Assisting Static Analysis with Large Language Models: A ChatGPT Experiment

HAONAN LI, YU HAO, YIZHUO ZHAI, ZHIYUN QIAN

UC RIVERSIDE

Background & Motivation

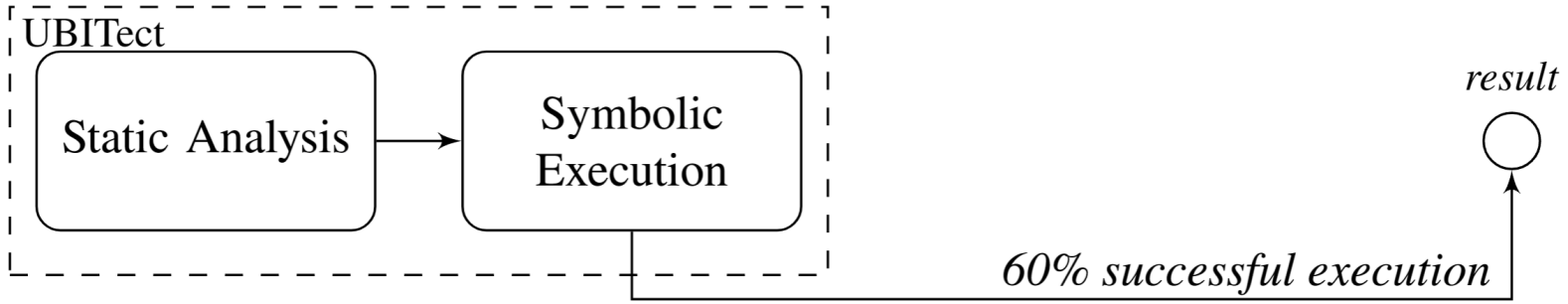
1. Existing static analysis, UBITect [1], detecting use-before-initialization in the Linux kernel, *may ignore 40% potential bugs* due to time/mem out
2. ChatGPT can understand code, may handle these ignored potential bugs

[1] Yizhuo Zhai, Yu Hao, et.al. "UBITect: A Precise and Scalable Method to Detect Use-before-Initialization Bugs in Linux Kernel." In FSE'20. https://doi.org/10.1145/3368089.3409686.

Background: UBITect

Two stages:

1. Static analysis (scalable, imprecise)
2. Symbolic execution to verify potential bugs (precise, inefficient)



40% undecided: symbolic execution timeout/memory out

Case study: `sscanf`

No UBI bug here, but:

1. Static analysis (path-insensitive): may initialize a, b, c, d
2. Symbolic execution: *timeout because of path explosion*

```
1  static int libcfs_ip_str2addr(...){
2    unsigned int a, b, c, d;
3    if (sscanf(str, "%u.%u.%u.%u%n",
4           &a, &b, &c, &d, &n) >= 4 && ...){
5      // use of a, b, c, d
6    }
7  }
```

Case study: `sscanf`

No UBI bug here, but:

1. Static analysis (path-insensitive): may initialize a, b, c, d
2. Symbolic execution: *timeout because of path explosion*

```
14  int vsscanf(const char *buf, const char *fmt,
    ↪  va_list args){
15    const char *str = buf;
16    ...
17    while (*fmt) {
18      switch (*fmt++) {
19        case 'c': {
20          char *s = (char *)va_arg(args, char*);
21          ...
22          do { *s++ = *str++; }
23          while (...*str); num++;
24        }
25      }
26      ...
27      num++;
28          ...
29    }
30    return num;
31  }
```

Idea: Ask ChatGPT for Code Behavior

```
1    static int libcfs_ip_str2addr(...){
2      unsigned int a, b, c, d;
3      if (sscanf(str, "%u.%u.%u.%u%n",
4            &a, &b, &c, &d, &n) >= 4 && ...){
5              // use of a, b, c, d
6      }
7    }
```

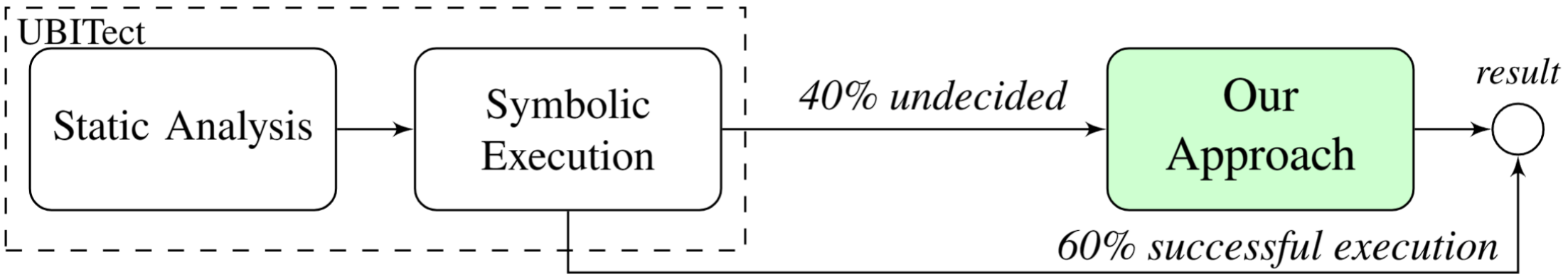
Scan to see the chat!

Ask ChatGPT (simplified):

- *Q: "are variables* **`a, b, c, d`** *to be initialized before reaching Line 5"*
- *A: "*`a, b, c, d`*" are initialized at* `sscanf(…)>=4`

Workflow

1. For each potential use-before-initialization, find the “use” site
2. Ask ChatGPT, “*Whether the variables be initialized* **before the use**”
3. If answered “**initialized**”, then not a bug



Workflow: with UBITect

Challenge: Unfamiliar Functions

How about “unfamiliar functions”🤔

- Not all functions are popular as **`sscanf`**, ChatGPT can’t recognize them

We can’t provide all relevant context:

- Token limitation: GPT-4 supports 32k tokens
- Expensive, slow, and low-quality response for long content [2]

Intuition: we have an AI! Let ChatGPT decide for itself!

