
# CodeQL: Also a Powerful Binary Analysis Engine

Haiquan Zhang

rhettxie

@tencent security yunding lab

# Outline

- The story begins with static analysis
- CodeQL Introduction(Architecture and Core Engine)
- Exploring the implementation details of CodeQL through the lifecycle of QL statements
- How to extend CodeQL to support binary analysis
- A newly developed dedicated debugger
- Show time

# The Story Begins With Static Program Analysis

- Static program analysis is the art of reasoning about the behavior of computer programs without actually running them
- A static program analyzer is a program that reasons about the behavior of other programs
- It’s a hard work last for several decades. Resolved many issues, but still many problems remaining
- Precision and efficiency are the two big problems

# Many amazing technologies have been developed

- Type Inference and Abstract Interpretation
- Data Flow & Control Flow & Dependency Analysis
- Pointer Analysis
- Path Sensitive and Context Sensitive Analysis
- Constraint Solving and Symbolic Execution

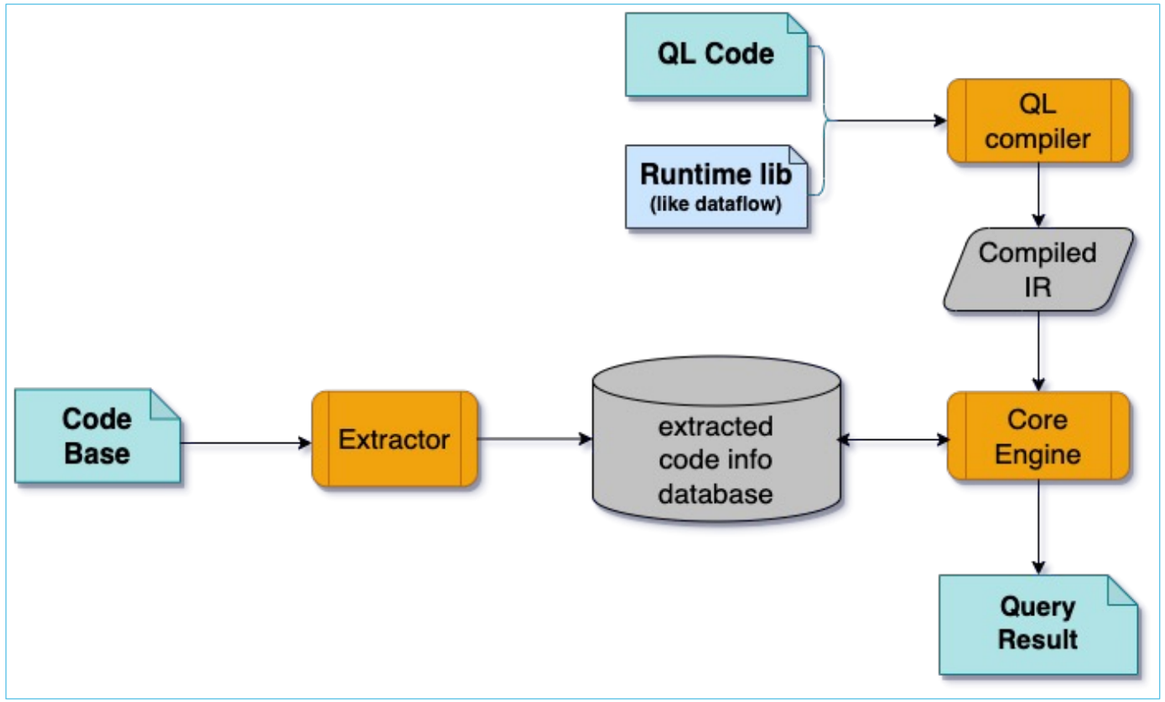
# Also Some Amazing Tools

| Name | Pay or not | Open source | Compiler depend | Tech stack |
|---|---|---|---|---|
| Coverity | commercial | source close | need compile | classic analysis tech |
| Fortify | commercial | source close | need compile | classic analysis tech |
| Symgrep | free use | source open | code scan | pattern matching |
| SVF | free use | source open | need compile<br>depends on LLVM | classic analysis tech<br>academic use |
| Clang Static Analyzer | free use | source open | need compile<br>clang only | AST travel<br>symbolic execution |

# What is CodeQL

- Founded in 2006, a research project from Oxford University, acquired by Github in 2019
- Partly open source , but the core engine is source closed
- Basically scan code, make database and run query logic, find patterns
- Make code analysis to code property query

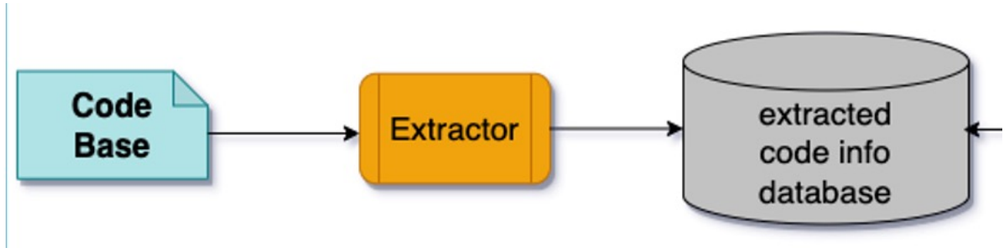
# Architecture of CodeQL




- The extractor can be regarded as language frontend，so it’s language depends
- Extractor scan code, make extra analysis and store code property to database
- Database store code information, can be shared and reused
- CodeQL introduced a query language, the ql is related with query logic, but not related with code that being analyzed,so it’s language agnostic
- CodeQL has developed a mature and comprehensive library that can perform various data flow analysis, such as the classic taint analysis
- The core engine can be regarded as a database evaluate engine

# Engine of CodeQL(extractor)

**./codeql database create  -l cpp -j 8 -s /data/codeql/qemu-6.1.0  /data/codeql/qemu.db**




```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 2531 root      20   0  102168  61372   9384 R  11.0   0.2   0:00.33 extractor
  759 root      20   0   95172  70208   3484 S   1.0   0.2   0:01.62 ninja
 1630 root      20   0  119508   6132   1820 S   0.7   0.0  40:48.43 watchdog.sh
  578 root      20   0  161360  12452  11004 S   0.3   0.0   0:00.02 sshd
 1508 root      20   0 1675380  18656   4860 S   0.3   0.1  88:18.28 containerd
```

```
[root@VM-113-41-centos /data/codeql/codeql]# ps -ef|grep extractor
root       867   856  0 12:43 pts/2    00:00:00 /data/codeql/codeql/cpp/tools/linux64/extractor --mimic /opt/rh/devtoolset-10/root/usr
/bin/gcc -Ilibqemu-sh4-softmmu.fa.p -I. -I.. -Itarget/sh4 -I../target/sh4 -I../capstone/include/capstone -Iqapi -Itrace -Iui -Iui/shad
er -I/usr/include/pixman-1 -I/usr/include/glib-2.0 -I/usr/lib64/glib-2.0/include -fdiagnostics-color=auto -pipe -Wall -Winvalid-pch -s
td=gnu11 -O2 -g -isystem /data/codeql/qemu-6.1.0/linux-headers -isystem linux-headers -iquote . -iquote /data/codeql/qemu-6.1.0 -iquot
e /data/codeql/qemu-6.1.0/include -iquote /data/codeql/qemu-6.1.0/disas/libvixl -iquote /data/codeql/qemu-6.1.0/tcg/i386 -pthread -U_F
ORTIFY_SOURCE -D_FORTIFY_SOURCE=2 -m64 -mcx16 -D_GNU_SOURCE -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -Wstrict-prototypes -Wredundant
-decls -Wundef -Wwrite-strings -Wmissing-prototypes -fno-strict-aliasing -fno-common -fwrapv -Wold-style-declaration -Wold-style-defin
ition -Wtype-limits -Wformat-security -Wformat-y2k -Winit-self -Wignored-qualifiers -Wempty-body -Wnested-externs -Wendif-labels -Wexp
ansion-to-defined -Wimplicit-fallthrough=2 -Wno-missing-include-dirs -Wno-shift-negative-value -Wno-psabi -fstack-protector-strong -fP
IC -isystem../linux-headers -isystemlinux-headers -DNEED_CPU_H -DCONFIG_TARGET="sh4-softmmu-config-target.h" -DCONFIG_DEVICES="sh4-sof
tmmu-config-devices.h" -MD -MQ libqemu-sh4-softmmu.fa.p/accel_tcg_translate-all.c.o -MF libqemu-sh4-softmmu.fa.p/accel_tcg_translate-a
ll.c.o.d -o libqemu-sh4-softmmu.fa.p/accel_tcg_translate-all.c.o -c ../accel/tcg/translate-all.c
```

- For compiled languages, the extractor and compiler work in parallel
- Monitor compiler works, if failed, then the process will abort
- Intercept compiler command parameters, and add more
- Extractor has the necessary information to compile a code file
- Extractor scan code, analysis and get code property

# Engine of CodeQL(extractor)

## How does extractor interrupt a compiler?

protected void executeSubcommand()

```
int result;
try {
    LogbackUtils.addAppender(streamOutAppender);   streamOutAppender: "com.semmle.util.logging.StreamAppender[null]"
    result = p.execute();   p: "/data/blackhat/codeql/tools/linux64/preload_tracer /data/blackhat/codeql/cpp/tools/autobuild.sh"
} finally {
    LogbackUtils.removeAppender(streamOutAppender);
}
```

