The Midas Touch: Triggering the Capability of LLMs for RM-API Misuse Detection

Yi Yang[1,2], Jinghua Liu[1,2], Kai Chen*[1,2], Miaoqian Lin[1,2]

1. School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

2. Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

中国科学院大学

University of Chinese Academy of Sciences

中国科学院信息工程研究所

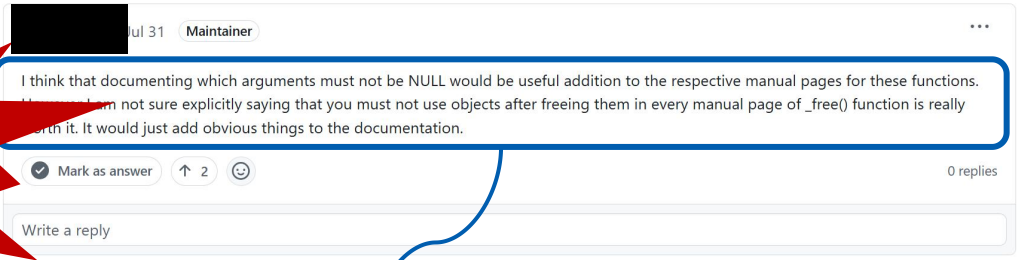
INSTITUTE OF INFORMATION ENGINEERING,CAS

What is RM-API?

Resource-management API (RM-API)

- ✓ Usually used in pairs
- ✓ Common sense to developers
- ✓ Omitted or unclear in documentation

Negative Attitude



However I am not sure explicitly saying that you must not use objects after freeing them in every manual page of _free() function is really worth it.

Violation of RM-APIs

Memory Corruption

Denial-of-Services(DoS)

Data Leakage

How to detect such misuses?

Code-analysis

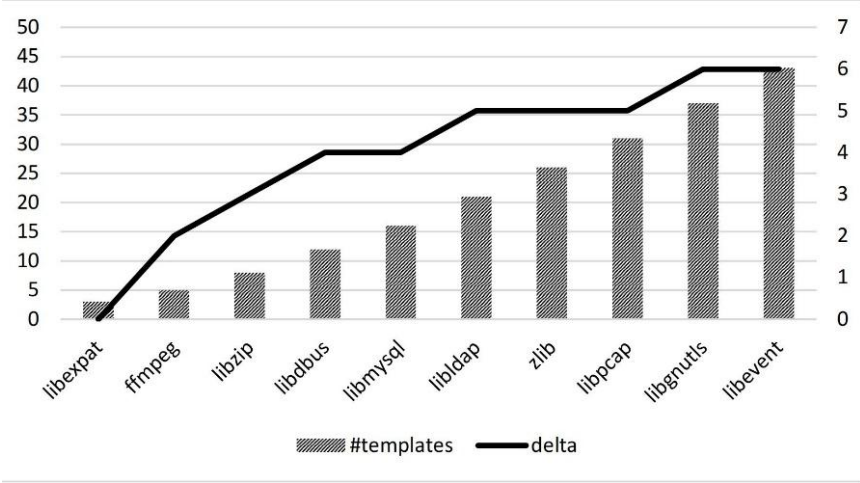
72.7% functions with multi-layered nesting **cannot precisely identified**

Keyword-matching

76.28% RM-APIs missing detected

Template-matching

Incapable of covering all the libraries



And recently

NLP-based Approach

- **Cannot handle constraints with neutral sentiment**
- **Incapable of detecting API pairs acrossing various sections/pages**

zip_source_buffer

The functions zip_source_buffer() and zip_source_buffer_create() **create a zip source from the buffer data of size len.**

zip_source_free

The function zip_source_free() decrements the reference count of source and **frees it if the reference count drops to 0**.

Neutral Sentiment

Various Locations

Eager for advanced tool on API document understanding!!

CHALLENGE

C1: LLMs fabricate answers without expertise

Q: Which API is used to free the malloc API 'evwatch_check_new'?

A: The API used for releasing the allocated memory 'evwatch_check_new' is 'evwatch_check_free'

- Consistent naming convention
- Seemingly reasonable operation
- Does not exist

→ Lack of information

CHALLENGE

C2: LLMs introduce incorrect answers with evidence

Q: Given the sentence of 'zip_discard': *The zip_discard function closes the archive and frees the memory allocated for it.* Does this API perform allocation?

releasing semantics

Contradictory!

allocation semantics

A: YES. The API zip_discard performs allocation.

