



DECEMBER 10-11, 2025

EXCEL LONDON / UNITED KINGDOM

Flaw And Order: Finding The Needle In The Haystack Of CodeQL Using LLMs

Simcha Kosman

CyberArk Labs

Linux Kernel
CVE-2025-38676

FFmpeg
CVE-2025-0518

RetroArch
CVE-2025-9136

Redis
CVE-2025-27151

2 days!

80\$ Budget

Bullet3
CVE-2025-8854

Linenoise
CVE-2025-9810

Libretro
CVE-2025-9809

RetroArch
CVE-2025-51938



DECEMBER 10-11, 2025

EXCEL LONDON / UNITED KINGDOM

Flaw And Order: Finding The Needle In The Haystack Of CodeQL Using **LLMs**

Simcha Kosman

CyberArk Labs

Can LLMs Really Find Vulnerabilities?

Here are 10 related curl/libcurl source files (HTTP, HTTP/2, proxy handling, cookies, URL parsing, transfer state machine).

Assume they are compiled with modern curl defaults.

Task: Identify 10 security vulnerabilities in this code (memory safety, auth/crypto issues, logic bugs, or injection issues).

For each vulnerability, explain:

1. The specific vulnerable lines/functions
2. The exact exploit scenario
3. Why existing checks or control flow do not prevent it

How to Use Data AI CME Stopperabilities?

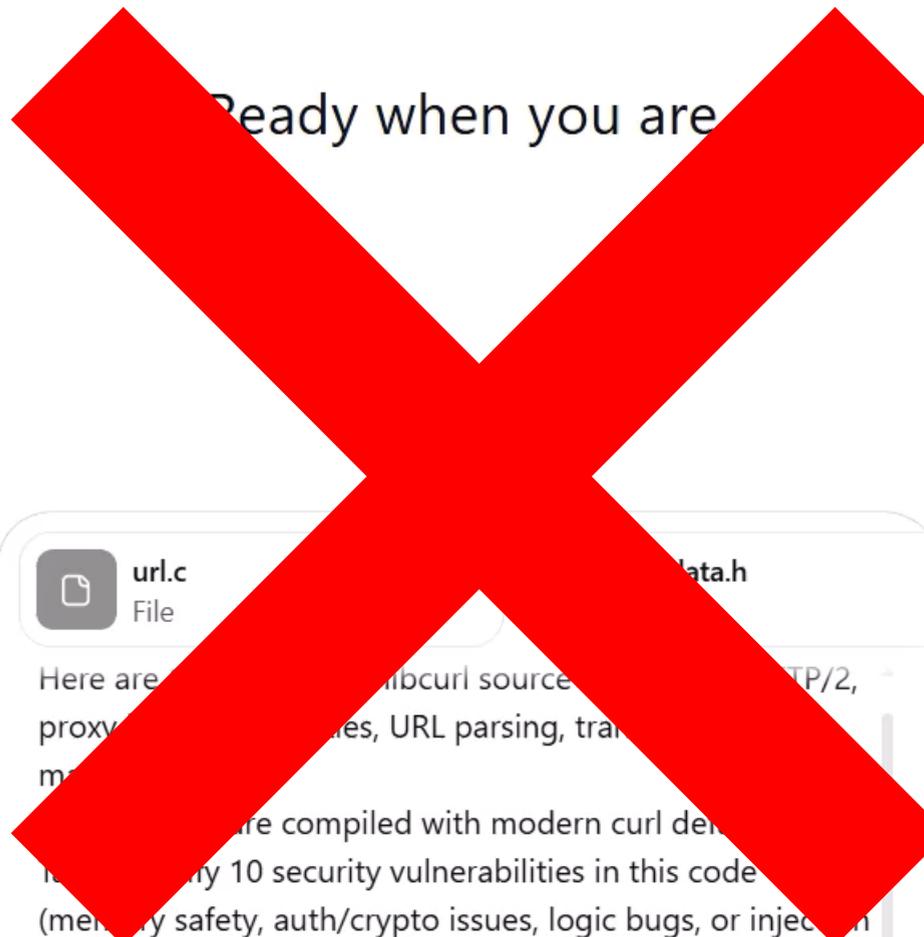
 **bagder** curl staff
closed the report and changed the status to ● Not Applicable.
October 29, 2025, 12:15pm UTC
First: drop the AI slop from your responses. It is insulting to us

 **bagder** curl staff posted a comment.
I presume found by AI and no human intelligence applied.

 **bagder** curl staff posted a comment.
Because of your offensive use of AI here, you are banned from our program.

ChatGPT 5.1 Instant

Ready when you are



 **url.c**
File

Here are the libcurl source files, URL parsing, trans...
...are compiled with modern curl de...
...any 10 security vulnerabilities in this code...
(memory safety, auth/crypto issues, logic bugs, or injection...
issues).
For each vulnerability, explain:
1. The specific vulnerable lines/functions
2. The exact exploit scenario

#BHEU @BlackHatEvents

Simcha Kosman

- **8+ years in Cyber Security**
- **Works for CyberArk Labs**
- **Security for AI**
- **Security FROM AI**
- **AI for Security**



CYBERARK[®]
THE IDENTITY SECURITY COMPANY[®]

Agenda

- **Challenges For Vulnerability Hunting Using LLMs**
- **Current Community Approaches**
- **Our Approach**
- **Overcoming Our Own Challenges**
- **Let's Have Some Fun**

Challenges for Vulnerability Hunting Using LLMs

What problems must you solve before you can even begin

Why a Simple Approach Fails



LLMs Predict, They Don't Understand Code



The WHERE and WHAT problems



The WHERE Problem

- **WHERE** in the code does the vulnerability exist?
- **More Options** → **More Hallucinations**



The WHAT Problem

- **WHAT** type of vulnerability are we looking for?
- **More Options** → **More Hallucinations**

Current Community Approaches

What others tried and what we learned from it

Google Big Sleep



Naptime and now Big Sleep ... discover exploits for variants of previously found and patched vulnerabilities

...

By providing a starting point – such as the details of a previously fixed vulnerability – we remove a lot of ambiguity from vulnerability research



<https://googleprojectzero.blogspot.com/2024/10/from-naptime-to-big-sleep.html>

Google Big Sleep

- **WHERE**



- **WHAT**



OpenAI Aardvark



Aardvark works by monitoring commits and changes to codebases ... Our testing shows that around 1.2% of commits introduce bugs.



<https://openai.com/index/introducing-aardvark>

OpenAI Aardvark

- **WHERE**



- **WHAT**

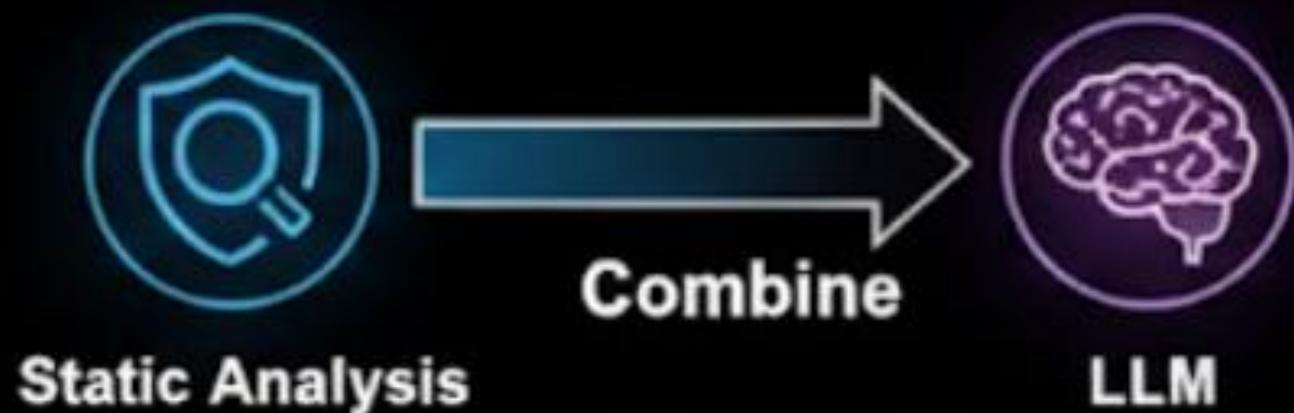


Our Approach

A different way of thinking about the problem

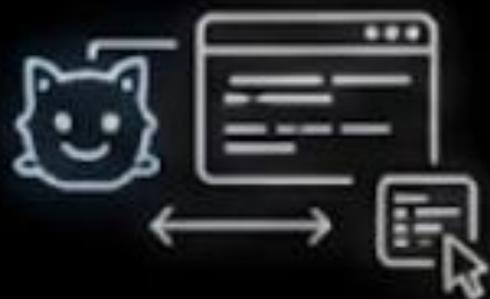
Our Approach

Let's combine static analysis with LLM

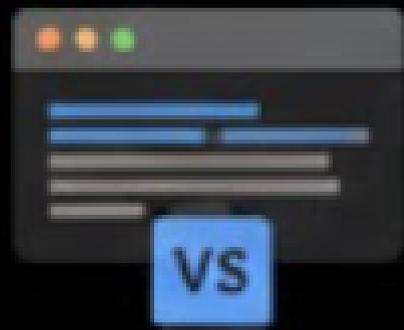


Why does static analysis need LLM?