1. Launch preload_tracer process while codeql starts to work

```
gs             0x0                    0
=> 0x410628 <set_preload_variable(char const*)+200>:    callq  0x40e820 <setenv@plt>
   0x41062d <set_preload_variable(char const*)+205>:    mov    %r13,%rsp
   0x410630 <set_preload_variable(char const*)+208>:    lea    -0x28(%rbp),%rsp
   0x410634 <set_preload_variable(char const*)+212>:    pop    %rbx
(gdb) x/s $rsi
0x7fffffffdbc0: "/data/codeql/codeql/tools/linux64/lib64trace.so"
(gdb)
```

3. GDB shows that codeql/tools/linux64/lib64trace.so injected to LD_PRELOAD

```
void __fastcall set_preload_variable(const char *runner)
{
  int v1; // eax
  int v2; // ebx
  unsigned int binary_kind; // r14d
  int v4; // eax
  char *const *v5; // r15
  char *v6; // rbx
  size_t libtrace_path_size; // r12
  size_t v8; // rax
  char *v9; // rbx
  uint8_t buf[1]; // [rsp+0h] [rbp-430h] BYREF
  ssize_t bufsize[6]; // [rsp+400h] [rbp-30h] BYREF

  bufsize[0] = 1024LL;
  v1 = open(runner, 0);
  if ( v1 == -1
    || (v2 = v1, binary_kind = get_binary_kind(v1, buf, bufsize), close(v2), binary_kind <= 5)
    && (v4 = 57, _bittest(&v4, binary_kind)) )
  {
    fwrite("Runner needs to be a full dynamic executable.\n", 0x2EuLL, 1uLL, stderr);
    exit(5);
  }
  v5 = semmle_env_current();
  v6 = getenv("LD_PRELOAD");
  libtrace_path_size = get_libtrace_path_size(v5);
  if ( v6 )
    v8 = strlen(v6);
  else
    v8 = 0LL;
  v9 = (char *)&buf[-((libtrace_path_size + v8 + 17) & 0xFFFFFFFFFFFFFFF0LL)];
  add_libtrace_to_preload(v9, libtrace_path_size + v8 + 2, v5, (a_binary_kind)binary_kind);
  setenv("LD_PRELOAD", v9, 1);
}
```

2. Disassembling shows that preload_tracer will inject something into LD_PRELOAD

# Engine of CodeQL(extractor)

## How does extractor compile code

```
**/configure:
  order compiler
  trace no
**/do-prebuild:
  order compiler
  trace no
**/as:
**/*-as:
  invoke ${config_dir}/extractor
  order compiler, extractor
  prepend --assembler
  prepend --codeql-assembler-executable
  prepend ${compiler}
**/collect2:
  invoke ${config_dir}/extractor
  order compiler, extractor
  prepend --linker
  prepend --semmle-linker-executable
  prepend ${compiler}
**/ld*:
**/*-ld*:
**/lld*:
  invoke ${config_dir}/extractor
  order compiler, extractor
  prepend --linker
  prepend --semmle-linker-executable
  prepend ${compiler}
**/armlink:
  invoke ${config_dir}/extractor
  order compiler, extractor
  prepend --arm-linker
  prepend --semmle-linker-executable
  prepend ${compiler}
**/*clang*:
**/*cc*:
**/*++*:
**/icpc:
  invoke ${config_dir}/extractor
  order compiler, extractor
  prepend --mimic
  prepend ${compiler}
```

```
{
  if ( !*v36 || !v36[1] || strcmp(v37, *v36) )
    goto LABEL_108;
  Logger::log(&v106, 0xFu, "Wrapped in runner.\n");
  v45 = semmle_detect_compiler(v36[1], &canonical_ptr);
  v46 = (char *const *)(v36 + 2);

  if ( !v45 )
    goto LABEL_108;
  v105.order.__begin_ = (std::vector<std::string>::pointer)"compiler";
  v105.order.__end_ = (std::vector<std::string>::pointer)canonical_ptr;
  v105.order.__end_cap_.__value_ = (std::string *)"compiler-architecture";
  v105.entries.__table_.__bucket_list_.__ptr_.__value_ = (std::__hash_node_
  v105.entries.__table_.__bucket_list_.__ptr_.__value_.__data_.__value_ =
  v105.entries.__table_.__p1_.__value_.__next_ = (std::__hash_node_base<st
  *(_OWORD *)&v105.entries.__table_.__p2_.__value_ = 0LL;
  if ( v45->run_compiler_only )
    goto LABEL_107;
  if ( !v45->run_compiler_first )
  {
    launch_extractor(v45, v46, (expandable_variable_t *)&v105);
    goto LABEL_107;
  }
```

```
Logger::Logger(&v48);
Logger::log(&v48, 0xAu, "launch_extractor(%s, argv, variables)\n", compiler->extractor);
Logger::log_strv(&v48, 0x14u, "argv", argv);
v5 = list_create();
add_element_expanding(v5, compiler->extractor, variables);
if ( compiler->num_args_prepend > 0 )
{
  v6 = 0LL;
  do
    add_element_expanding(v5, compiler->args_prepend[v6++], variables);
  while ( v6 < compiler->num_args_prepend );
}
v7 = *argv;
if ( *argv )
{
  v8 = (const char **)(argv + 1);
  do
  {
    v9 = strdup(v7);
    list_add_element(v5, v9);
    v7 = *v8++;
  }
  while ( v7 );
}
if ( compiler->num_args_append > 0 )
{
  v10 = 0LL;
  do
    add_element_expanding(v5, compiler->args_append[v10++], variables);
  while ( v10 < compiler->num_args_append );
}
list_add_element(v5, 0LL);
```

```
[root@VM-113-41-centos /data/codeql/codeql]# ps -ef|grep extractor
root       867   856  0 12:43 pts/2    00:00:00 /data/codeql/codeql/cpp/tools/linux64/extractor --mimic /opt/rh/devtoolset-10/root/usr
/bin/gcc -Ilibqemu-sh4-softmmu.fa.p -I. -I.. -Itarget/sh4 -I../target/sh4 -I../capstone/include/capstone -Iqapi -Itrace -Iui -Iui/shad
er -I/usr/include/pixman-1 -I/usr/include/glib-2.0 -I/usr/lib64/glib-2.0/include -fdiagnostics-color=auto -pipe -Wall -Winvalid-pch -s
td=gnu11 -O2 -g -isystem /data/codeql/qemu-6.1.0/linux-headers -isystem linux-headers -iquote . -iquote /data/codeql/qemu-6.1.0 -iquot
e /data/codeql/qemu-6.1.0/include -iquote /data/codeql/qemu-6.1.0/disas/libvixl -iquote /data/codeql/qemu-6.1.0/tcg/i386 -pthread -U_F
ORTIFY_SOURCE -D_FORTIFY_SOURCE=2 -m64 -mcx16 -D_GNU_SOURCE -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -Wstrict-prototypes -Wredundant
-decls -Wundef -Wwrite-strings -Wmissing-prototypes -fno-strict-aliasing -fno-common -fwrapv -Wold-style-declaration -Wold-style-defin
ition -Wtype-limits -Wformat-security -Wformat-y2k -Winit-self -Wignored-qualifiers -Wempty-body -Wnested-externs -Wendif-labels -Wexp
ansion-to-defined -Wimplicit-fallthrough=2 -Wno-missing-include-dirs -Wno-shift-negative-value -Wno-psabi -fstack-protector-strong -fP
IC -isystem../linux-headers -isystemlinux-headers -DNEED_CPU_H -DCONFIG_TARGET="sh4-softmmu-config-target.h" -DCONFIG_DEVICES="sh4-sof
tmmu-config-devices.h" -MD -MQ libqemu-sh4-softmmu.fa.p/accel_tcg_translate-all.c.o -MF libqemu-sh4-softmmu.fa.p/accel_tcg_translate-a
ll.c.o.d -o libqemu-sh4-softmmu.fa.p/accel_tcg_translate-all.c.o -c ../accel/tcg/translate-all.c
```

- lib64trace.so injected to every process
- detect whether the host process is a compiler process
- If so, add parameters from compiler-tracing.spec and launch an extractor process

# Engine of CodeQL(database)

## In fact , codeql generate a trap file firstly then convert the trap to the final database

```
[root@VM-113-41-centos /data/blackhat/code/test_c]# ll -h
total 20K
lrwxrwxrwx 1 root root    1 Jun 27 20:15 _lgtm_detected_source_root -> .
-rw-r--r-- 1  501 games  33 Jun 27 20:20 Makefile
-rwxr-xr-x 1 root root  11K Jul  9 14:37 test
-rw-r--r-- 1  501 games 875 Mar 23 11:30 test.c
[root@VM-113-41-centos /data/blackhat/code/test_c]#
```

```
[root@VM-113-41-centos /data/codeql/codeql/cpp]# ll -h
total 436K
-rw-r--r-- 1 root root  327 Feb 25  2022 codeql-extractor.yml
-rw-r--r-- 1 root root  661 Feb 25  2022 COPYRIGHT
drwxr-xr-x 5 root root 4.0K Feb 25  2022 downgrades
-rw-r--r-- 1 root root  10K Feb 25  2022 LICENSE
drwxr-xr-x 2 root root 4.0K Feb 25  2022 qltest
-rw-r--r-- 1 root root  50K Feb 25  2022 semmlecode.cpp.dbscheme
-rw-r--r-- 1 root root 351K Feb 25  2022 semmlecode.cpp.dbscheme.stats
drwxr-xr-x 6 root root 4.0K Jul  9 12:59 tools
[root@VM-113-41-centos /data/codeql/codeql/cpp]#
```

dbscheme is a guidance document for the content extraction, extractor extracts information according to that file