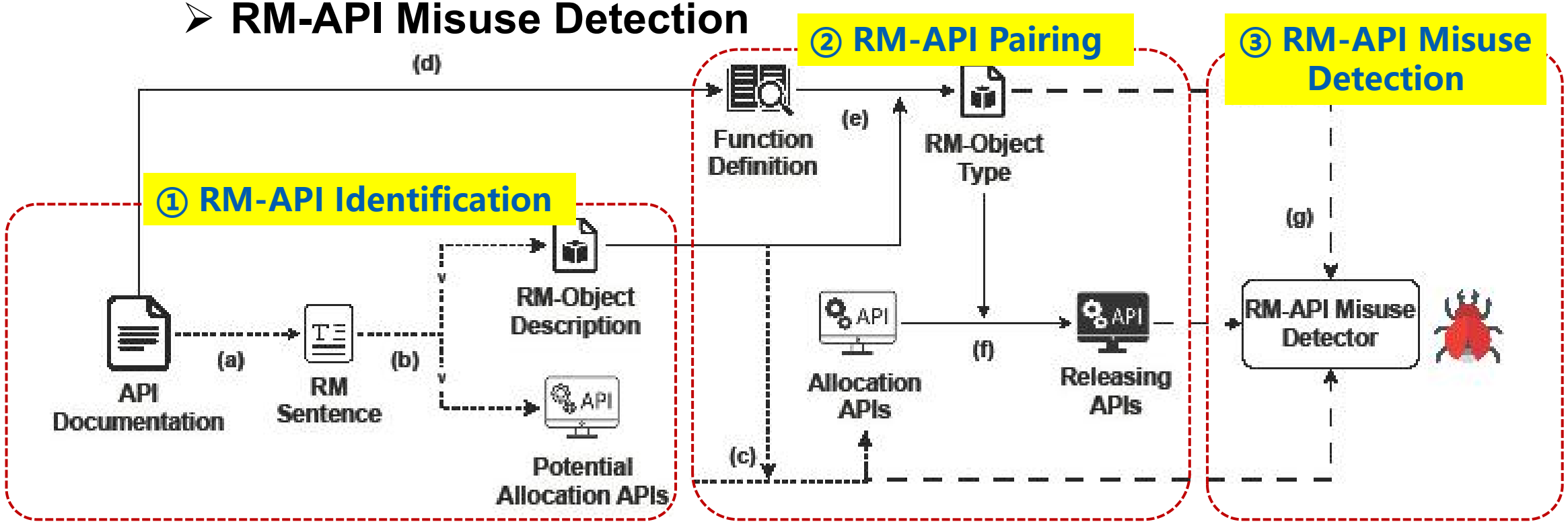
CHATDETECTOR

Complex task decomposition

- RM-API Identification
- RM-API Pairing
- RM-API Misuse Detection

External information

- API Description
- Function Definition



① RM-API Identification

★ Task Decomposition

- ✓ RM Sentence Identification
- ✓ Allocation API Identification

★ Two-dimensional Cross-validation

Given the description of the API '**{api}**' is: **{desc}.** Output the following answers in JSON format.

1. Is there a sentence describing the semantics of the API '{api}' related to allocation or releasing in the text?
   If YES, please output the first statement that meets the criteria;
   if NO, please output NULL.
2. Output the evidence sentence grounded for getting the result in '1'.
3. Output the reasoning process for getting the answers in '1'.

- ✓ Answer
- ✓ Evidence
- ✓ Reasoning process

RM Sentence Identification

LLM's Answer
- **Binary classification**
- **Open Question-Answering**

Is there a sentence describing the semantics of API'{api}' related to allocation or releasing in the text? If YES, ...; if NO, ...

RM Sentence

YES. The RM sentence is "**pcap_findalldevs() constructs a list of network devices that can be opened with pcap_create(3PCAP) "**

PCAP_FINDALLDEVS(3PCAP) MAN PAGE

Updated: 9 August 2024 • View in plain text • Return to Main Contents

This man page documents libpcap version 1.11.0-PRE-GIT (see also: 1.10.5, 1.10.4, 1.10.2, 1.10.1, 1.10.0, 1.9.1, 1.8.1, 1.7.4, 1.6.2, 1.5.3).

Your system may have a different version installed, possibly with some local modifications. To achieve the best results, please make sure this version of this man page suits your needs. If necessary, try to look for a different version on this web site or in the man pages available in your installation.

NAME

pcap_findalldevs, pcap_freealldevs - get a list of capture devices, and free that list

SYNOPSIS

```
#include <pcap/pcap.h>

char errbuf[PCAP_ERRBUF_SIZE];

int pcap_findalldevs(pcap_if_t **alldevsp, char *errbuf);
void pcap_freealldevs(pcap_if_t *alldevs);
```

DESCRIPTION

pcap_findalldevs() constructs a list of network devices that can be opened with **pcap_create**(3PCAP) and **pcap_activate**(3PCAP) or with **pcap_open_live**(3PCAP). (Note that there may be network devices that cannot be opened by the process calling **pcap_findalldevs**(), because, for example, that process does not have sufficient privileges to open them for capturing; if so, those devices will not appear on the list.) *alldevsp* is a pointer to a `pcap_if_t *`; *errbuf* is a buffer large enough to hold at least `PCAP_ERRBUF_SIZE` chars.