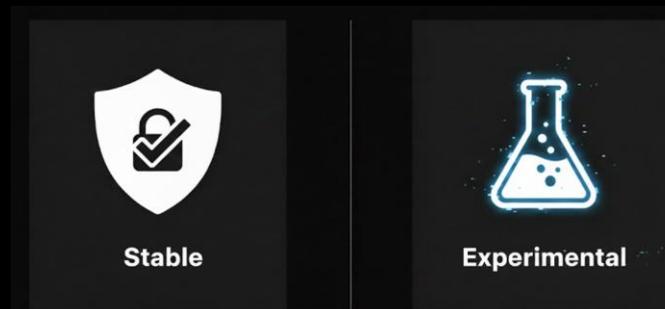
CodeQL Static Analyzer



GitHub Actions



CLI / VScode



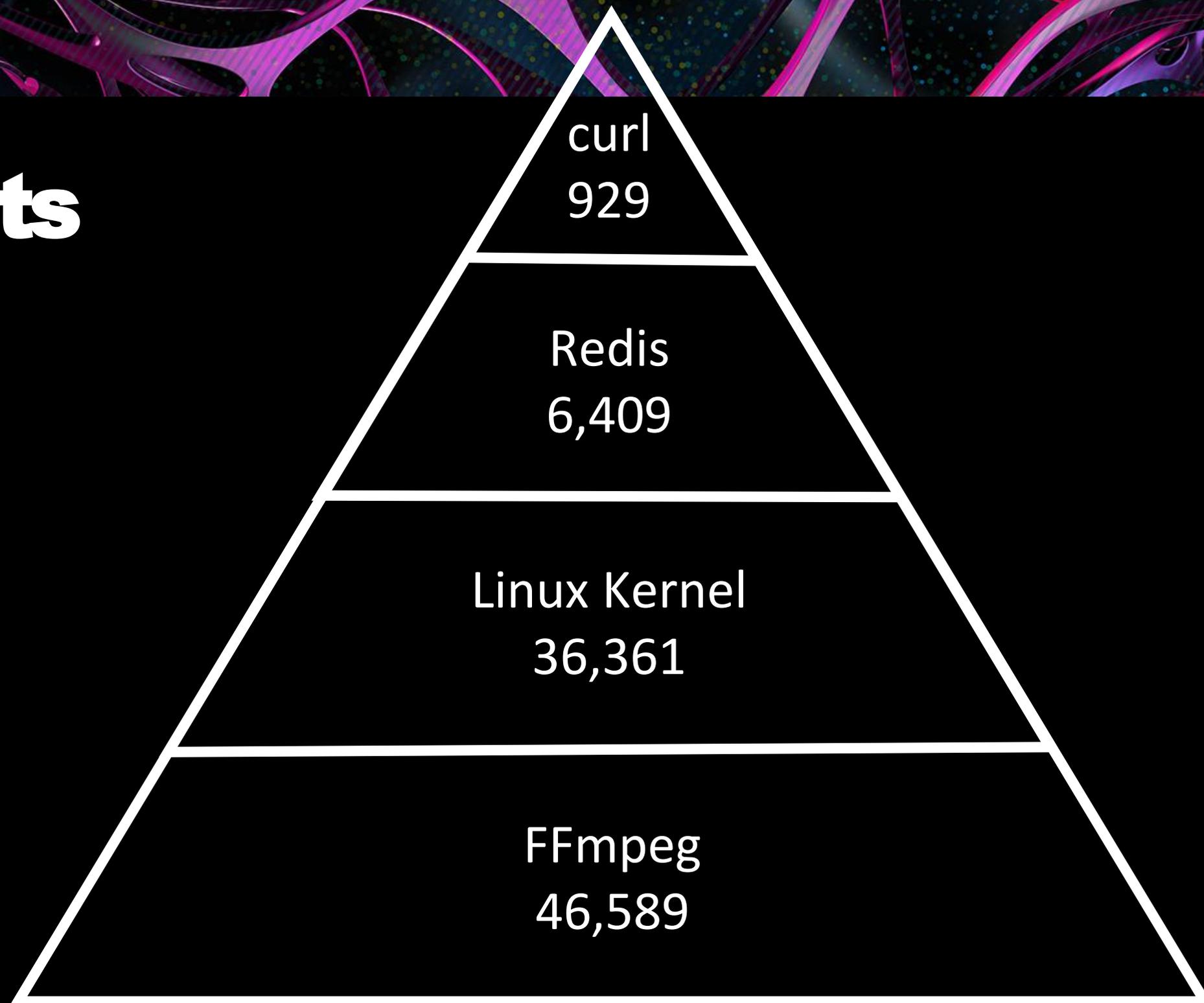
Two types of rules



**Multi-Language
Support**

Compile / Scan

CodeQL Results



Let's Do Some Math

$$929 \text{ (curl)} + 6,409 \text{ (redis)} + 36,361 \text{ (linux)} + 46,589 \text{ (ffmpeg)} = 90,288 \text{ issues}$$

$$90,288 \times 3 \text{ min} = 270,864 \text{ min} = 4,514 \text{ hrs}$$

$$4,514 \text{ hrs} \div 9 = 501.6 \text{ days} = \mathbf{2.1 \text{ years}}$$

Amount of False Positive in Static Analysis

Research by **Amit Seal Ami, Kevin Moran, Denys Poshyvanyk, and Adwait Nadkarni**
University of Central Florida

“

In one tool we use, about 80% of findings are false positives; only ~20% lead to fixes, and even those rarely serious.

“

I wouldn't mind wading through 100 false positives, if I thought there were actually going to be genuine issues there... we went back to manual reviews.