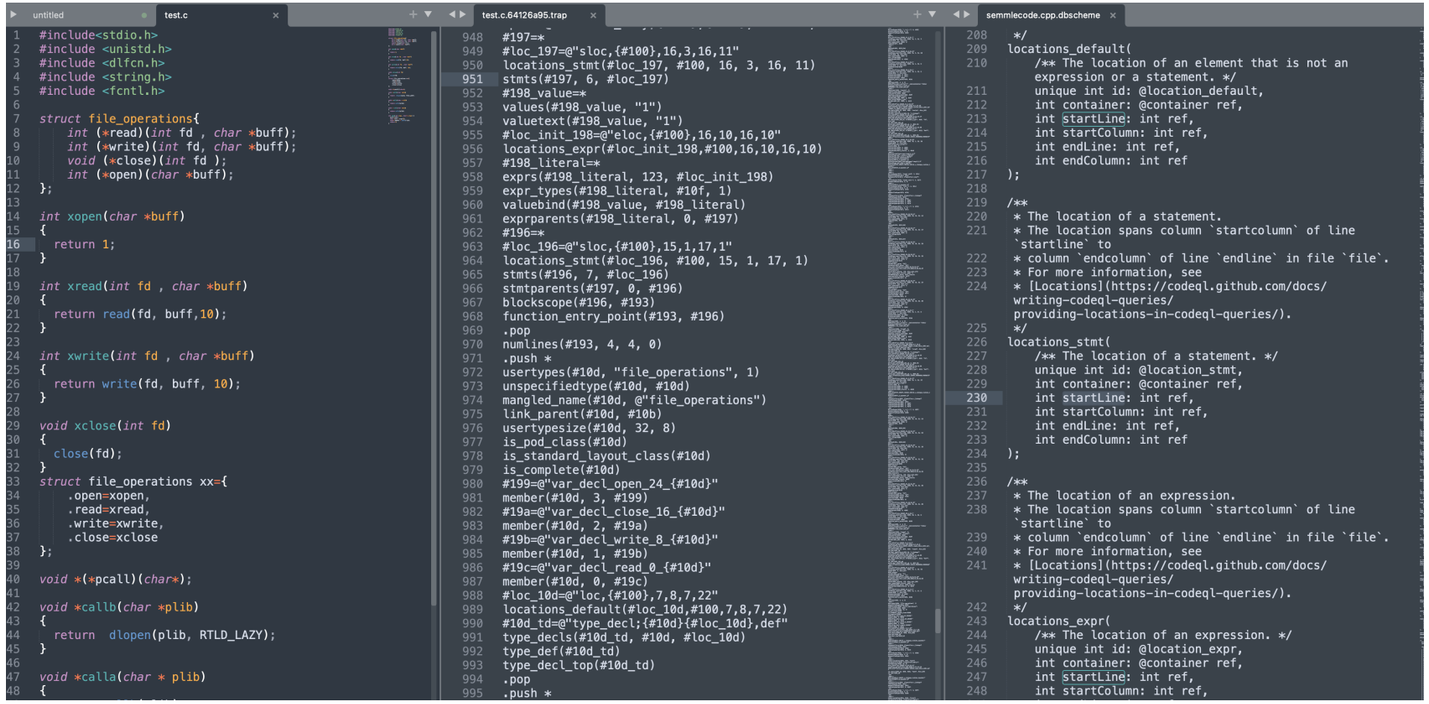
test.c.64126a95.trap

```
1  // Current directory: /data/blackhat/code/test_c
2  // Command: /data/blackhat/codeql/cpp/tools/linux64/extractor
   --mimic /usr/libexec/gcc/x86_64-redhat-linux/4.8.2/cc1 -quiet
   test.c -quiet -dumpbase test.c -mtune=generic -march=x86-64
   -auxbase test -g -o /tmp/ccB7Bw87.s
3  // Processed command line: /data/blackhat/codeql/cpp/tools/
   linux64/extractor --trapfolder /data/blackhat/codedb/trap/cpp
   --src_archive /data/blackhat/codedb/src --mimic_config /data/
   blackhat/codedb/working/compiler_mimic_cache/3c10c62e246b
   --object_filename /tmp/ccB7Bw87.s -w --error_limit 1000
```

**Trap file located at**
/data/blackhat/codedb/trap/cpp/tarballs/data/blackhat/code/test_c/source/data/blackhat/code/test_c

# Engine of CodeQL(database)

## Let’s map these files together, and see an example




- Extractor scan code, extracts information described in dbscheme
- All the code information extracted out will be placed at trap files
- Trap is a text file， it’s not convenient for further process It will be converted to structed db file

# Engine of CodeQL(database)

## Let's explore the dbscheme file

semmlecode.cpp.dbscheme

```
346
347  /*
348    case @function.kind of
349      1 = normal
350    | 2 = constructor
351    | 3 = destructor
352    | 4 = conversion
353    | 5 = operator
354    | 6 = builtin     // GCC built-in functions, e.g.
       __builtin___memcpy_chk
355    ;
356  */
357  functions(
358      unique int id: @function,
359      string name: string ref,
360      int kind: int ref
361  );
362
```

**Take functions as an example**

- functions is the collection of all functions
- It's unique lable is @function, can be referenced by this name
- It has two field, name & kind， name is string and kind is an int
- kind 1 means normal function, 2 means constructor

```
stmts(
    unique int id: @stmt,
    int kind: int ref,
    int location: @location_stmt ref
);

case @stmt.kind of
    1 = @stmt_expr
|   2 = @stmt_if
|   3 = @stmt_while
|   4 = @stmt_goto
|   5 = @stmt_label
|   6 = @stmt_return
|   7 = @stmt_block
|   8 = @stmt_end_test_while // do { ... } while ( ... )
|   9 = @stmt_for
|  10 = @stmt_switch_case
|  11 = @stmt_switch
|  13 = @stmt_asm // "asm" statement or the body of an asm
function
|  15 = @stmt_try_block
|  16 = @stmt_microsoft_try // Microsoft
|  17 = @stmt_decl
|  18 = @stmt_set_vla_size // C99
|  19 = @stmt_vla_decl // C99
|  25 = @stmt_assigned_goto // GNU
|  26 = @stmt_empty
|  27 = @stmt_continue
|  28 = @stmt_break
|  29 = @stmt_range_based_for // C++11
// ...  30 @stmt_at_autoreleasepool_block deprecated
// ...  31 @stmt_objc_for_in deprecated
// ...  32 @stmt_at_synchronized deprecated
|  33 = @stmt_handler
// ...  34 @stmt_finally_end deprecated
|  35 = @stmt_constexpr_if
|  37 = @stmt_co_return
;
```

```
locations_stmt(
    /** The location of a
statement. */
    unique int id:
@location_stmt,
    int container: @container
ref,
    int startLine: int ref,
    int startColumn: int ref,
    int endLine: int ref,
    int endColumn: int ref
);
```

Stmts is complicated

Is has recursion type

# Engine of CodeQL(database)

## Let's explore the trap file

test.c.64126a95.trap

```
927   .push *
928   functions(#193, "xopen", 1)
929   mangled_name(#193, @"xopen")
930   link_parent(#193, #10b)
931   function_return_type(#193, #10f)
932   #193_194=@"{#193}_0_par"
933   params(#193_194, #193, 0, #113)
```

simple example

In trap file
xopen is a function, named "xopen", and it kind is 1(normal)

```
7   struct file_operations{
8       int (*read)(int fd , char *buff);
9       int (*write)(int fd, char *buff);
10      void (*close)(int fd );
11      int (*open)(char *buff);
12  };
13
14  int xopen(char *buff)
15  {
16    return 1;
17  }
```

test.c.64126a95.trap

```
948   #197=*
949   #loc_197=@"sloc,{#100},16,3,16,11"
950   locations_stmt(#loc_197, #100, 16, 3, 16, 11)
951   stmts(#197, 6, #loc_197)
952   #198_value=*
```

complicated example

It's a statement
Kind 6 means a stmt_return
Location is at  startLine 16, startColumn 3
Endline is 16 and endColumn is 11

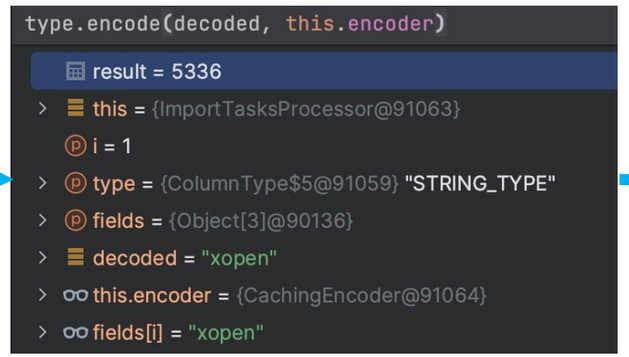
```
14  int xopen(char *buff)
15  {
16    return 1;
17  }
18
19  int xread(int fd , char *buff)
20  {
```