If **pcap_findalldevs**() succeeds, the pointer pointed to by *alldevsp* is set to point to the first element of the list, or to `NULL` if no devices were found (this is considered success). Each element of the list is of type `pcap_if_t`, and has the following members:

`next`
: if not `NULL`, a pointer to the next element in the list; `NULL` for the last element of the list

`name`
: pointer to a string giving a name for the device to pass to **pcap_open_live**()

`description`
: if not `NULL`, a pointer to a string giving a human-readable description of the device

`addresses`
: a pointer to the first element of a list of network addresses for the device, or `NULL` if the device has no addresses

`flags`
: device flags:

`PCAP_IF_LOOPBACK`
: set if the device is a loopback interface

`PCAP_IF_UP`
: set if the device is up

Documentation

RM Sentence Identification

LLM's Evidence
- **In-context answering**
- **Retrieve information from context**

Given the description of the API:

Output the evidence sentence grounded for getting the result in '1'.

Evidence

The evidence is "pcap_findalldevs() **constructs** a list of network devices that can be opened with pcap_create(3PCAP) "

Consistent with RM Sentence

PCAP_FINDALLDEVS(3PCAP) MAN PAGE

Updated: 9 August 2024 • View in plain text • Return to Main Contents

This man page documents libpcap version 1.11.0-PRE-GIT (see also: 1.10.5, 1.10.4, 1.10.2, 1.10.1, 1.10.0, 1.9.1, 1.8.1, 1.7.4, 1.6.2, 1.5.3).

Your system may have a different version installed, possibly with some local modifications. To achieve the best results, please make sure this version of this man page suits your needs. If necessary, try to look for a different version on this web site or in the man pages available in your installation.

NAME

pcap_findalldevs, pcap_freealldevs - get a list of capture devices, and free that list

SYNOPSIS

```
#include <pcap/pcap.h>

char errbuf[PCAP_ERRBUF_SIZE];

int pcap_findalldevs(pcap_if_t **alldevsp, char *errbuf);
void pcap_freealldevs(pcap_if_t *alldevs);
```

DESCRIPTION

pcap_findalldevs() constructs a list of network devices that can be opened with **pcap_create**(3PCAP) and **pcap_activate**(3PCAP) or with **pcap_open_live**(3PCAP). (Note that there may be network devices that cannot be opened by the process calling **pcap_findalldevs**(), because, for example, that process does not have sufficient privileges to open them for capturing; if so, those devices will not appear on the list.) *alldevsp* is a pointer to a `pcap_if_t *`; *errbuf* is a buffer large enough to hold at least `PCAP_ERRBUF_SIZE` chars.

If **pcap_findalldevs**() succeeds, the pointer pointed to by *alldevsp* is set to point to the first element of the list, or to `NULL` if no devices were found (this is considered success). Each element of the list is of type `pcap_if_t`, and has the following members:

`next`
: if not `NULL`, a pointer to the next element in the list; `NULL` for the last element of the list

`name`
: a pointer to a string giving a name for the device to pass to **pcap_open_live**()

`description`
: if not `NULL`, a pointer to a string giving a human-readable description of the device

`addresses`
: a pointer to the first element of a list of network addresses for the device, or `NULL` if the device has no addresses

`flags`
: device flags:

`PCAP_IF_LOOPBACK`
: set if the device is a loopback interface

`PCAP_IF_UP`
: set if the device is up

Documentation

Allocation API Identification

LLM's Reasoning Process
- Open QA
- The logic for getting the answer

Given the RM sentence of API:

1. Does the API performs allocation? Answer YES/NO.

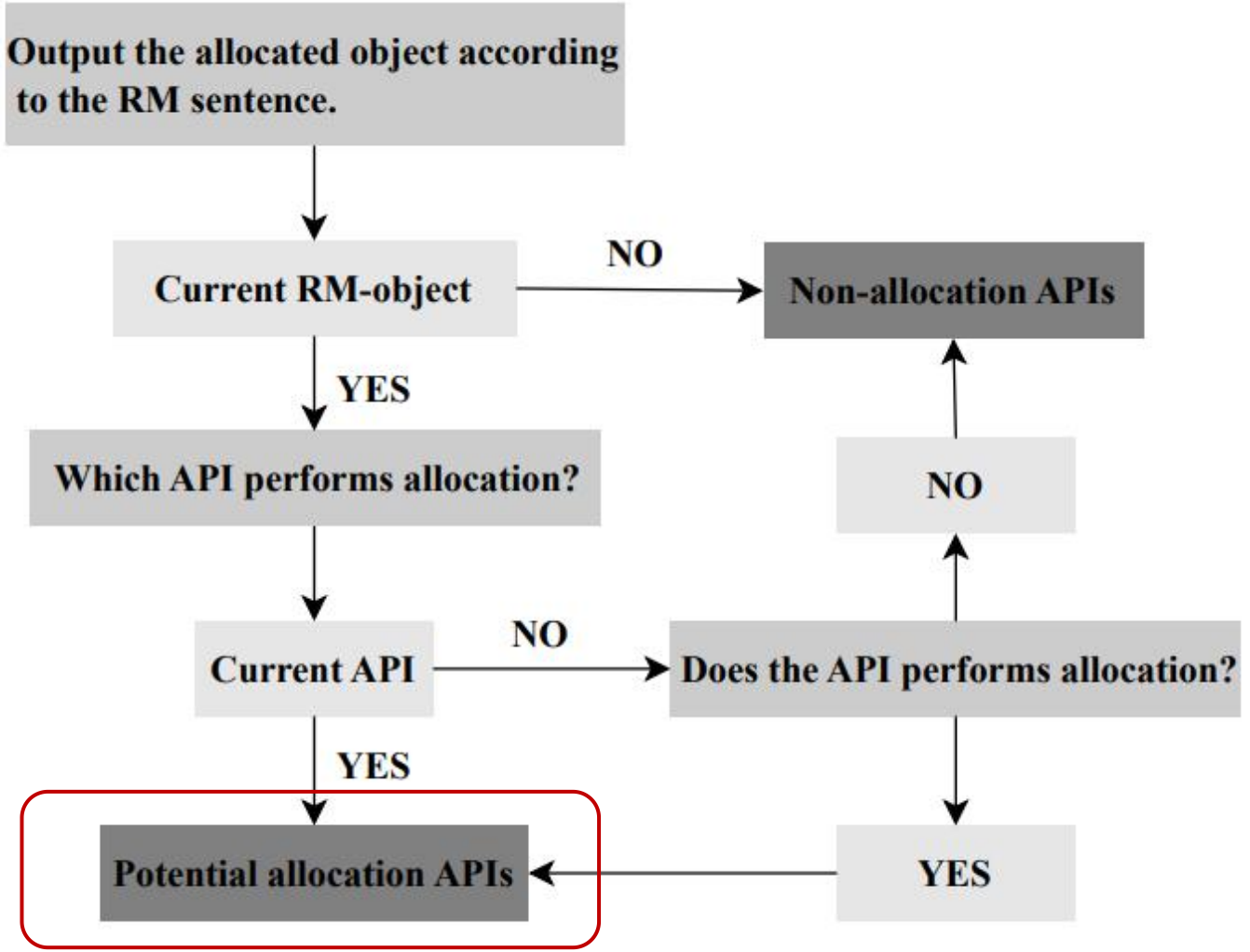
2. Present the evidence for getting the answer in '1'.

3. Present your reasoning process for getting the answer in '1'.

{ "1": **"YES"**,
"2": "The sentence indicates that pcap_findalldevs() constructs a list of network devices, **which implies memory allocation for this list.**",
"3": "The verb 'constructs' suggests that pcap_findalldevs() is building or creating a data structure (in this case, a list of network devices). **Construction of a list typically involves allocating memory for storing the list items, which is a form of allocation.**"}

Two-dimensional Cross-validation

Rules for Cross-validation

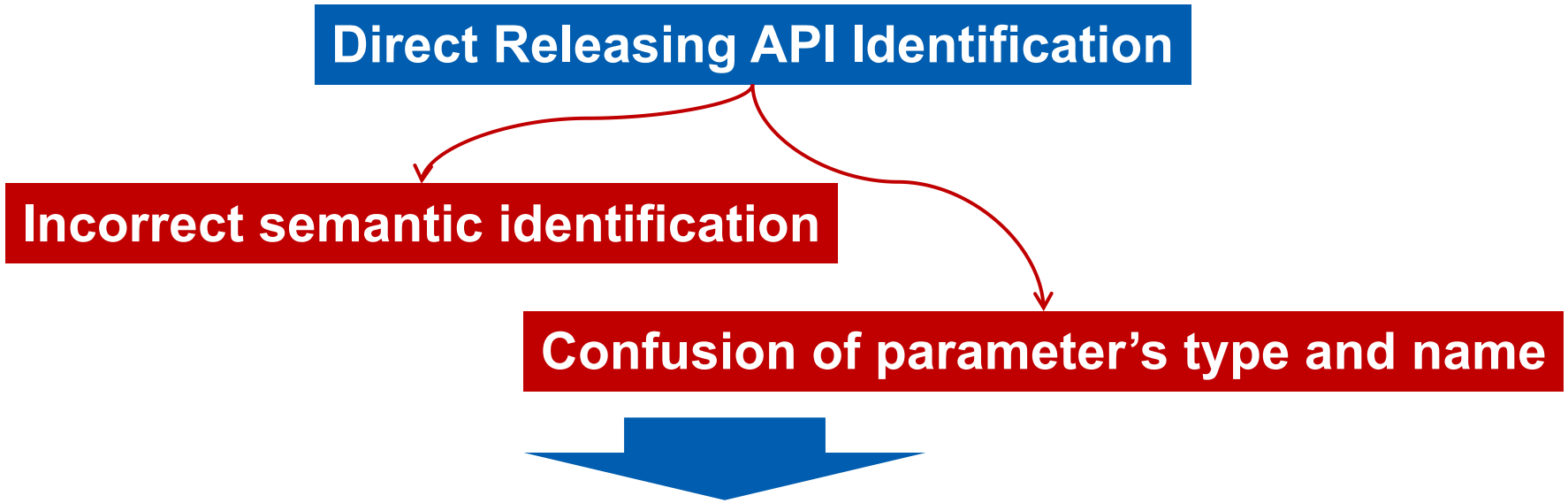


Does the RM Object exist?