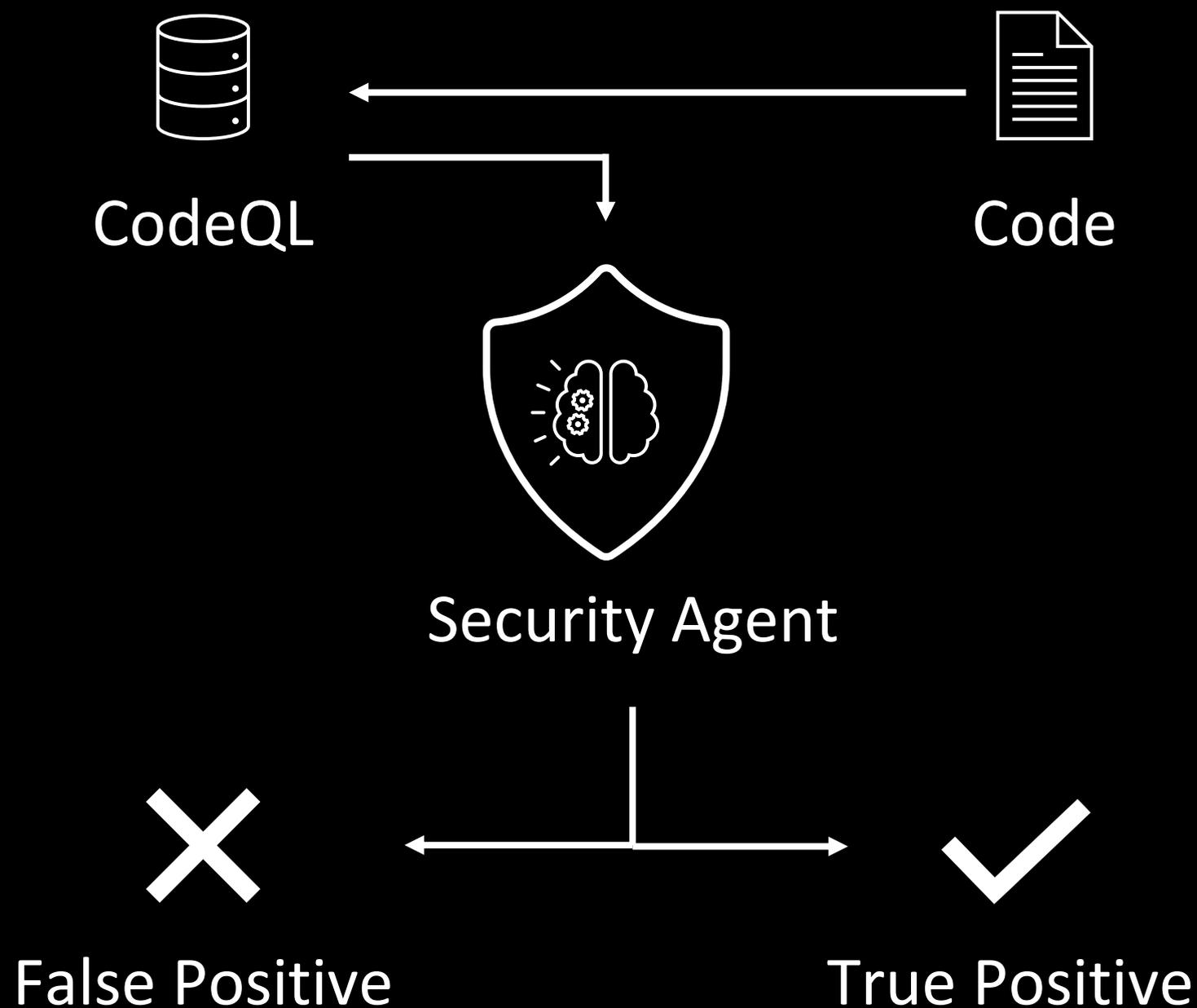
”

”

Our Approach:

CodeQL + LLM

- **CodeQL has many false positives**
- **LLM can pull the **WHERE** and **WHAT** from CodeQL**



Our Twist on the Process



Other tools focus on finding true positives



Our approach focuses on finding false positives



Overcoming Our Own Challenges

How we solved the unexpected obstacles we ran into

Testing Our Approach

33

34

35

```
memcpy(destBuf, sourceBuf, sourceLen + 1);
```

CODEQL ...

LANGUAGE

- All languages
- Actions
- ✓ C / C++
- C#
- Go
- Java
- JavaScript

DATABASES

- ✓ mydb cpp

QUERIES

- ql
 - cpp / ql
 - NotInitialised.ql cpp
 - OverflowCalculated.ql cpp
 - OverflowDestination.ql cpp ▶
 - OverflowStatic.ql cpp
 - ReturnStackAllocatedObject.ql cpp

QUERY HISTORY

- Copy function using source size on mydb - fi... 🗑️

CodeQL Query Results X

« 1 / 1 » Copy function using source size on mydb - finished in 10 seconds (1 results) [11/29/2025, 5:00:23 PM] [Open OverflowDestination.ql](#)

alerts ▼ 1 result Show results in Problems view

Message

> ⓘ 8 To avoid overflow, this operation should be bounded by destination-buffer size, not source-buffer size. [main.c:34:5](#)

Building the Prompt

Static analysis finding:

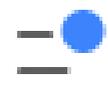
Name: Copy function using source size

Message: To avoid overflow, this operation should be bounded by destination-buffer size, not source-buffer size.

Location: look at main.c:36 with 'memcpy'

Here is the relevant code snippet:

```
memcpy(destBuf, sourceBuf, sourceLen + 1);
```

 ChatGPT 5.1 



Unknown: Need to see the size/allocation of `destBuf` and how `sourceLen` is validated to determine if the copy can exceed the destination buffer.



 Ask anything



ChatGPT can make mistakes. Check important info.

Maybe We Need More Lines?

```
33     sourceLen = read_input(&sourceBuf);
34     if (sourceLen <= 0) {
35         |     return 1;
36     }
37
38     memcpy(destBuf, sourceBuf, sourceLen + 1);
39
40     printf("%s\n", destBuf);
41
42     free(sourceBuf);
43
```

Static analysis finding:

.....

Here is the relevant code snippet:

```
sourceLen = read_input(&sourceBuf);  
if (sourceLen <= 0) {  
    return 1;  
}  
memcpy(destBuf, sourceBuf, sourceLen + 1);  
printf("%s\n", destBuf);  
free(sourceBuf);
```

ChatGPT 5.1

Unknown: I need to see how `destBuf` is allocated (its size) and how `read_input` allocates or fills `sourceBuf` (including whether it guarantees a terminating byte and bounds).



+ Ask anything

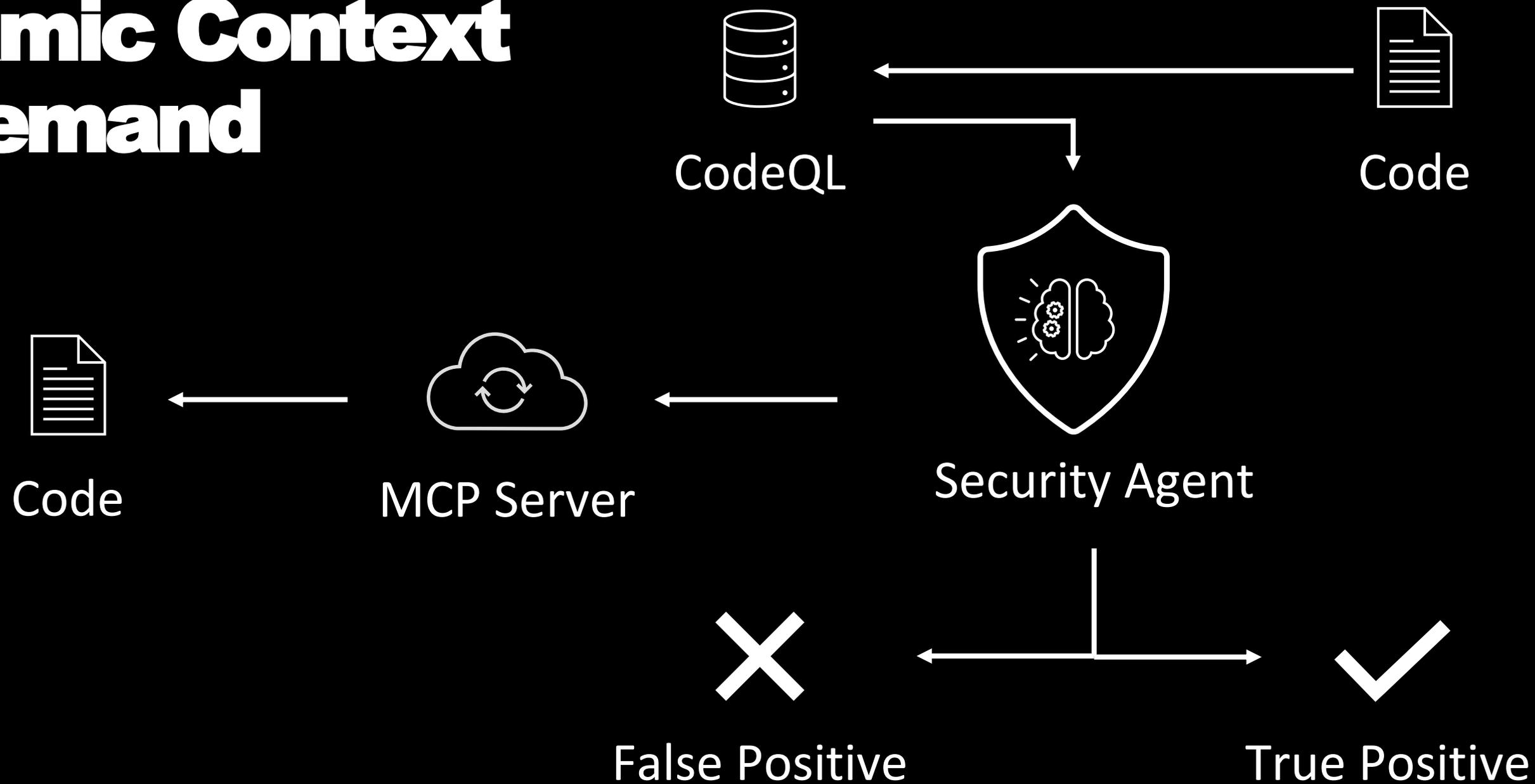


ChatGPT can make mistakes. Check important info.