# Engine of CodeQL(database)

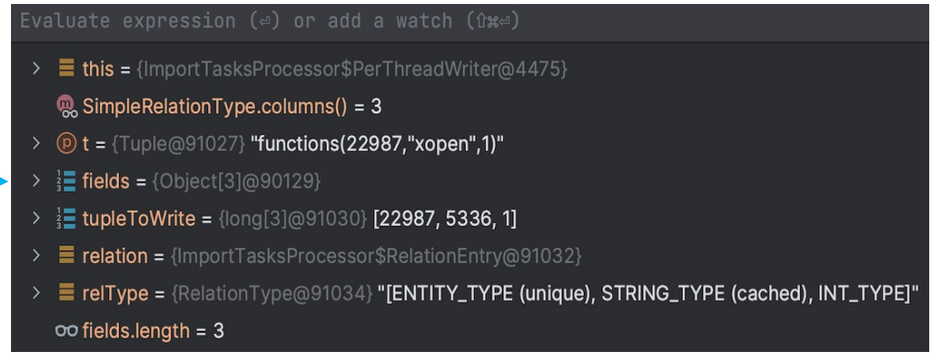
## trap to database

```
numlines(#180, 4, 4, 0)
#193=@"fun_decl_xopen(?)" .implementation
.push *
functions(#193, "xopen", 1)
mangled_name(#193, @"xopen")
link_parent(#193, #10b)
function_return_type(#193, #10f)
#193_194=@"{#193}_0_par"
params(#193_194, #193, 0, #113)
.pop
```

xopen function in trap file




xopen string encoded to 5336




a function declaration converted to an int tuple and write to db file

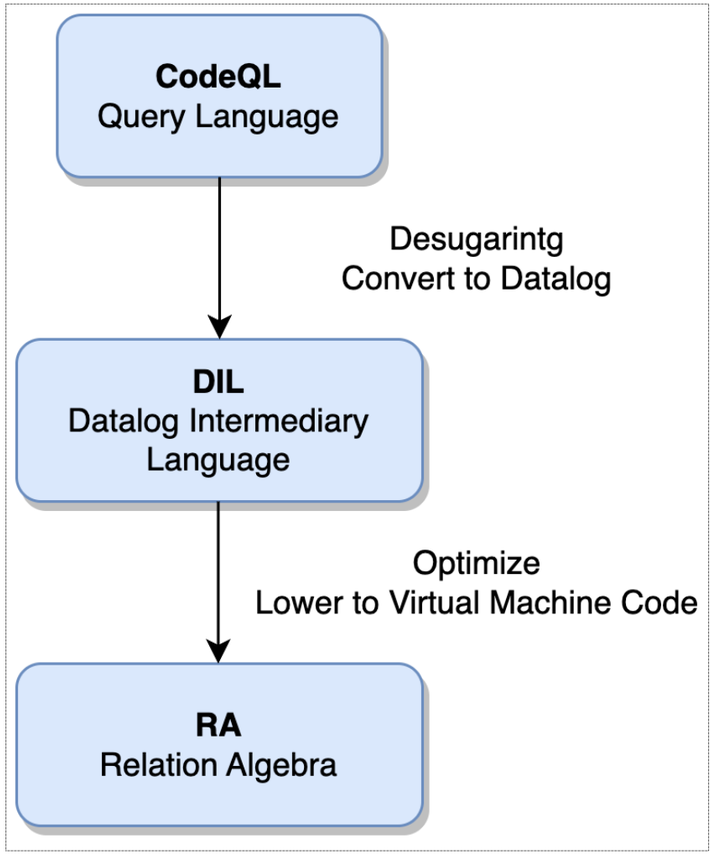
```
[root@VM-113-41-centos /data/blackhat/file]# hexdump -C functions.rel | tail
00000ff0  00 00 53 ca 00 00 13 30  00 00 00 01 00 00 53 cd  |..S....0......S.|
00001000  00 00 13 31 00 00 00 01  00 00 53 d1 00 00 13 32  |...1......S....2|
00001010  00 00 00 01 00 00 58 a1  00 00 14 cf 00 00 00 01  |......X.........|
00001020  00 00 58 bd 00 00 14 d2  00 00 00 01 00 00 59 05  |..X...........Y.|
00001030  00 00 14 d3 00 00 00 01  00 00 59 1e 00 00 14 d4  |..........Y.....|
00001040  00 00 00 01 00 00 59 4b  00 00 14 d5 00 00 00 01  |......YK........|
00001050  00 00 59 69 00 00 14 d6  00 00 00 01 00 00 59 9a  |..Yi..........Y.|
00001060  00 00 14 d7 00 00 00 01  00 00 59 cb 00 00 14 d8  |..........Y.....|
00001070  00 00 00 01                                       |....|
00001074
```

0x59cb=22987  0x14d8=5336

the rel file is db file, tuples data are stored simply and directly in database file

- The int values in tuple can be regarded as index to the actual resources
- It's a complex and tedious process, and we will not go into further details here

# The Query Language




- Declarative Logic Programming
- With class and OOP support
- Target language independent
- Built-in with many libraries, such as the most common data flow analysis, taint analysis

# QL to DIL

## The first lowing phase

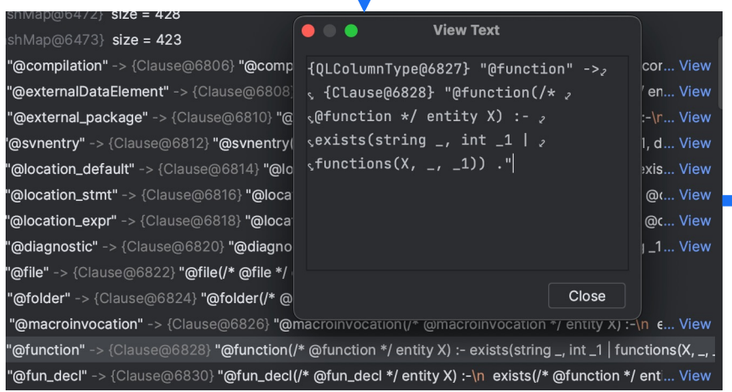
untitled | debug.ql

```
import cpp

from FunctionCall fc, Function f

where
    fc.getTarget() = f and
    f.getName() = "dlopen"
select fc.getEnclosingFunction().getName(), f
```

```
public static DBSchemeToDIL run(QlPrimitives primitives, Collection<DBSchemeBinding> dbSchemeBindings) {
    TypeHierarchyBuilder typeHeirarchyBuilder = DBSchemeToTypes.run(primitives, dbSchemeBindings);
    QLSourceMap sourceMap = new QLSourceMap();
    RelationBuilder var10000 = new RelationBuilder();
```




debug.dil

```
8545
8546  #select#ff(cached string select, /* Function::Function */ cached entity f) :-
8547    exists(/* Function::Function */ cached entity call_result#4 |
8548      exists(/* Call::FunctionCall */ cached entity fc |
8549        exists(cached string arg1, cached dontcare int _ |
8550          arg1 = "dlopen", functions(f, arg1, _)
8551        ),
8552        Call#39248e3c::FunctionCall::getTarget#0#dispred#fb(fc, f),
8553        Enclosing#c50c5fbf::exprEnclosingElement#1(fc, call_result#4)
8554      ),
8555      exists(cached dontcare int _ | functions(call_result#4, select, _))
8556    )
8557  .
```

- It's a lowing process
- High level user friendly language to machine friendly language
- From where select, high level logic to child query operation
- Table query and bool calculus

# DIL to RA

## The second lowing phase

debug.dil

```
8545
8546  #select#ff(cached string select, /* Function::Function */ cached entity f) :-
8547    exists(/* Function::Function */ cached entity call_result#4 |
8548      exists(/* Call::FunctionCall */ cached entity fc |
8549        exists(cached string arg1, cached dontcare int _ |
8550          arg1 = "dlopen", functions(f, arg1, _)
8551        ),
8552        Call#39248e3c::FunctionCall::getTarget#0#dispred#fb(fc, f),
8553        Enclosing#c50c5fbf::exprEnclosingElement#1(fc, call_result#4)
8554      ),
8555      exists(cached dontcare int _ | functions(call_result#4, select, _))
8556    )
8557  .
8558
```