Is the output of LLM consistent with the current API?

Is the current API an allocation API?

② RM-API Pairing



Off-the-shelf NLP tool

- ✓ Function pre-process
- ✓ Semantics classification

② RM-API Pairing

★ Task Decomposition

- ✓ RM-object Identification
- ✓ Releasing API Identification

Function Pre-process

Before

```
int pcap_findalldevs
( pcap_if_t **alldevsp,
char *errbuf);
```

After

‘0’: int pcap_findalldevs
‘1’: pcap_if_t **alldevsp
‘2’: char *errbuf

② RM-API Pairing

RM-object Identification

Given the allocated object description of API and the function definition:

1. According to the function definition, output the object type.

{ "1": "pcap_if_t "}

Releasing API Identification

Given the allocated object type of the API according to the function definition:

Which API with the same object type should be used to release the allocate memory of API.

{ "1": "pcap_freealldevs"}

③ RM-API Misuse Detection

- ◆ CodeQL
- ✓ Manual-construct QL code
- ✓ 3 types of security issues

```
1  predicate isSourceFC(FunctionCall fc)
2  {
3  // malloc function: ev
4  fc.getTarget().hasName
5  }
6
7  Expr getSourceExpr(FunctionCall fc)
8  {
9  //parameter 1:
10 result = fc.getArgument(0)
11 //return value:
12 //result = fc
13 }
14
15 import cpp
16 from BasicBlock bb, FunctionCall malloc
17 where
18   // locate the malloc function call by allocation
         API name
19   isSourceFC(malloc)
20   //Make sure that the malloc function operates on
       a local variable
21   and isLocalSource(malloc)
22   //There is a path in the current malloc function
       postpath that does not have a releasing
       operation
23   and bb = getLeakBlock(malloc)
24 select malloc, malloc.getLocation().toString()
```

pcap_findalldevs

- ✓ Memory Leak
- ✓ Use-after-free
- ✓ Double-free

- ➢ Denial-of-Services (DoS)
- ➢ Memory Corruption

```
2535        if (interface == NULL)
2536          {
2537            if (pcap_findalldevs (&alldevsp, errbuf) < 0)
2538              g_message ("Error for pcap_findalldevs(): %s", errbuf);
2539            if (alldevsp != NULL)
2540              interface = alldevsp->name;
2541          }
2542      }
2543
2544    if (interface != NULL)
2545      bpf = bpf_open_live (interface, filter);
2546
2547    if (bpf < 0)
2548      {
2549        nasl_perror (lexic, "pcap_next: Could not get a bpf\n");
2550        return NULL;
2551
```

Missing the releasing operation for ‘&alldevsp, leading to memory leak

Example

av_dict_set()

*int av_dict_set (AVDictionary **pm, const char *key, const char * value, int flags)*

==========

Set the given entry in *pm, overwriting an existing entry.

Note: If AV_DICT_DONT_STRDUP_KEY or AV_DICT_DONT_STRDUP_VAL is set, these arguments will be freed on error.

Parameters

pm pointer to a pointer to a dictionary struct. **If *pm is NULL a dictionary struct is allocated and put in *pm.**

...

Returns

>= 0 on success otherwise an error code <0

Documentation

```
int av_dict_set(AVDictionary **pm, const char *key, const char *
value)
  {
    AVDictionary *m = *pm;
    AVDictionaryEntry *tag = NULL;
    char *copy_key = NULL, *copy_value = NULL;
    int err;
    if(!key){
        err = AVERROR(EINVAL);
        goto err_out;}
    err_out:
        if (m&&!m->count){
            av_freep(&m-elems);
            av_freep(pm);
        }
        av_free(copy_key);
        av_free(copy_value);
        return err;
  }
```

Source code

Detected misuse in code

```
int encavcodecInit( hb_work_object_t * w, hb_job_t * job )
{
    int ret = 0;
    char reason[80];
    ...
    if (job->vcodec == HB_VCODEC_FFMPEG_VCE_H265 || job->vcodec ==
HB_VCODEC_FFMPEG_VCE_H265_10BIT)
            {
                av_dict_set( &av_opts, "qmin",  "0", 0 );
                av_dict_set( &av_opts, "qmax", "51", 0 );
            }
    ...
    ...
    if (hb_avcodec_open(context, codec, &av_opts,
HB_FFMPEG_THREADS_AUTO))
    {
        hb_log( "encavcodecInit: avcodec_open failed" );
        ret = 1;
        goto done;
    }
    done:
    return ret;
}
```

Missing av_dict_free

21 memory leak bugs
found in popular software

Example

Detected incorrect documentation

av_get_token()

==========

Unescape the given string until a non escaped terminating char, and return the token corresponding to the unescaped string.