New Flow

```
28  int main(void) {
29      char destBuf[DEST_SIZE + 1];
30      char* sourceBuf;
31      size_t sourceLen;
32
33      sourceLen = read_input(&sourceBuf);
34      if (sourceLen <= 0) {
35          return 1;
36      }
37
38      memcpy(destBuf, sourceBuf, sourceLen + 1);
39
40      printf("%s\n", destBuf);
41
42      free(sourceBuf);
43
44      return 0;
45  }
```

Dynamic Context on Demand



Can You Even Extract Functions?

vscode-codeql-starter (Workspace) [Dev Container]

CodeQL Query Results

« 1 / 1 »

Copy function using source size on mydb - finished in 10 seconds (1 results)
[11/29/2025, 5:00:23 PM]

[Open OverflowDestination.sql](#)

alerts 1 result

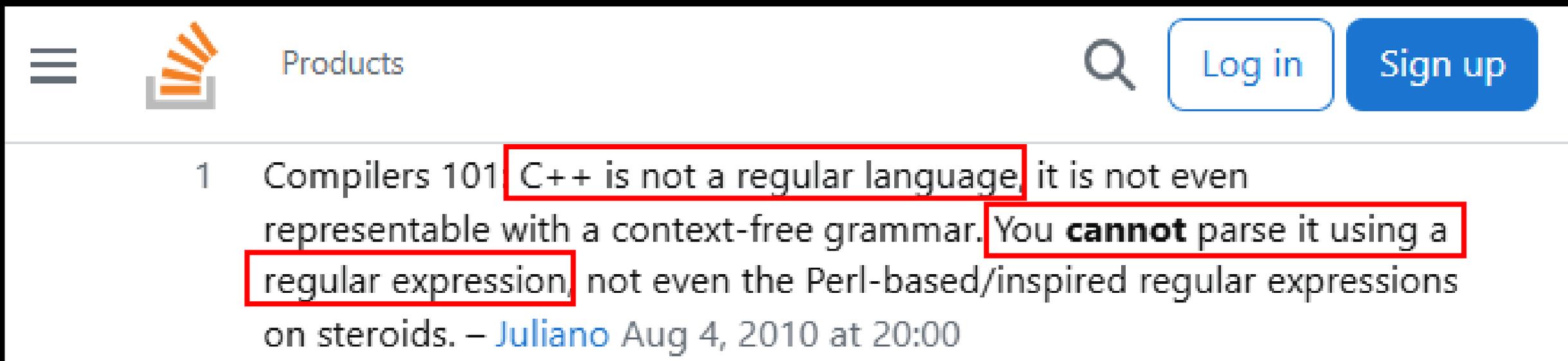
Show results in Problems view

Message

> 8 To avoid overflow, this operation should be bounded by destination-buffer size, not source-buffer size.

[main.c:34:5](#)

Can We Use Regex or A Bracket Count?



The screenshot shows a Stack Overflow page with a navigation bar at the top containing a menu icon, a logo, the word "Products", a search icon, and "Log in" and "Sign up" buttons. Below the navigation bar is a list of answers. The first answer, numbered "1", is by user "Juliano" and is dated "Aug 4, 2010 at 20:00". The answer text is: "Compilers 101 C++ is not a regular language, it is not even representable with a context-free grammar. You cannot parse it using a regular expression, not even the Perl-based/inspired regular expressions on steroids." The text "C++ is not a regular language", "You cannot parse it using a regular expression", and "not even the Perl-based/inspired regular expressions on steroids." are highlighted with red boxes.

1 Compilers 101 C++ is not a regular language, it is not even representable with a context-free grammar. You cannot parse it using a regular expression, not even the Perl-based/inspired regular expressions on steroids. – Juliano Aug 4, 2010 at 20:00

<https://stackoverflow.com/questions/3409210/regex-to-parse-c-c-functions-declarations>

Parsing C Isn't Trivial

- **Macro can replace {}**
- **Brace chars inside string**
- **Looks like a function, but it's a pointer**

```
#define BEGIN {  
#define END }  
  
int log_and_add(int x, int y)  
BEGIN  
    printf("adding {x,y} = %d\n", x + y);  
  
    int (*callback)(int);  
    return x + y;  
END
```

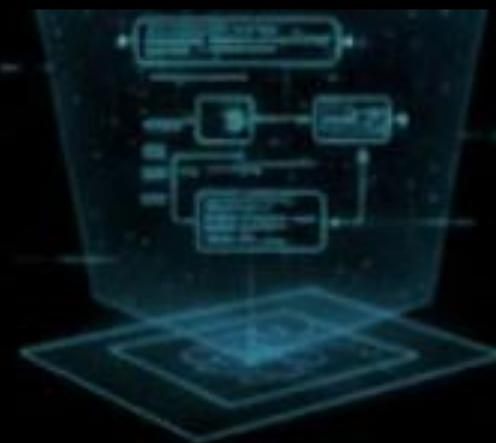
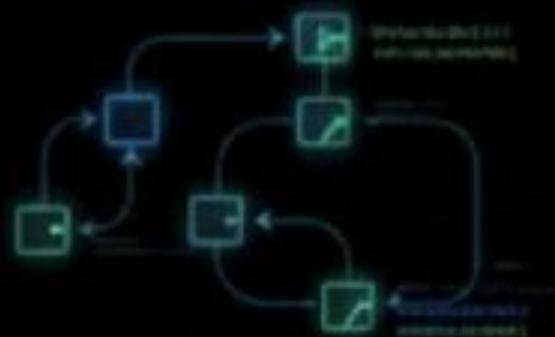
The Context Challenge

- **Function body**
- **Macro definition**
- **Callee function**
- **And more**

```
28 int main(void) {
29     char destBuf[DEST_SIZE + 1];
30     char* sourceBuf;
31     size_t sourceLen;
32
33     sourceLen = read_input(&sourceBuf);
34     if (sourceLen <= 0) {
35         return 1;
36     }
37
38     memcpy(destBuf, sourceBuf, sourceLen + 1);
39
40     printf("%s\n", destBuf);
41
42     free(sourceBuf);
43
44     return 0;
45 }
```

Maybe We Can Use Code Indexer?

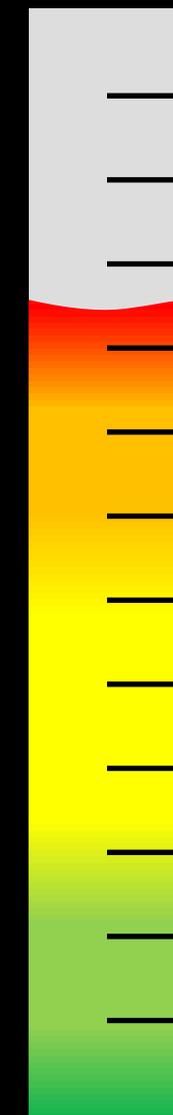
- **Like Static analyzer: Builds a database without running the code**
- **Provides deep semantic information**



Let's try Code Indexer



```
WSL x + v - □ X  
$ docker exec -it -e PYTHONUNBUFFERED=1 elixir-container  
/usr/local/elixir/utils/index-repository connectedhomeip  
https://github.com/project-chip/connectedhomeip
```



Conclusion: Code Indexer Won't Work

- **Indexing large repos takes hours**
- **Third-party deps complicate things**

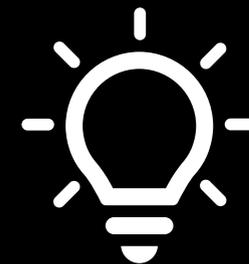
```
WSL x + v - □ x
$ docker exec -it -e PYTHONUNBUFFERED=1 elixir-container
/usr/local/elixir/utils/index-repository connectedhomeip
https://github.com/project-chip/connectedhomeip
```

To check out the Matter repository with all platforms, run the following command:

```
git clone --recurse-submodules git@github.com:project-chip/connectedhomeip.git
```



So... Why Not CodeQL?