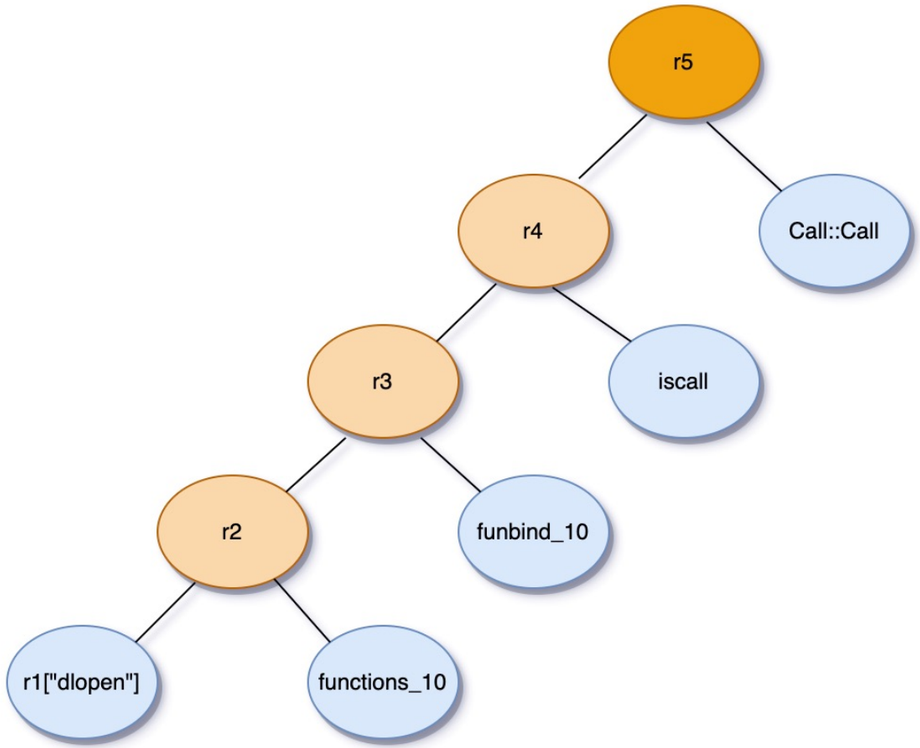
debug.ra

```
6113
6114  EVALUATE NONRECURSIVE RELATION:
6115    #select#ff(cached string select, entity f) :-
6116      SENTINEL Call#39248e3c::FunctionCall::getTarget#0#dispred#fb_10#join_rhs
6117      SENTINEL Enclosing#c50c5fbf::exprEnclosingElement#1#ff@staged_ext
6118      {1} r1 = CONSTANT(unique string)["dlopen"]
6119      {1} r2 = JOIN r1 WITH functions_10#join_rhs ON FIRST 1 OUTPUT Rhs.1 'f'
6120      {2} r3 = JOIN r2 WITH Call#39248e3c::FunctionCall::getTarget#0#dispred#fb_10#join_rhs ON
         FIRST 1 OUTPUT Rhs.1, Lhs.0 'f'
6121      {2} r4 = JOIN r3 WITH Enclosing#c50c5fbf::exprEnclosingElement#1#ff@staged_ext ON FIRST 1
         OUTPUT Rhs.1, Lhs.1 'f'
6122      {2} r5 = JOIN r4 WITH functions ON FIRST 1 OUTPUT Rhs.1 'select', Lhs.1 'f'
6123      return r5
```

- Continue lowing
- Data logic to relation algebra operations
- RA operations are classic JOIN UNION and BOOL calculus
- R1 R2, they are not registers

# Evaluate Engine

## RA code and query tree




```
no usages
public RelationWriter evaluate(RelationK
    this.metadata.getMetrics().counter(E
    CatastrophicError.throwIfNull(enclos
    this.context.cancelToken.checkCancel
    this.metadata.getProfiler().beginPip
    Exceptions.assertion(this.allocatedW
    Pipeline p = this.pipeline(expr);
    RelationWriter dest = provider.creat

    QueryableRelationWriter var35;
    try {
        this.metadata.getProfiler().log(
        if (this.metadata.getProfiler().
            p = new TupleLoggingPipeline
        }
```

```
{1} r1 = CONSTANT(unique string)["dlopen"]
{1} r2 = JOIN r1 WITH functions_10#join_rhs ON FIRST 1 OUTPUT Rhs.1
{2} r3 = JOIN r2 WITH funbind_10#join_rhs ON FIRST 1 OUTPUT Rhs.1, Lhs.0
{2} r4 = JOIN r3 WITH iscall ON FIRST 1 OUTPUT Lhs.0, Lhs.1
{2} r5 = JOIN r4 WITH Call#39248e3c::Call#b ON FIRST 1 OUTPUT Lhs.0, Lhs.1
return r5
```

- RA expr is a query tree
- A query tree is a set of logic related together, and can not be separated
- The engine will consume a tree at a time
- Evaluate from bottom to up
- Support recursion evaluation

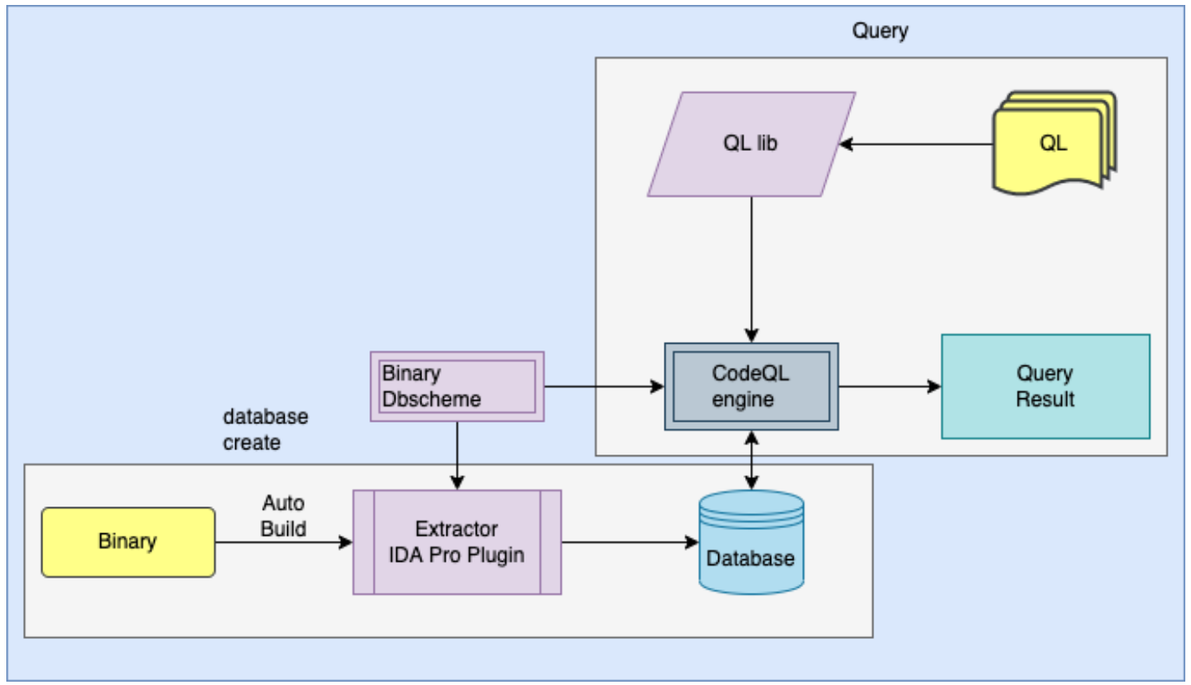
# Summary

- CodeQL has several language frontend, as extractor
- Extractor extracts code information, do analysis, and store information to database
- CodeQL implemented a datalog query language to query code property
- CodeQL stores code property as database, and regard code analysis as database query

# The Advance of CodeQL

- The architecture design is very elegant, highly modular, and separates the front-end and back-end
- CodeQL is a faithful implementer of database technology, convert code analysis to data query
- Well designed query language, focus on code pattern or semantic matching. One query language to handle many other code language
- Thanks to a unified query language, the same logic for different analysis tasks can be reused on a large scale

# Extending CodeQL to binary




Architecture

- new dbscheme for binary
- new extractor for binary
- new QL library for binary

# Design of binary dbshceme

- Store the file, architecture, string, and function information of the binary.
- Store the information of instructions, registers, and basic blocks inside the function.
- Store the use and def information of registers inside the function.
- Store the information of global variables and pointers.
- Store memory layout information.(address ,section etc)

# New Dbscheme For Binary

## Basic information

asm.dbscheme ●

ql > lib > asm.dbscheme

```
1   sourceLocationPrefix(varchar(900) prefix : string ref);
2
3   files(unique int id:@file,string name:string ref);
4
5   export_function(unique int id:@export_func,string name:string ref);
6
7   import_function(unique int id:@import_func,string name:string ref,
8     string module:string ref);
9
10  string_literals(unique int id:@strlit, string ident:string ref);
11
12  functions(unique int id:@func,string name:string ref);
```

## Control flow information

```
25  objects(unique int  id:@object,int kind:int ref,
26    string name :string ref);
27  case @object.kind of
28  0 = @lable;
```

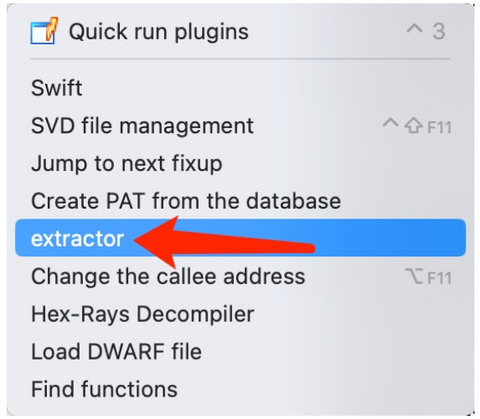
## Instruction and register information

```
16  exprs(unique int id:@expr,int kind : int ref,int func:@func ref);
17
18  registers(unique int  id:@regs,int kind:int ref);
19
20  defs(int ident:@ident ref,int reg:@regs ref);
21
22  uses(int ident:@ident ref,int reg:@regs ref);
```

```
30  case @regs.kind of
31    0= @r_ax
32  | 1= @r_cx
33  | 2= @r_dx
34  | 3= @r_bx
35  | 4= @r_sp
36  | 5= @r_bp
37  | 6= @r_si
38  | 7= @r_di
39  | 8= @r_r8
```

```
205  case @expr.kind of
206    0 = @ident
207    | 1= @nn_aaa
208    | 2= @nn_aad
209    | 3= @nn_aam
210    | 4= @nn_aas
211    | 5= @nn_adc
212    | 6= @nn_add
213    | 7= @nn_and
214    | 8= @nn_arpl
215    | 9= @nn_bound
```

# New Extractor For Binary




The extractor needs to extract information according to the tables defined by the dbscheme. The main challenge here is how to efficiently save the relationships between the tables.