Returns

the malloced unescaped string, which must be av_freed by the user, NULL in case of allocation failure

- Incorrect releasing API

- Result in FP by previous work

PERFORMANCE

6 popular libraries

165 detected RM-API pairs

115 security bugs

Compared with manual work

- **47%** more RM sentences

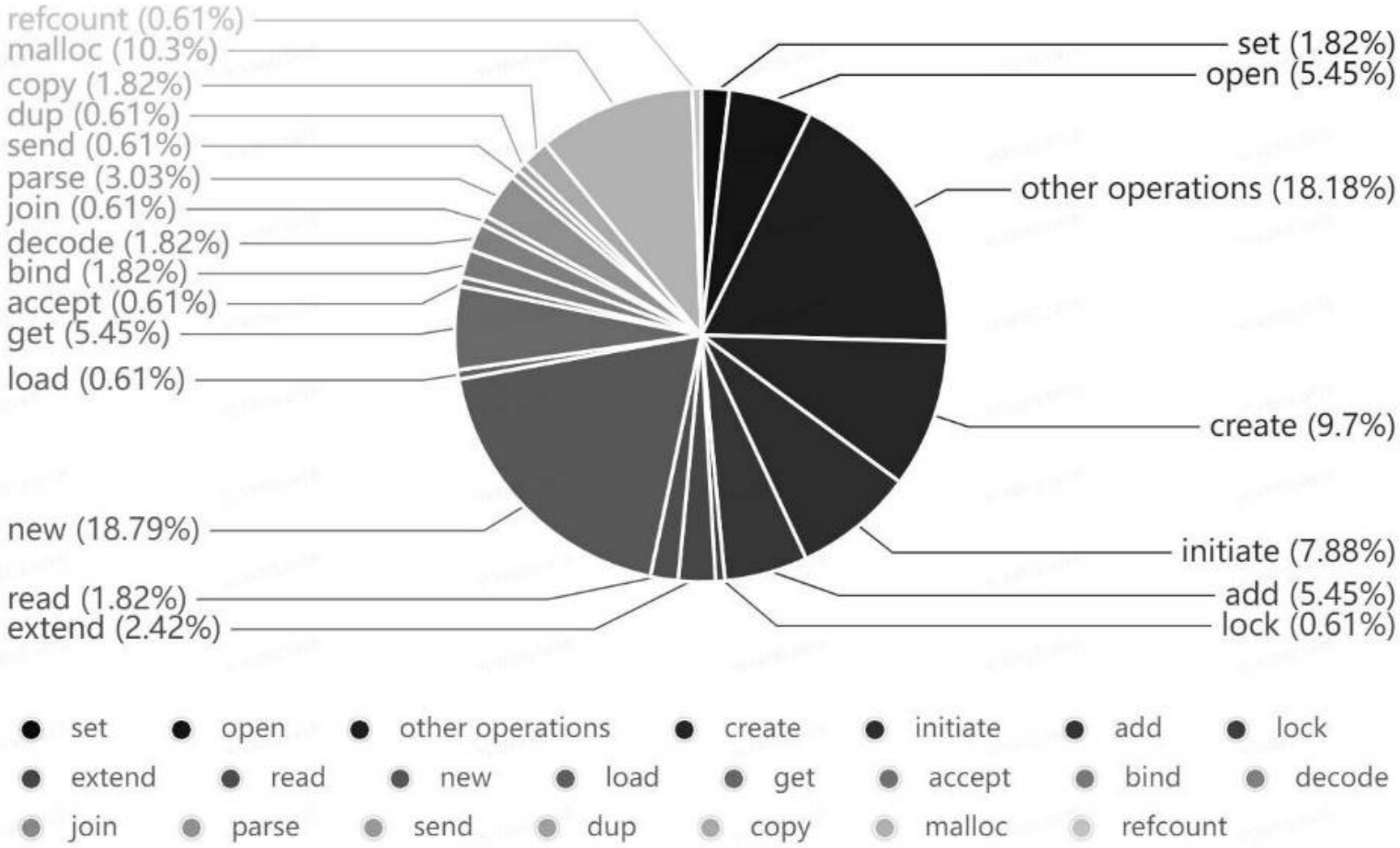
- **80.85%** more RM-API pairs

TABLE VI: Discovered RM-API misuses

Lib	Software	Allocation API	Releasing API	Bug Type	#Bugs
libevent	seafile	event_base_new	event_base_free	memleak	1
	evpp	event_base_new	event_base_free	memleak	2
		event_config_new	event_config_free	memleak	1
		event_new	event_free	memleak	2
		evhttp_new	evhttp_free	memleak	1
	transmission	event_new	event_free	memleak	1
libzip	radare2	zip_source_buffer_create	zip_source_free	memleak	1
	OpenRCT2	zip_source_buffer	zip_source_free	memleak	1
ffmpeg	gpac	av_dict_set	av_dict_free	memleak	2
		av_dict_copy	av_dict_free	memleak	1
	HandBrake	av_dict_set	av_dict_free	memleak	21
		avfilter_graph_create_filter	avfilter_free	memleak	1
		av_buffersrc_parameters_alloc	av_free	double free	1
	FFmpeg	av_frame_new_side_data	av_frame_remove_side_data	memory leak	29
		av_frame_alloc	av_frame_free	double free	1
		av_new_program	av_free	memory leak	8
	owntone-server	av_dict_set	av_dict_free	memleak	3
	vlc	av_malloc	av_freep	memleak	1
libpcap	PF_RING	pcap_findalldevs	pcap_freealldevs	memleak	2
		pcap_compile	pcap_freecode	memleak	4
	arp-scan	pcap_compile	pcap_freecode	memleak	1
	freeradius-server	pcap_compile	pcap_freecode	memleak	1
	nmap	pcap_compile	pcap_freecode	memleak	1
	ntopng	pcap_compile	pcap_freecode	memleak	2
	tcpdump	pcap_compile	pcap_freecode	memleak	1
	wireshark	pcap_compile	pcap_freecode	memleak	2
	openvas-scanner	pcap_compile	pcap_freecode	memleak	1
		pcap_findalldevs	pcap_freealldevs	memleak	2
ldap	freebsd-src	ldap_search_s	ldap_msgfree	memleak	3
	freeradius-server	ldap_result	ldap_msgfree	double free	2
		ldap_result	ldap_msgfree	memleak	1
	gpdb	ldap_search_s	ldap_msgfree	memleak	1
	openldap	ldap_search_ext_s	ldap_msgfree	memleak	6
		ldap_url_parse	ldap_free_urldesc	memleak	2
		ldap_initialize	ldap_unbind_ext	use after free	1
		ldap_initialize	ldap_unbind_ext	memleak	2
	yugabyte-db	ldap_search_s	ldap_msgfree	memleak	1
	postgres	ldap_search_s	ldap_msgfree	memleak	1
Total	-	-	-	-	115