**The CodeQL
database for C is
generated during
compilation**



**Third-party code is
included during
compilation**



**We already use the
database to identify
issues**

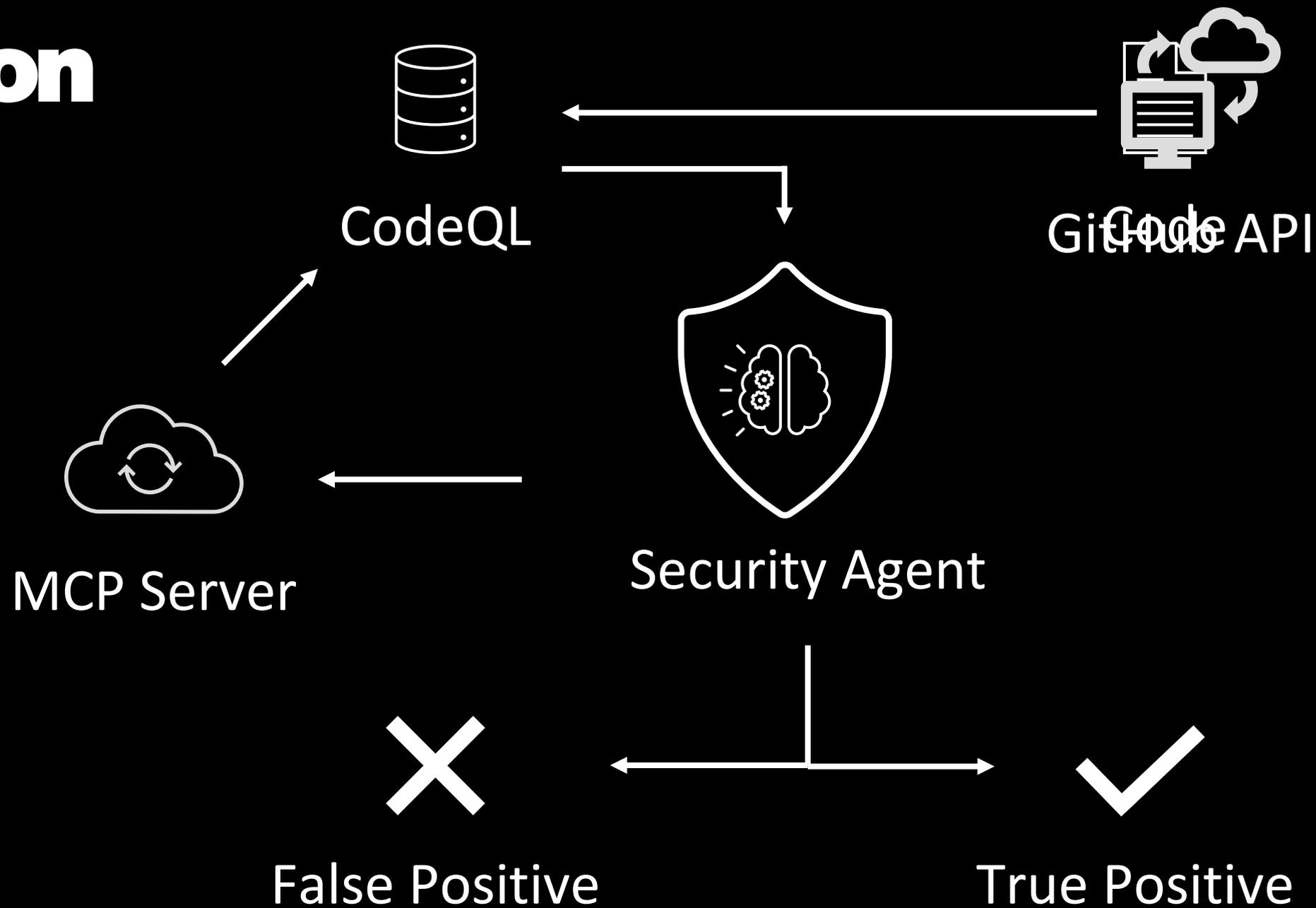
But Building CodeQL Takes Time

About CodeQL databases [↗](#)

To analyze a project, you need to select a CodeQL database for that project. You can select a database locally (from a ZIP archive or an unarchived folder), from a public URL, or from a project's URL on GitHub. Alternatively, you can create a database using the CodeQL CLI, see [Preparing your code for CodeQL analysis](#).

<https://docs.github.com/en/code-security/codeql-for-vs-code/getting-started-with-codeql-for-vs-code/managing-codeql-databases>

Fast Context on Demand

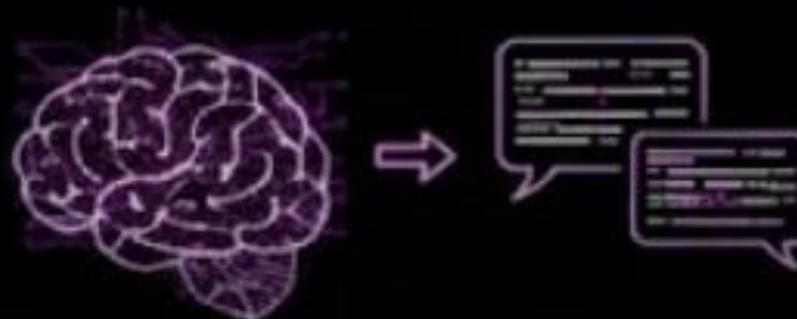


Let's Test "Fast Context on Demand"

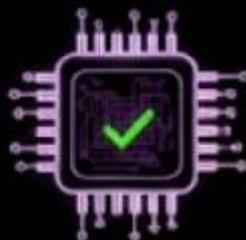


2.5
minutes

Average query execution



Our queries are dynamically generated by LLM requests

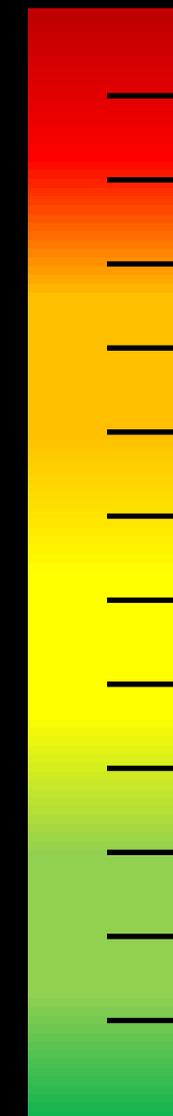


40
seconds

Pre-compiled queries

$$2.5 \text{ min} \times 46,589 \text{ FFmpeg issues} = 93,178$$

A lot of minutes!



What If We Pre Extract Everything

```
1 import cpp
2
3 from GlobalOrNamespaceVariable g
4 select g.getName() as global_var_name,
5        g.getLocation().getFile() as file,
6        g.getLocation().getStartLine() as start_line,
7        g.getLocation().getEndLine() as end_line
```

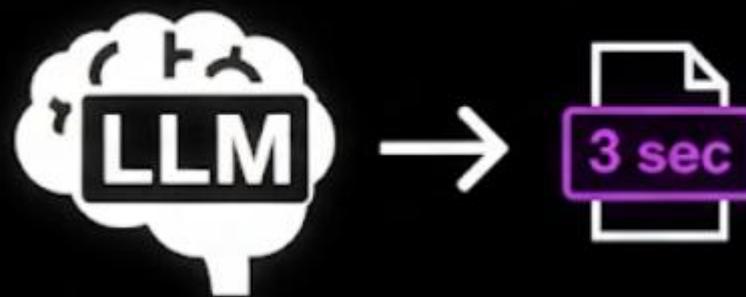
And Save It as a CSV File

```
1  "global_var_name","file","start_line","end_line"  
2  "linktype","/home/runner/work/AdAway/AdAway/tcpdump/jni/libpcap/gencode.c",935,935  
3  "snaplen","/home/runner/work/AdAway/AdAway/tcpdump/jni/libpcap/gencode.c",426,426
```

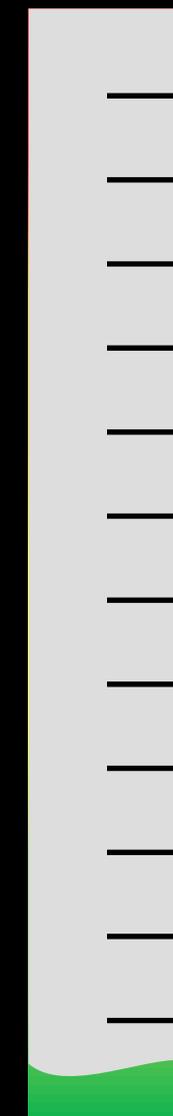
How Long Will It Take?