[2] Bito AI, Claude 2.1 (200K Context Window) Benchmarks. https://bito.ai/blog/claude-2-1-200k-context-window-benchmarks/

Progressive Prompt

- Prompt with, *"if you meet **unfamiliar functions**, tell me"*
- Then we provide function definitions
- Fully automated

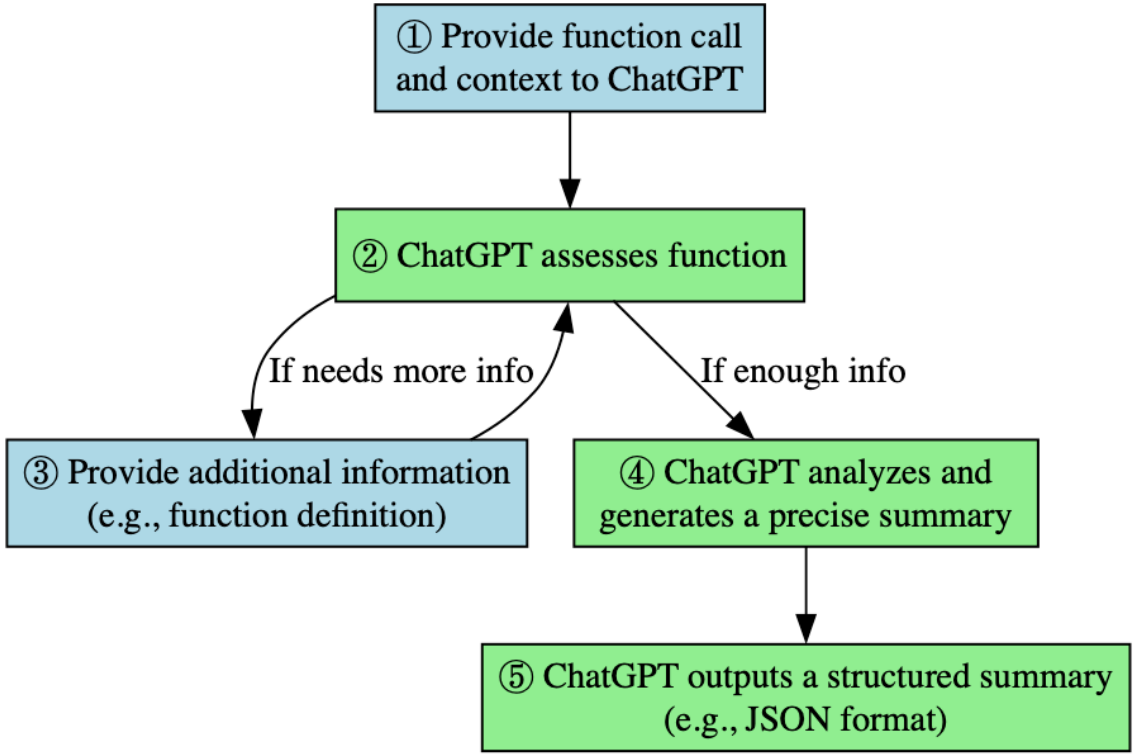


Figure 3: The workflow. Green stands for what ChatGPT (API) responds (role: assistant) and the blue stands for what the user (script, role: user) prompts.

Result

- For false positives (not a bug), performs well
- For false negatives (missed bugs), not perfect yet

Table 1: Selected results. "S?" for Soundness and "C?" for Completeness. Type for challenges: *Inherent Knowledge Boundaries* (KB) and *Exhaustive Path Exploration* (PE).

Function Call	Type	GPT-3.5 S?	GPT-3.5 C?	GPT-4 S?	GPT-4 C?
False Positives of UBITect					
sscanf	KB, PE	✓	✓	✓	✓
read_mii_word	KB, PE	✗	✗	✓	✓
acpi_decode_pld_buffer	KB, PE	✓	✓	✓	✓
of_graph_get_remote_node	KB	✓	✓	✓	✓
msr_read	KB	✓	✓	✓	✓
cpuid	KB	✗	✗	✓	✓
bq2415x_i2c_read	KB	✓	✓	✓	✓
parse_nl_config	PE	✓	✗	✓	✓
snd_interval_refine	PE	✗	✗	✓	✓
xfs_iext_lookup_extent	PE	✓	✗	✓	✓
__skb_header_pointer	PE	✗	✗	✓	✓
snd_rawmidi_new	PE	✓	✗	✓	✓
snd_hwdep_new	PE	✗	✗	✓	✓
xdr_stream_decode...	PE	✓	✓	✓	✓
of_parse_phandle...	PE	✓	✓	✓	✓
kstrtoul	PE	✓	✓	✓	✓
False Negatives of UBITect					
pv_eoi_get_user	PE	✗	✗	✓	✓
p9pdu_readf	KB, PE	✗	✗	✗	✗

Limitation & Future work

1. Only UBI, but the principles should work on other bugs
2. Highly relies on GPT-4, need additional designs for weaker models

Conclusion

1. Can LLM assist in program analysis? Yes
2. How can LLM assist in program analysis? By asking program behavior
3. How can we limit the analysis scope? With progressive prompt

Thanks for Your Listening

GPTs on a break 😴: No AI assistance for this paper and presentation

Prompt Design: seclab-ucr/GPT-Expr

Learn more about our subsequent works: arXiv:2308.00245 [cs.SE]

UC RIVERSIDE