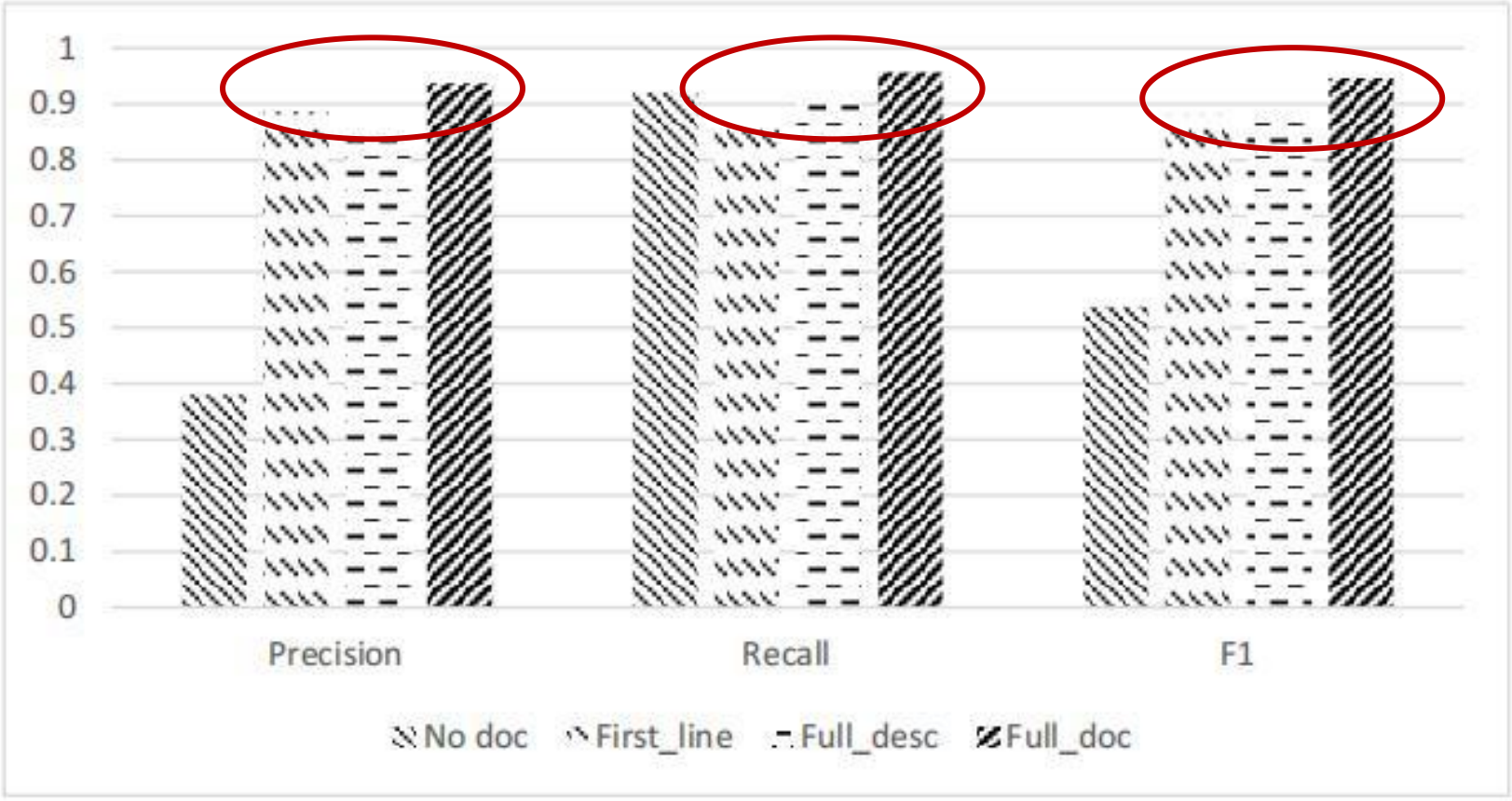
Various types

22 types of RM-APIs



Stable performance

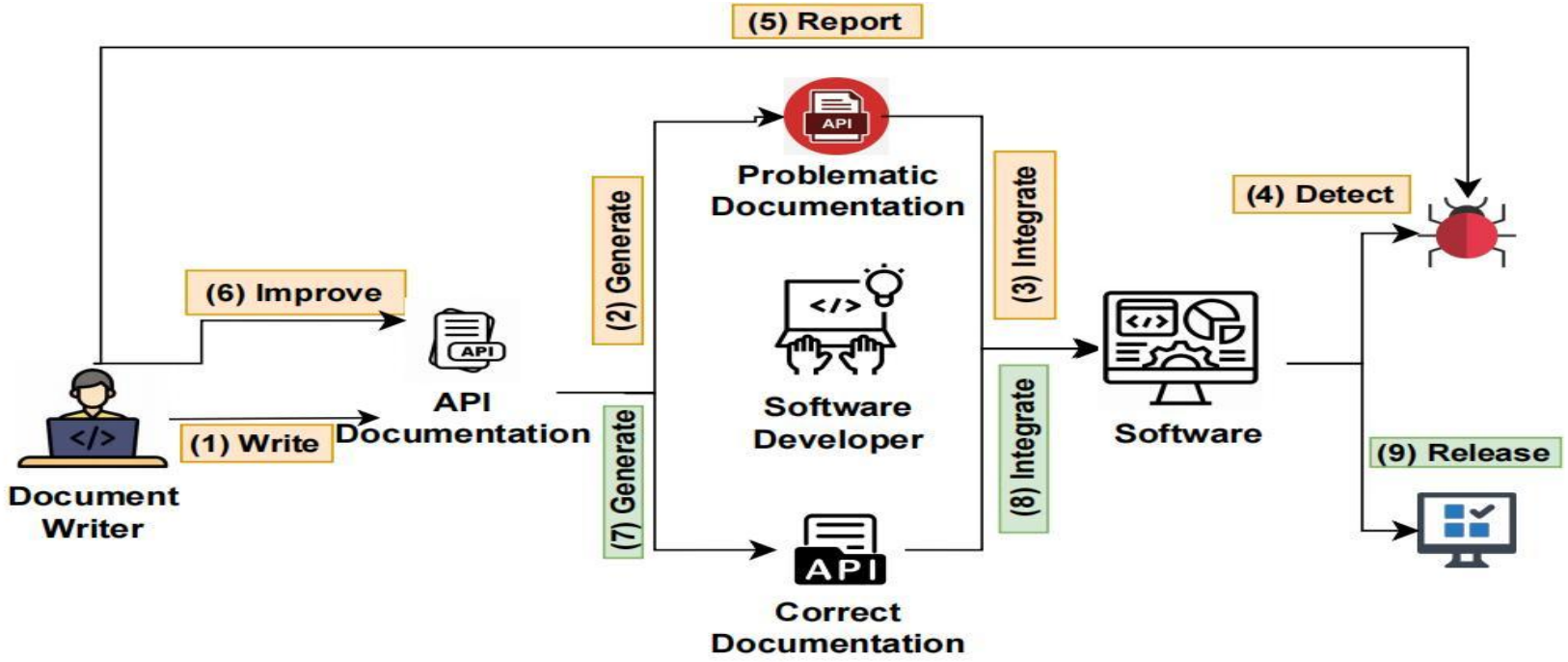
Robust to document quality



FUTURE DIRECTION

Bottleneck of traditional NLP tools on document understanding

Triggering the capability of LLMs for security research



Thank You
For Your Attention

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS

yangyi@iie.ac.cn