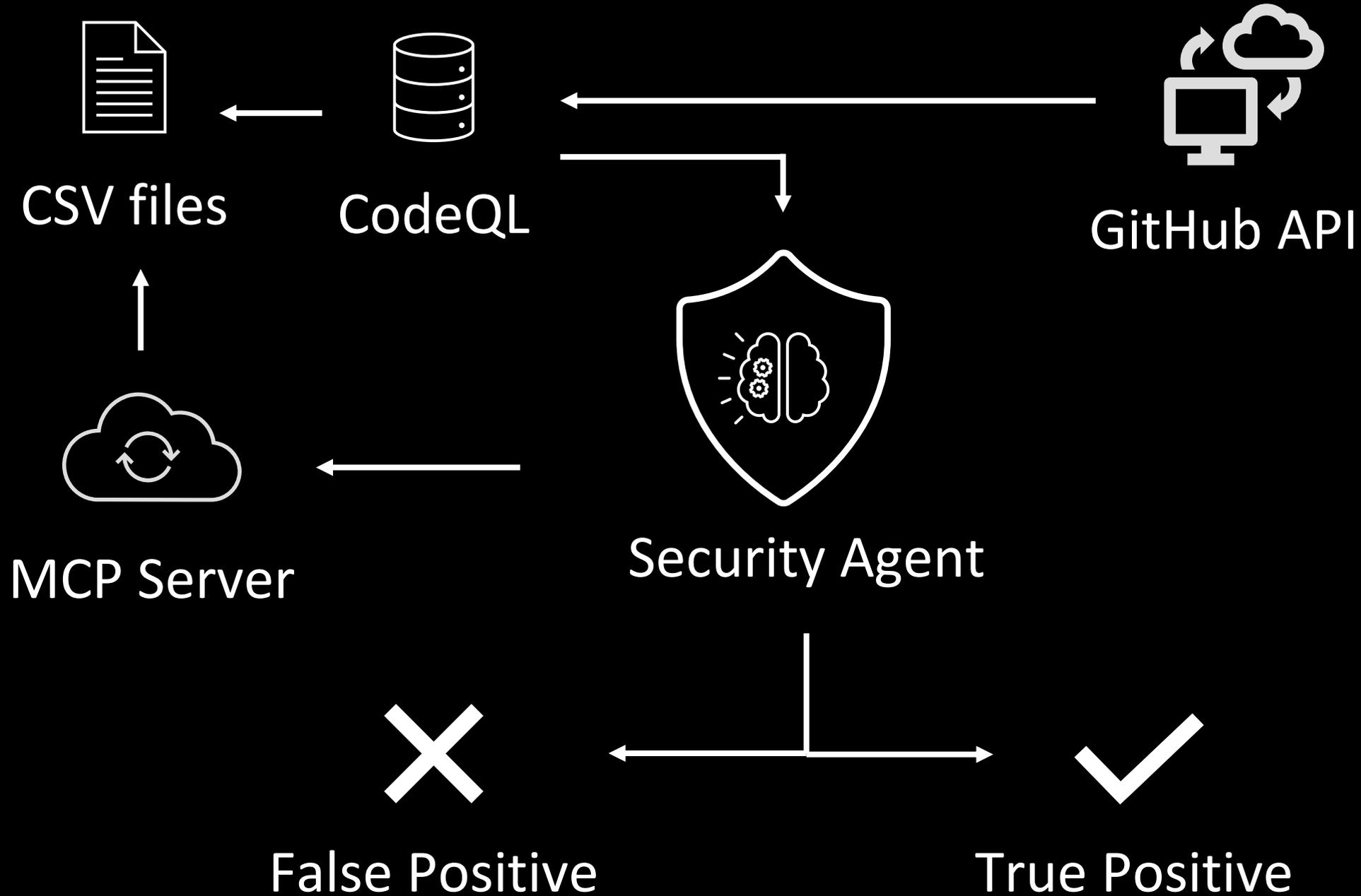
**Queries execution
time**



CSV file search



***Very* Fast Context on Demand**



```
9 size_t read_input(char** sourceBuf) {
10     char* buf = malloc(DEST_SIZE + 1);
11     size_t bufSize = 0;
12     int c;
13
14     while (bufSize < DEST_SIZE) {
15         c = getchar();
16         if (c == EOF || c == '\n') {
17             break;
18         }
19         buf[bufSize++] = (char)c;
20     }
21
22     buf[bufSize] = '\0';
23
24     *sourceBuf = buf;
25     return bufSize;
26 }
27
```

```
7 #define DEST_SIZE 16
```

```
28 int main(void) {
29     char destBuf[DEST_SIZE + 1];
30     char* sourceBuf;
31     size_t sourceLen;
32
33     sourceLen = read_input(&sourceBuf);
34     if (sourceLen <= 0) {
35         return 1;
36     }
37
38     memcpy(destBuf, sourceBuf, sourceLen + 1);
39
40     printf("%s\n", destBuf);
41
42     free(sourceBuf);
43
44     return 0;
45 }
```

- **Source actual size**

Static analysis finding:

.....

Here is the relevant code snippet:

```
#define DEST_SIZE 16
size_t read_input(char** sourceBuf) {
    char* buf = malloc(DEST_SIZE + 1);
    size_t bufSize = 0;
    int c;
    while (bufSize < DEST_SIZE) {
        c = getchar();
        if (c == EOF || c == '\n') {
            break;
        }
        buf[bufSize++] = (char)c;
    }
    buf[bufSize] = '\0';
    *sourceBuf = buf;
    return bufSize;
}
int main(void) {
    char destBuf[DEST_SIZE + 1];
    char* sourceBuf;
    size_t sourceLen;
    sourceLen = read_input(&sourceBuf);
    if (sourceLen <= 0) {
        return 1;
    }
    memcpy(destBuf, sourceBuf, sourceLen + 1);
    printf("%s\n", destBuf);
    free(sourceBuf);
    return 0;
}
```

☰ ChatGPT 5.1 ▾



Bug: The copy uses `sourceLen + 1` without checking it against the size of `destBuf`, so a full-length read (16 chars plus terminator) will overflow the 17-byte destination by one byte.



+ Ask anything



ChatGPT can make mistakes. Check important info.