## Extracting all strings from binary

```
5307 #3632=*
5308 string_literals(#3632,"Read lines from the standard input into an indexed array variable.
5309
5310      Read lines from the standard input into the indexed array variable ARRAY, or
5311      from file descriptor FD if the -u option is supplied.  The variable MAPFILE
5312      is the default ARRAY.
5313
5314      Options:
5315        -n count  Copy at most COUNT lines.  If COUNT is 0, all lines are copied.
5316        -O origin Begin assigning to ARRAY at index ORIGIN.  The default index is 0.
5317        -s count  Discard the first COUNT lines read.
5318        -t                Remove a trailing newline from each line read.
5319        -u fd             Read lines from file descriptor FD instead of the standard input.
5320        -C callback       Evaluate CALLBACK each time QUANTUM lines are read.
5321        -c quantum        Specify the number of lines read between each call to CALLBACK.
5322
5323      Arguments:
5324        ARRAY             Array variable name to use for file data.
5325
5326      If -C is supplied without -c, the default quantum is 5000.  When
5327      CALLBACK is evaluated, it is supplied the index of the next array
5328      element to be assigned and the line to be assigned to that element
5329      as additional arguments.
5330
5331      If not supplied with an explicit origin, mapfile will clear ARRAY before
5332      assigning to it.
5333
5334      Exit Status:
5335      Returns success unless an invalid option is given or ARRAY is readonly or
5336      not an indexed array.")
```

## Extracting imported function information from binary

```
8527 #4598=*
8528 import_function(#4598,"__fprintf_chk@@GLIBC_2.3.4",".dynsym")
8529 #4599=*
8530 import_function(#4599,"getrlimit@@GLIBC_2.2.5",".dynsym")
8531 #4600=*
8532 import_function(#4600,"setuid@@GLIBC_2.2.5",".dynsym")
8533 #4601=*
8534 import_function(#4601,"getaddrinfo@@GLIBC_2.2.5",".dynsym")
```

```
8577 #4624=*
8578 #4625=*
8579 exprs(#4624,0,#4623)
8580 exprs(#4625,209,#4623)
8581 registers(#4624,4)
8582 defs(#4624,#4624)
8583 #4626=*
8584 #4627=*
8585 exprs(#4626,0,#4623)
8586 exprs(#4627,122,#4623)
8587 registers(#4626,0)
8588 defs(#4626,#4626)
8589 #4628=*
8590 #4629=*
8591 exprs(#4628,0,#4623)
8592 exprs(#4629,210,#4623)
8593 registers(#4628,0)
8594 defs(#4628,#4628)
8595 registers(#4628,0)
8596 uses(#4628,#4628)
8597 #4630=*
8598 #4631=*
8599 exprs(#4630,0,#4623)
8600 exprs(#4631,85,#4623)
8601 #4632=*
8602 #4633=*
8603 exprs(#4632,0,#4623)
8604 exprs(#4633,16,#4623)
8605 #4634=*
8606 #4635=*
8607 exprs(#4634,0,#4623)
8608 exprs(#4635,6,#4623)
8609 registers(#4634,4)
8610 defs(#4634,#4634)
```

Extracting usage information of instructions and registers from binary.

# Configure a new workflow

➢ The above has completed the creation of the two most important key components for creating a database.Now, they need to be integrated into the CodeQL data creation workflow. Below is how we supplemented the configuration files needed for the workflow through dynamic debugging.

## Register a new language option

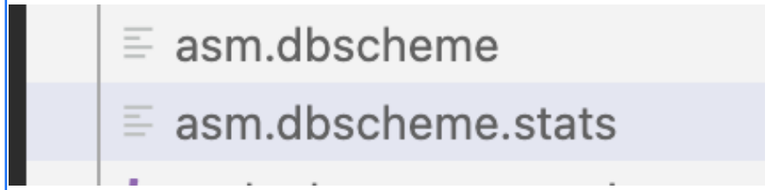
codeql-extractor.yml

codeql-extractor-asm > codeql-extractor.yml

```
1   name: "asm"
2   display_name: "ASM"
3   version: 0.1.0
4   column_kind: "utf8"
5   file_types:
6     - name: asm
7       display_name: ASM
8       extensions:
9         - .asm
10  legacy_qltest_extraction: true
```

## Configure the compiler


## Generate asm.dbscheme.stats file



# Autobuild.sh For Binary

## database create

```
zhangsan@HUNTAZHANG-MB0 codeql-asm % codeql-debug database create -l asm  -s build/stats/demo/   --search-path . build/stats/demo_db1

Listening for transport dt_socket at address: 5005
Initializing database at /Users/zhangsan/codeql-asm/build/stats/demo_db1.
Running build command: [/Users/zhangsan/codeql-asm/codeql-extractor-asm/tools/autobuild.sh]
```

codeql-extractor-asm > tools > $ autobuild.sh

```
1   #!/usr/bin/env bash
2   set
3   export LGTM_SRC=`pwd`
4   mkdir ${CODEQL_EXTRACTOR_ASM_WIP_DATABASE}/src
5   mkdir ${CODEQL_EXTRACTOR_ASM_WIP_DATABASE}/trap
6   mkdir ${CODEQL_EXTRACTOR_ASM_WIP_DATABASE}/trap/asm
7   gzip *.trap
8   cp *.gz  ${CODEQL_EXTRACTOR_ASM_WIP_DATABASE}/trap/asm/
9
10
```

```
zhangsan@HUNTAZHANG-MB0 demo_db1 % ls
baseline-info.json  codeql-database.yml db-asm          log         src         trap        working
zhangsan@HUNTAZHANG-MB0 demo_db1 %
```

Different from the data workflow for creating source code, we separate the extractor process, and autobuild.sh is just for packaging the trap files and copying them to a specific directory in the database.

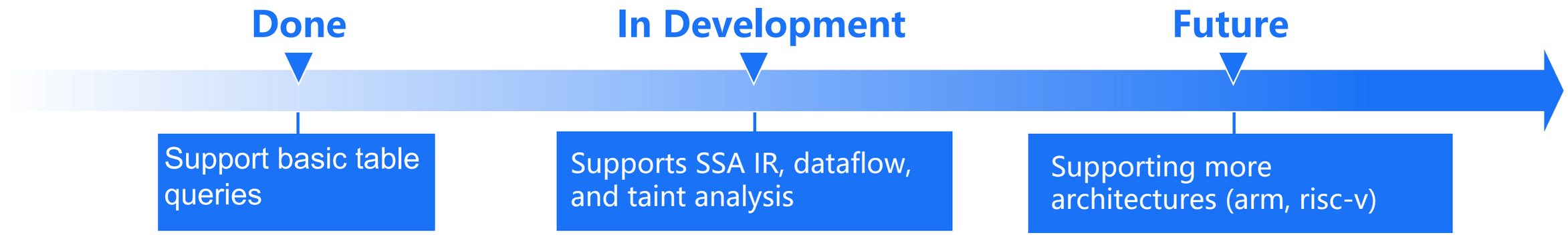
# QL library For Binary

➢ The QL library can help us use the QL language, and with this flexible language, we can accomplish even more powerful tasks.

