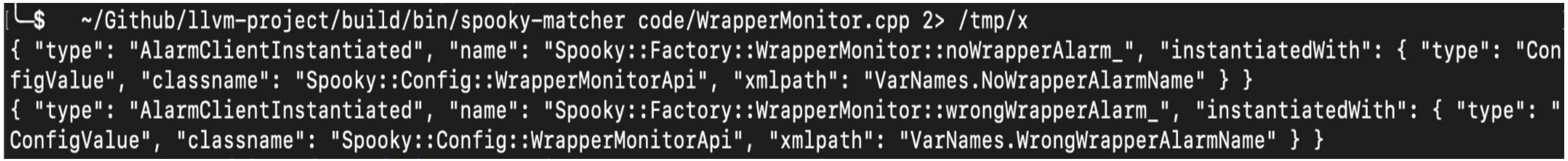
```
namespace Spooky::Factory {

class WrapperMonitor {
    private:
        Spooky::Config::WrapperMonitorApi cfgApi_;
        std::unique_ptr<Spooky::Factory::AlarmClient>  noWrapperAlarm_;
        std::unique_ptr<Spooky::Factory::AlarmClient>  wrongWrapperAlarm_;
    public:
        WrapperMonitor():
        noWrapperAlarm_(std::make_unique<Spooky::Factory::AlarmClient>(cfgApi_.VarNames.NoWrapperAlarmName))
        {
            wrongWrapperAlarm_= std::make_unique<Spooky::Factory::AlarmClient>(cfgApi_.VarNames.WrongWrapperAlarmName);
        }
        void run() {
            /*
             * We do some sort of logic and determine to raise an alarm
             */
            wrongWrapperAlarm_->raise();
        }
}}
```

More Complicated



```
Match #1:

/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/WrapperMonitor.cpp:14:72: note: "cfgClass" bi
nds here
        noWrapperAlarm_(std::make_unique<Spooky::Factory::AlarmClient>(cfgApi_.VarNames.NoWrapperAlarmName))
                                                                       ^~~~~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/WrapperMonitor.cpp:14:72: note: "cfgSection"
binds here
        noWrapperAlarm_(std::make_unique<Spooky::Factory::AlarmClient>(cfgApi_.VarNames.NoWrapperAlarmName))
                                                                       ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/WrapperMonitor.cpp:14:9: note: "memberInit" b
inds here
        noWrapperAlarm_(std::make_unique<Spooky::Factory::AlarmClient>(cfgApi_.VarNames.NoWrapperAlarmName))
        ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Match #2:

/home/workme/Github/talks/Using_Clang_LibASTMatchers_For_Compliance_In_Codebases/code/WrapperMonitor.cpp:13:9: note: "memberInit" b
inds here
        WrapperMonitor():
        ^~~~~~~~~~~~~
2 matches.
clang-query> m cxxConstructorDecl(forEachConstructorInitializer(cxxCtorInitializer(forField(hasType(asString("std::unique_ptr<Spook
y::Factory::AlarmClient>"))), withInitializer(alarmName)).bind("memberInit")))
```

More Complicated



```
└─$   ~/Github/llvm-project/build/bin/spooky-matcher code/WrapperMonitor.cpp 2> /tmp/x
{ "type": "AlarmClientInstantiated", "name": "Spooky::Factory::WrapperMonitor::noWrapperAlarm_", "instantiatedWith": { "type": "Con
figValue", "classname": "Spooky::Config::WrapperMonitorApi", "xmlpath": "VarNames.NoWrapperAlarmName" } }
{ "type": "AlarmClientInstantiated", "name": "Spooky::Factory::WrapperMonitor::wrongWrapperAlarm_", "instantiatedWith": { "type": "
ConfigValue", "classname": "Spooky::Config::WrapperMonitorApi", "xmlpath": "VarNames.WrongWrapperAlarmName" } }
```

Wrapping up the tool

- These outputs can now be consumed by another tool that can reconcile config files
- Some cases are too complicated or impossible. Had to have an override mechanism

Resources

- AST Matcher Tutorial
- LibASTMatchers reference
- Clang documentation
- Code used