The Focus Challenge

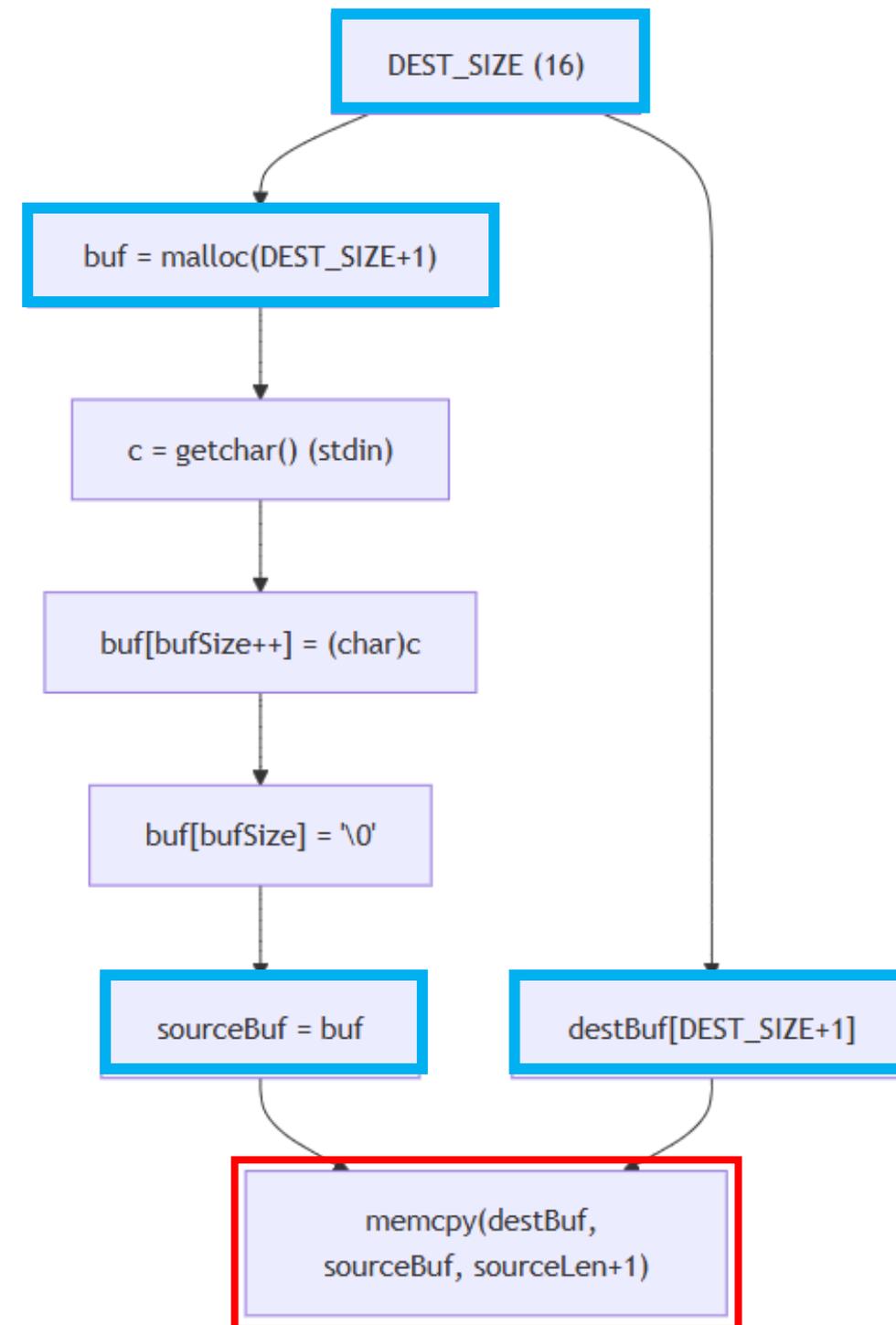
- CodeQL results ***focus*** on whether this is the source size
- We need the LLM to ***focus*** on the actual source size



```
7 #define DEST_SIZE 16
8
9 size_t read_input(char** sourceBuf) {
10     char* buf = malloc(DEST_SIZE + 1);
11     size_t bufSize = 0;
12     int c;
13
14     while (bufSize < DEST_SIZE) {
15         c = getchar();
16         if (c == EOF || c == '\n') {
17             break;
18         }
19         buf[bufSize++] = (char)c;
20     }
21
22     buf[bufSize] = '\0';
23
24     *sourceBuf = buf;
25     return bufSize;
26 }
27
28 int main(void) {
29     char destBuf[DEST_SIZE + 1];
30     char* sourceBuf;
31     size_t sourceLen;
32
33     sourceLen = read_input(&sourceBuf);
34     if (sourceLen <= 0) {
35         return 1;
36     }
37
38     memcpy(destBuf, sourceBuf, sourceLen + 1);
39
40     printf("%s\n", destBuf);
41
42     free(sourceBuf);
43
44     return 0;
45 }
```

The Focus Challenge

- CodeQL results ***focus*** on whether this is the source size
- We need the LLM to ***focus*** on the actual source size



The Guided Question Technique



Use only low-temperature models

Instruct the LLM to answer general data-flow and code-flow questions

Static analysis finding:

.....

Here is the relevant code snippet:

.....

Before deciding if the issue is real, the LLM must reason about the lifecycle of the source and destination variables

1. Where are they **declared**, what are their **sizes**, and do those sizes ever **change**?
2. If they **take values from other variables**, where are those variables declared and what sizes do they have?
3. Are there any **checks** or constraints on the source, the destination, or the variables assigned to them?
4. What **other operations** act on these variables?

Let's Test Guided Questions

Final determination:

Not a bug: The copy length is derived from `sourceLen`, which is guaranteed not to exceed `DEST_SIZE`, and both buffers are `DEST_SIZE + 1` bytes.



ChatGPT 5.1

Answers to your questions:

1. Declarations and sizes

- `destBuf` is declared in `main` as `char destBuf[DEST_SIZE + 1]`, which gives it a fixed size of 17 bytes. Its size never changes.
- `sourceBuf` is a `char*` in `main`; it points to a heap buffer allocated in `read_input`.
- Inside `read_input`, the allocated buffer is `malloc(DEST_SIZE + 1)`, so it is also always 17 bytes. Its size never changes after allocation.

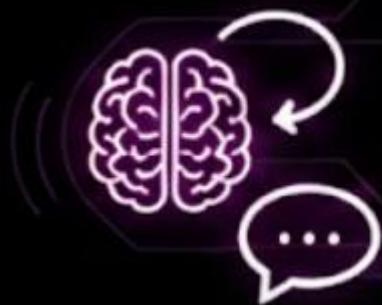
2. Variables they take values from

- `sourceBuf` receives the pointer to the allocated buffer `buf` from `read_input`.
- `sourceLen` receives the return value of `read_input`, which is a `size_t` counting

- `memcpy(destBuf, sourceBuf, sourceLen + 1)` copies the string including the null terminator.
- `printf("%s\n", destBuf)` reads from `destBuf`.
- `free(sourceBuf)` frees the heap allocation.

Final determination:

Why Guided Questions Worked



LLM predicts the next word from past context

Past context now includes what the LLM should focus on



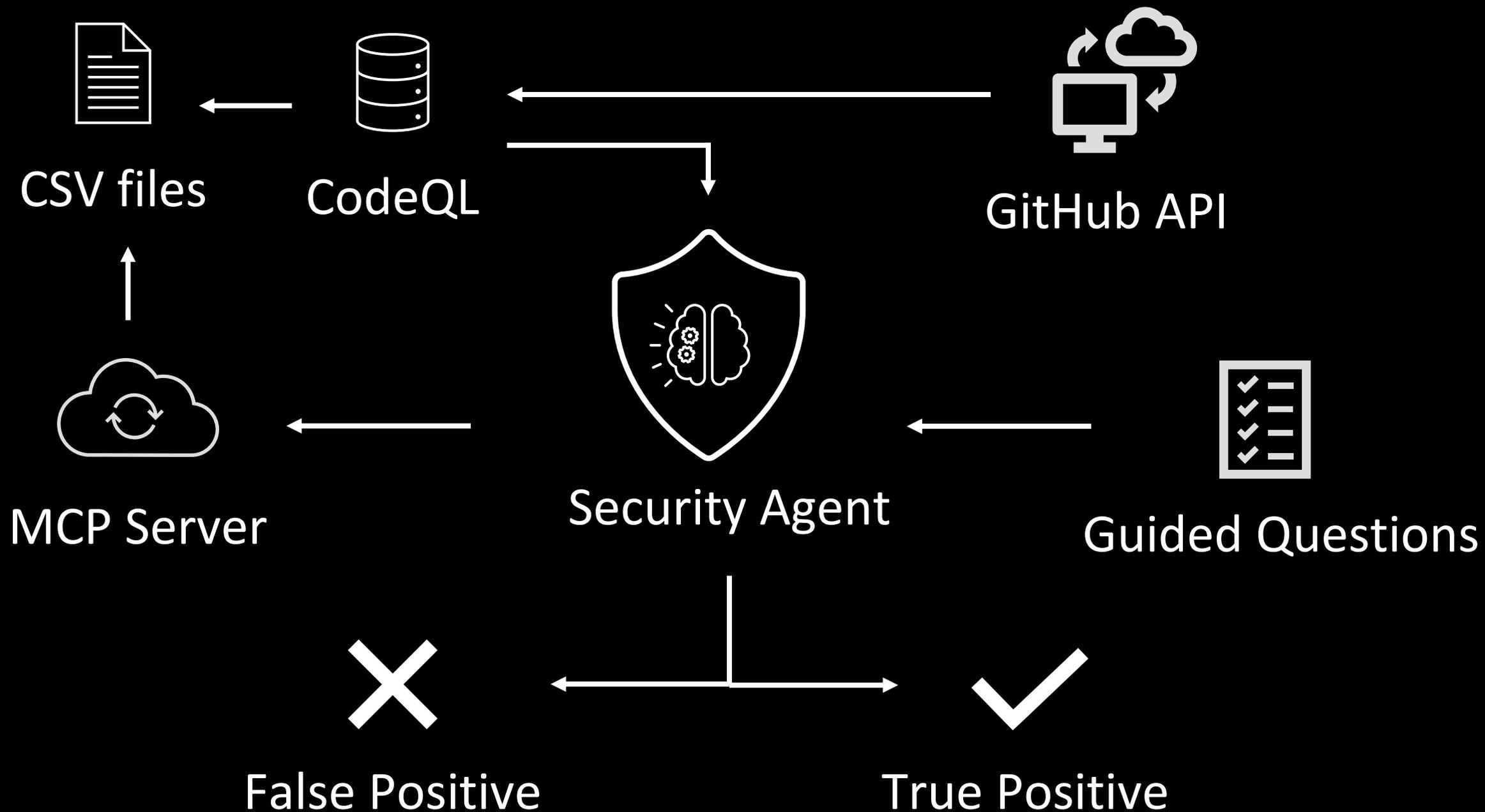
Why Guided Questions Worked



For each issue, we must understand what the LLM should focus on and guide it with questions to explain with words