- Export functions
- Functions
- Registers, instructions
- Import functions
- Strings

ql > lib > asm.qll > {} asm

```
1  import tencent.asm.ExportFunctions
2  import tencent.asm.Functions
3  import tencent.asm.Files
4  import tencent.asm.Expr
5  import tencent.asm.Reg
6  import tencent.asm.ImportFunctions
7  import tencent.asm.StringLit
```



# Simple example

export_func.ql

ql > src > examples > export_func.ql > { } export_func

```
1  import asm
2
3  from ExportFunc func
4  select func.getFuncName()
5
```

CodeQL Query Results

« 1 / 10 »  export_func.ql on demo_db - finished in 0 seconds (1985 results)
[7/6/2023, 4:35:59 PM]  Open export_func.ql

#select  1985 results

| # | [0] |
|---|---|
| 1 | unsetenv |
| 2 | putenv |
| 3 | stdout |
| 4 | stderr |
| 5 | group_member |
| 6 | __bss_start |
| 7 | the_replace_len |
| 8 | sigwinch_sighandler |
| 9 | sh_modcase |
| 10 | parse_shellopts |
| 11 | parse_string |
| 12 | protected_mode |
| 13 | rl_show_char |
| 14 | rl_byte_oriented |
| 15 | __libc_csu_fini |
| 16 | rl_filename_quoting_desired |
| 17 | search_for_command |
| 18 | rl_eof_char |

# Complex example

expr.ql M

ql > src > examples > expr.ql > { } expr > dumpStr

```
1   import asm
2
    Quick Evaluation: hasGNU
3   predicate  hasGNU(StrLit s ){
4       s.getStringLiteral().matches("%GNU GPL%")
5   }
6
    Quick Evaluation: dumpStr
7   query   string  dumpStr(StrLit s) {
8       result = s.getStringLiteral()
9   }
10
11  query string dumpFunc(FunctionType f)
12  {
13      result = f.getFuncName()
14  }
15
16  from StrLit s,ImportFunc importFunc,FunctionType func,CallInsn callinsn
17  where hasGNU(s) and importFunc.getFuncName().matches("exec%")
18  and func.getFuncName()= "save_input_line_state"
19  and callinsn.getInsnFunction()=func
20  select s.getStringLiteral(), func.getFuncName(),callinsn,importFunc.getFuncName()
```

CodeQL Query Results

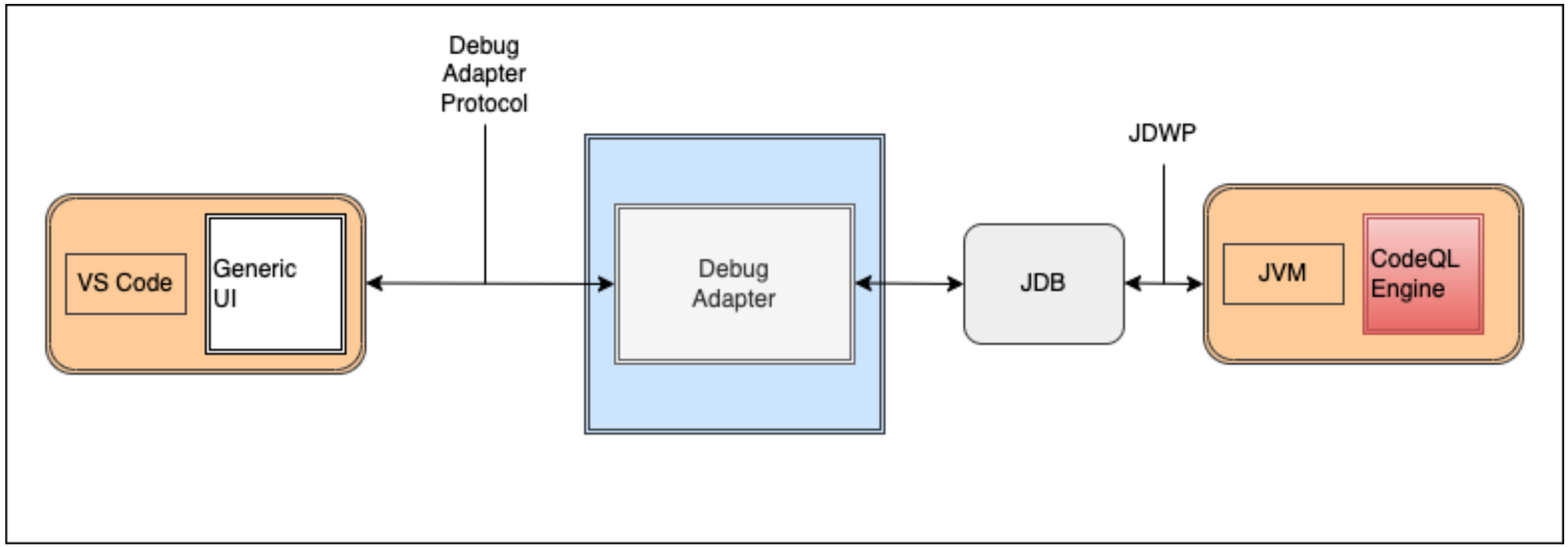
« 1 / 1 » expr.ql on demo_db - finished in 0 seconds (1 results) [7/6/2023, 4:40:46 PM] Open expr.ql

#select 1 result

| # | [0] | [1] | callinsn | [3] |
|---|---|---|---|---|
| 1 | License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html> | save_input_line_state | call xmalloc | execve@@GLIBC_2.2.5 |

# QL language Debugger At RA-level

- The current version of the debugger supports breakpoints, execution, single-stepping, viewing relations, and tracing.




Architecture

# com.semmle.inmemory.eval.Evaluate.evaluate

- It is the starting point for RA's interpreted execution, where we can use CountingPrinter.print(expr) to print out the decompiled text of RA

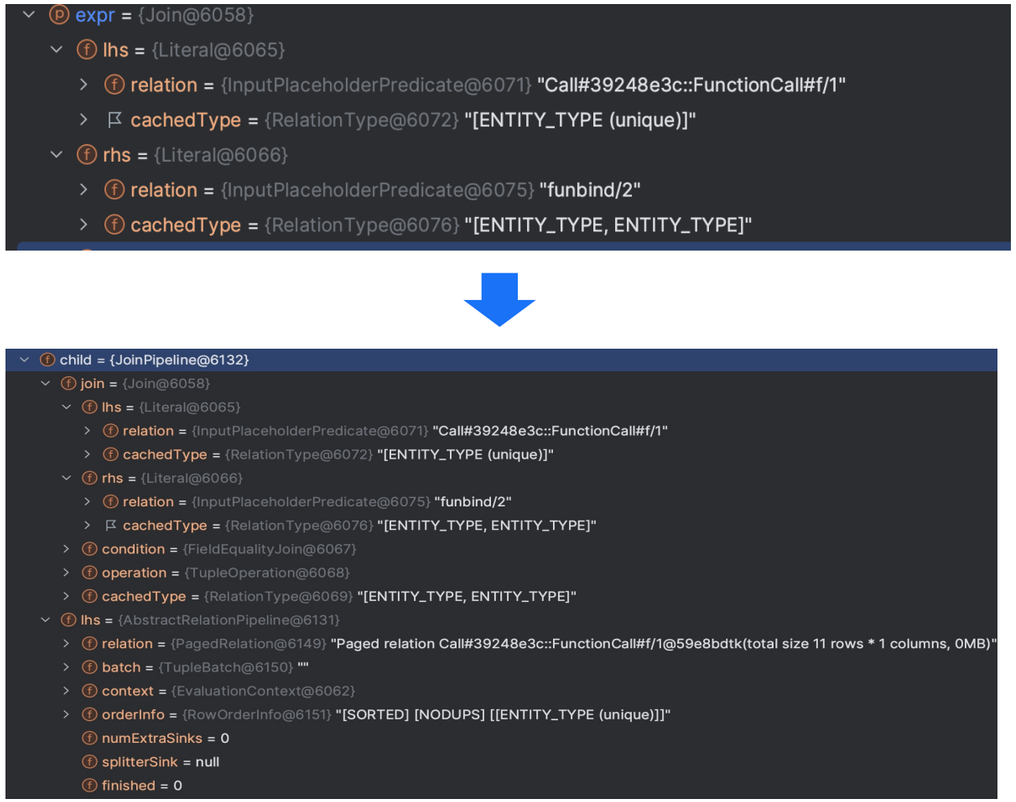
```
80    public RelationWriter evaluate(RelationKey enclosing, int iteration, RelationWriterProvider provider, RA expr, List<String> pipelineColumnNames)
      this.metadata.getMetrics().counter(EvalCountStat.PIPELINE_EVALUATE.metricName).inc();
82    CatastrophicError.throwIfNull(enclosing);
```

- Decompiled text

```
CountingPrinter.print(expr)
> CountingPrinter.print(expr) = "\n    {2} r1 = JOIN Call#39248e3c::FunctionCall#f WITH funbind ON FIRST 1 OUTPUT Lhs.0, Rhs.1\n    return r1\n" ... View
```

# RA To Pipeline

## 13 types of operation pipelines




| RA operation | Pipeline operation |
|---|---|
| ApplyTupleOperation | TupleOperationPipeline |
| StreamDedup | StreamDedupPipeline |
| SelectionByTest | SelectionByTestPipeline |
| Literal | AbstractRelationPipeline |
| EmptySet | EmptySetPipeline |
| Union | UnionPipeline |
| InvokeHigherOrderRelation | InvokeHigherOrderRelationPipeline |
| Join | JoinPipeline |
| ... | ... |

# Pipeline Run

- Each type of pipeline will eventually call its own runinternal method to perform operations such as reading and writing tables. Tables in the database are saved and used in memory in the form of page relations.

```
19 @    public AbstractRelationPipeline(EvaluationContex
20          super(context, new RowOrderInfo(relation, ty
21          this.relation = relation;
22          this.batch = TupleBatch.create(relation.type
23      }
24
25      public AbstractRelation getRelation() { return t
28
        no usages
29      protected void runInternal(BatchSink sink) {
            this.relation.batchedMap(this.batch, sink);
31          this.batch.flush(sink);
32      }
```

Running pipeline

```
((PagedRelation) this.relation).getPage(0).data
  result = {IntArray[1]@6174}
    0 = {JavaIntArray@6177}
      data = {int[11]@6178} [11958, 14539, 14802, 14968, 15155, 22711, 22736, 22764, 22798, 22833, 22882]
      handedOutBy = {ArrayAllocator@6179}
  CountingPrinter.print(expr) = Cannot find local variable 'expr'
  this = {AbstractRelationPipeline@6131}
  sink = {JoinEvaluator@6170}
  this.batch = {TupleBatch@6150} ""
  this.relation = {PagedRelation@6149} "Paged relation Call#39248e3c::FunctionCall#f/1@59e8bdtk(total size 11 rows * 1 columns, 0MB)"
```