Impact: We can use weaker CodeQL queries

Final Flow



Recap



- **The 'Where' and 'What' problems**

Use a Static analyzer



- **The Context challenge**

Use CodeQL GitHub database with CSV files



- **The Focus challenge**

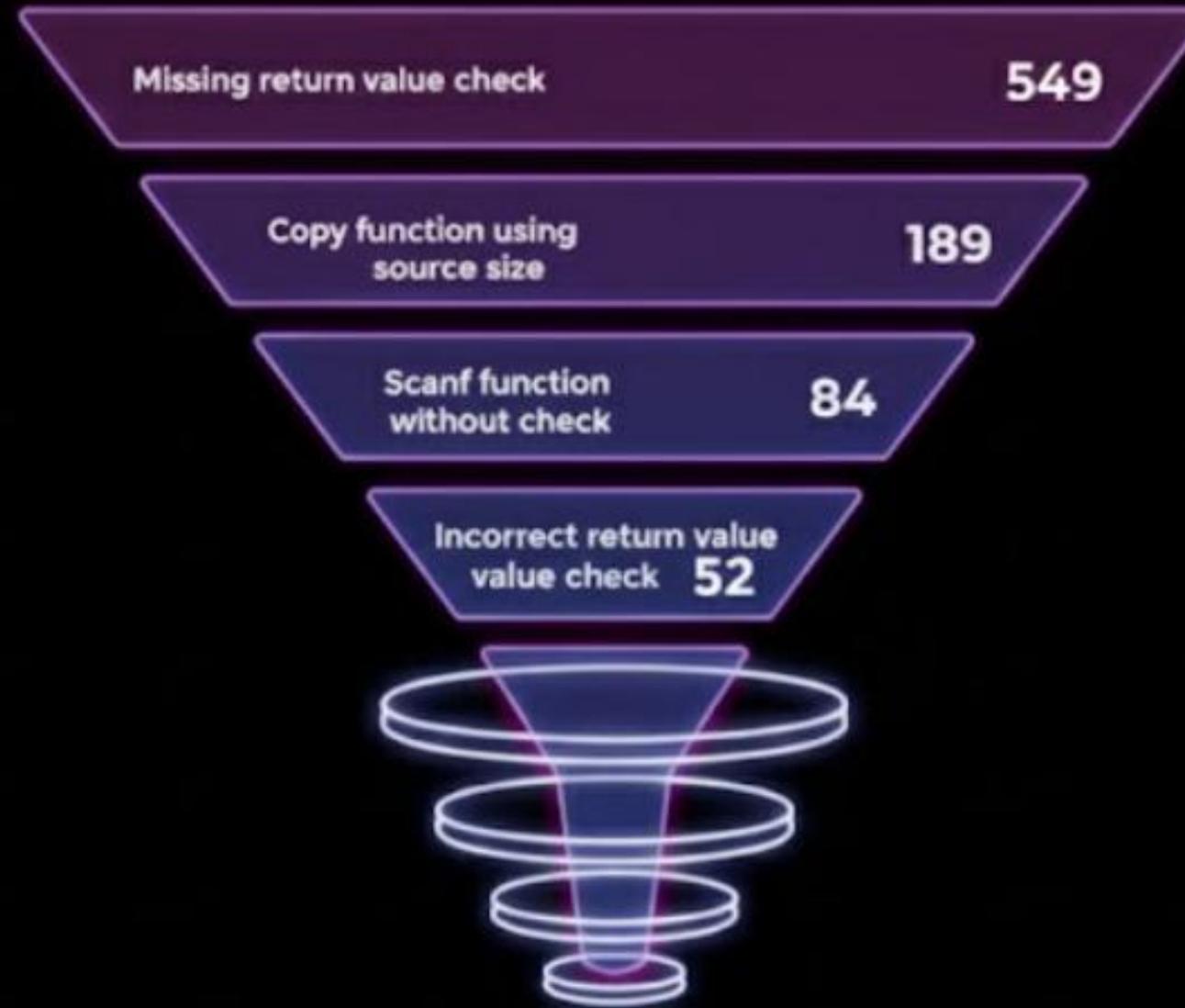
Use the Guided Questions technique

Let's Have Some Fun

Testing our theory in the real world



874



239 (27.34%)

Missing Return Value Check for Scanf



549

Issues found



162

**70% reduction in
False Positive**



101

Real issues



CVE

1

**RetroArch
CVE-2025-51938**

Incorrect Return Value Check for Scanf



52

Issues found



19

**63% reduction in
False Positive**



1

Real issues



CVE

1

**FFmpeg
CVE-2025-0518**

Copy Function Using Source Size



189

Issues found



16

**91% reduction in
False Positive**



2

Real issues



CVE

2

**Redis
CVE-2025-27151
Libretro
CVE-2025-9809**

Scanf Function Without Specified Length



84

Issues found



42

**50% reduction in
False Positive**



35

Real issues



CVE

3

**Linux Kernel
CVE-2025-38676
Bullet3
CVE-2025-8854
RetroArch
CVE-2025-9136**

Linux Kernel
CVE-2025-38676

FFmpeg
CVE-2025-0518

RetroArch
CVE-2025-9136

Redis
CVE-2025-27151

Bullet3
CVE-2025-8854

Linenoise
CVE-2025-9810

Libretro
CVE-2025-9809

RetroArch
CVE-2025-51938



Vulnhalla

**A tool that allows
only the True and
Brave issues from
CodeQL to pass
through its gates**



Issues

Select an issue to view details

ID	LLM decision	Manual decision	Repo	Issue
1	False Positive	Not Set	AFLplusplus/cpp	Copy f
2	False Positive	Not Set	AFLplusplus/cpp	Copy f
3	True Positive	Not Set	aircrack-ng/cpp	Copy f
4	False Positive	Not Set	alibaba/cpp	Copy f
5	False Positive	Not Set	allinurl/cpp	Copy f
6	False Positive	Not Set	allinurl/cpp	Copy f
7	False Positive	Not Set	allinurl/cpp	Copy f
8	False Positive	Not Set	allinurl/cpp	Copy f
13	False Positive	Not Set	b4winckler/macvim	Copy f
14	False Positive	Not Set	b4winckler/macvim	Copy f
15	False Positive	Not Set	b4winckler/macvim	Copy f
16	True Positive	Not Set	b4winckler/macvim	Copy f
18	False Positive	Not Set	b4winckler/macvim	Copy f
19	False Positive	Not Set	b4winckler/macvim	Copy f
20	False Positive	Not Set	b4winckler/macvim	Copy f
21	False Positive	Not Set	b4winckler/macvim	Copy f
23	False Positive	Not Set	b4winckler/macvim	Copy f

Showing 183 of 183 issues

Search by issue name, file, repo, LLM decision, or manual decision...

Language: C (only language currently supported)

Filter:

All

Refresh

Run Analysis

Key Bindings:

↑/↓: Navigate | Tab: Switch focus | Enter: Show details | /: Search | [: Resize left |]: Resize right | r: Reload | q: Quit

q Quit r Reload / Search esc Clear Search [Resize Left] Resize Right

^p palette

How Can You Help



Join us in adding support for more programming language

Contribute new guided questions that strengthen issue coverage





Vulnhalla



Blog

Black Hat Europe - Sound Bytes

- **Code navigation for nearly any public repo in under 15 minutes**
- **Write simpler CodeQL queries combined with guided questions**
- **Cut false positives dramatically and uncover real vulnerabilities faster**



Simcha Kosman

THANK YOU!



CYBERARK[®]

THE IDENTITY SECURITY COMPANY[®]