Get data corresponding to the page relation

# com.semmle.inmemory.caching.RelationManager.addRelation

- CodeQL will save the page relations mentioned earlier here. Each table also has a different storage type, and we have implemented reading for each type in the debugger.

```
try {
    PagedRelationInfo info = paged.getDictionary();   paged: "Paged relation Call#39248e3c::FunctionCall::getTarget#ff/2@6f2e224u(total size 11 rows * 2 columns, 0MB)"
    Optional<PredicateMetadata> prev = this.cache.getPredicate(key);
    if (prev.isPresent()) {...}

    this.cache.putPredicate(key, info.getStorageKey());
    this.cache.putRelation(info);
    rows = paged.size();
    result = info.getStorageKey();
```

Entry point

```
paged.getDictionary()
result = {PagedRelationInfo@6185}
  tupleCount = 11
  pageCount = 1
  pageKeys = {String[1]@6189} ["17e43eclu302o1d..."]
  maxKeyLength = 29
  pageStarts = {long[1][]@6190}
  pageEnds = {long[1][]@6191}
  type = {RelationType@6192} "[ENTITY_TYPE, ENTITY_TYPE]"
  sorted = true
  storageKey = null
```

Page information

```
paged.getPage(0).data
result = {IntArray[2]@6199}
  0 = {JavaIntArray@6206}
    data = {int[11]@6208} [11958, 14539, 14802, 14968, 15155, 22711, 22736, 22764, 22798, 22833, 22882]
    handedOutBy = {ArrayAllocator@6179}
  1 = {JavaIntArray@6207}
    data = {int[11]@6209} [11954, 14535, 14535, 14535, 15130, 22709, 22734, 19345, 16640, 16543, 16574]
    handedOutBy = {ArrayAllocator@6179}
```

Page relation data

The storage format of a relation

- BaseGeneralIntArrayRelation
- BaseIntArrayRelation
- BorrowedIntArrayRelation
- CBTreeRelation
- IntArrayRelation

# codeql-debug Agent

- We discovered a hidden performance tuning parameter 'semmle.profiler.verbosity' during dynamic debugging, which can save the engine's execution state to a file. This way, we don't have to implement a trace function through jdb, we just need to add this parameter to the startup parameters.

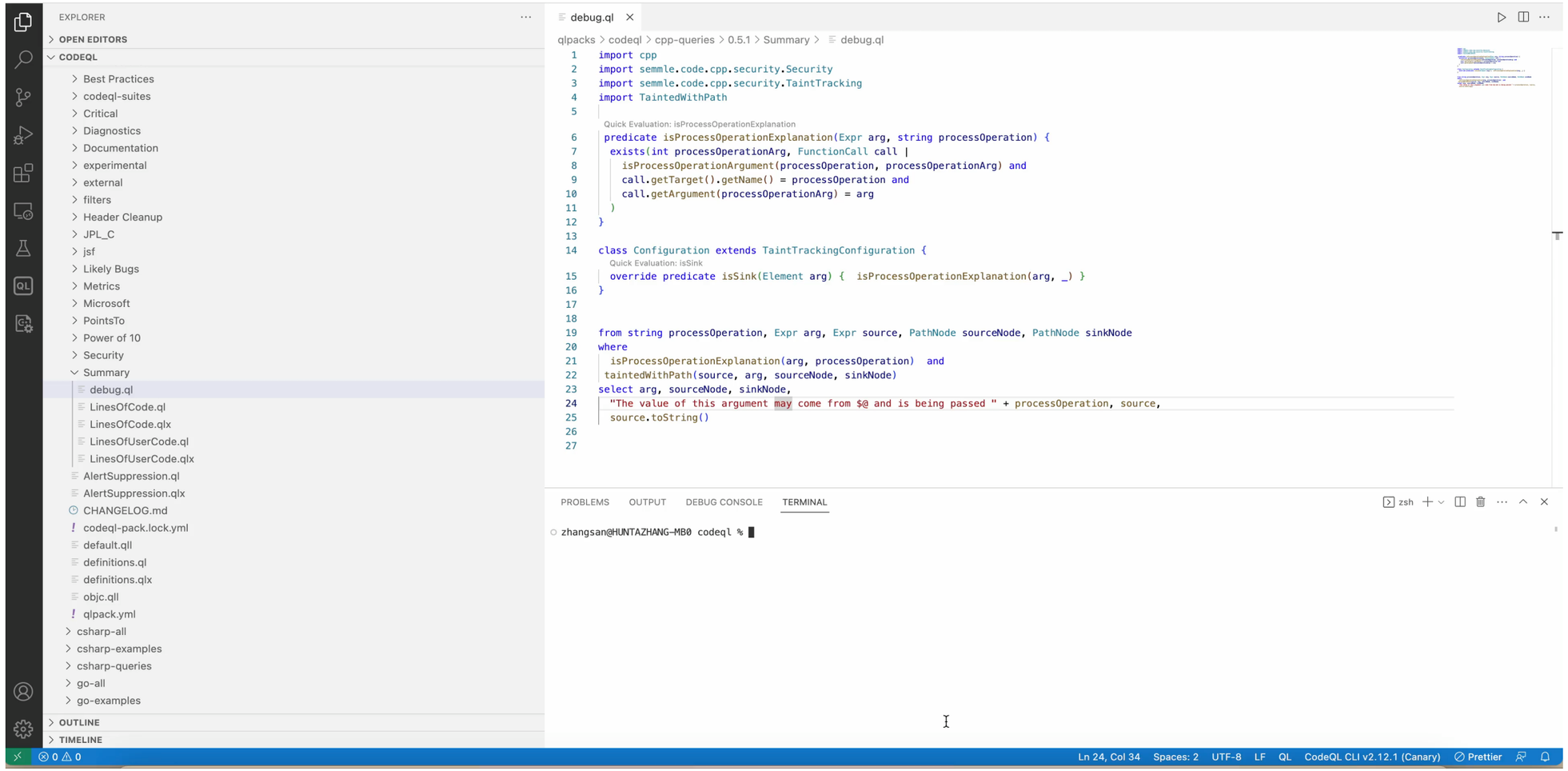
```
99      private static int determineVerbosity() {
100         try {
101             return Integer.parseInt(System.getProperty( key: "semmle.profiler.verbosity",  def: "0"));
102         } catch (NumberFormatException var1) {
103             Exceptions.ignore(var1,  message: "If unparseable, default to 0");
104             return 0;
105         }
106     }
```

Undisclosed system tuning parameters

```
130 exec "/opt/homebrew/opt/openjdk/bin/java" \
131 -Dsemmle.profiler.verbosity=5  -Dcom.semmle.loglevels="com.semmle=ALL"
132 -agentlib:jdwp=transport=dt_socket,server=y,suspend=y,address=*:5005\
```

Added launch parameters

# Demo


EXPLORER
OPEN EDITORS
CODEQL
Best Practices
codeql-suites
Critical
Diagnostics
Documentation
experimental
external
filters
Header Cleanup
JPL_C
jsf
Likely Bugs
Metrics
Microsoft
PointsTo
Power of 10
Security
Summary
debug.ql
LinesOfCode.ql
LinesOfCode.qlx
LinesOfUserCode.ql
LinesOfUserCode.qlx
AlertSuppression.ql
AlertSuppression.qlx
CHANGELOG.md
codeql-pack.lock.yml
default.qll
definitions.ql
definitions.qlx
objc.qll
qlpack.yml
csharp-all
csharp-examples
csharp-queries
go-all
go-examples
OUTLINE
TIMELINE
debug.ql
qlpacks > codeql > cpp-queries > 0.5.1 > Summary > debug.ql
import cpp
import semmle.code.cpp.security.Security
import semmle.code.cpp.security.TaintTracking
import TaintedWithPath
Quick Evaluation: isProcessOperationExplanation
predicate isProcessOperationExplanation(Expr arg, string processOperation) {
exists(int processOperationArg, FunctionCall call |
isProcessOperationArgument(processOperation, processOperationArg) and
call.getTarget().getName() = processOperation and
call.getArgument(processOperationArg) = arg
)
}
class Configuration extends TaintTrackingConfiguration {
Quick Evaluation: isSink
override predicate isSink(Element arg) { isProcessOperationExplanation(arg, _) }
}
from string processOperation, Expr arg, Expr source, PathNode sourceNode, PathNode sinkNode
where
isProcessOperationExplanation(arg, processOperation) and
taintedWithPath(source, arg, sourceNode, sinkNode)
select arg, sourceNode, sinkNode,
"The value of this argument may come from $@ and is being passed " + processOperation, source,
source.toString()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
zsh
zhangsan@HUNTAZHANG-MB0 codeql %
0 0
Ln 24, Col 34
Spaces: 2
UTF-8
LF
QL
CodeQL CLI v2.12.1 (Canary)
Prettier

# OPEN SOURCE

https://github.com/YunDingLab/codeql-binary

# Q&A

huntazhang@tencent.com
jitxie@tencent.com

腾讯安全云鼎实验室
TENCENT SECURITY YUNDING LAB

# Thanks