Detecting API Post-Handling Bugs Using Code and Description in Patches

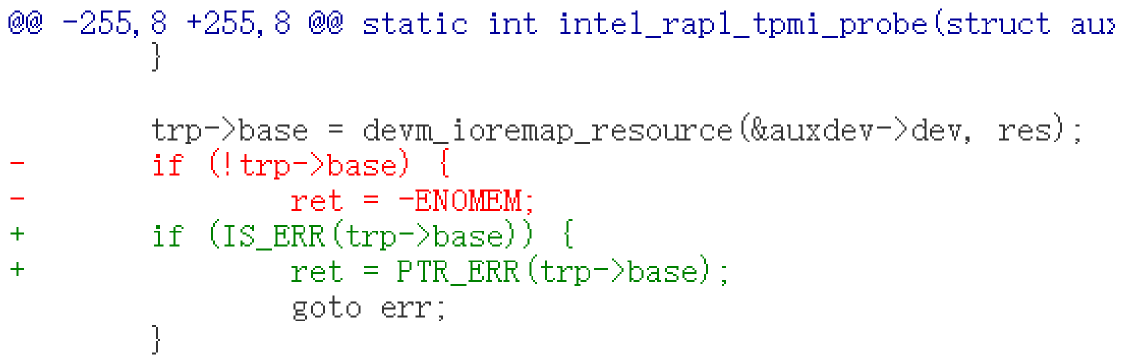
Miaoqian Lin, Kai Chen, Yang Xiao

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

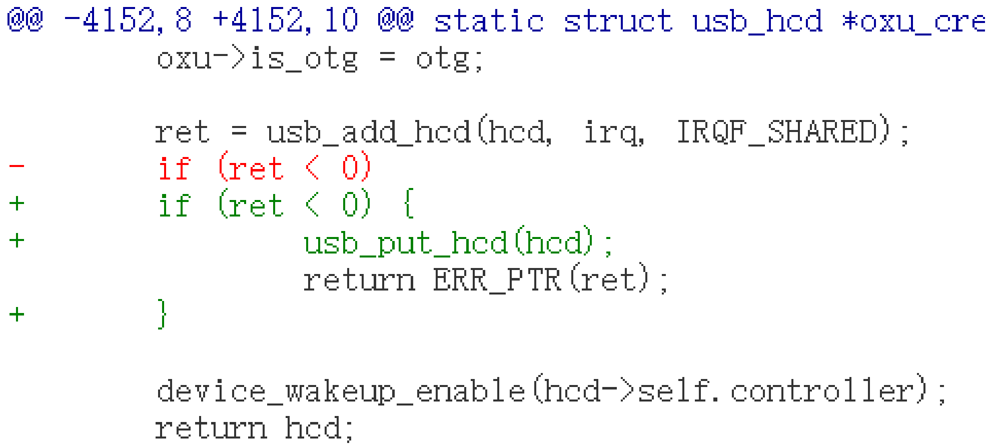
API Post-Handling (APH)

API Call

API Post-handling ✗



```
@@ -255,8 +255,8 @@ static int intel_rapl_tpmi_probe(struct aux
        }

        trp->base = devm_ioremap_resource(&auxdev->dev, res);
-       if (!trp->base) {
-               ret = -ENOMEM;
+       if (IS_ERR(trp->base)) {
+               ret = PTR_ERR(trp->base);
                goto err;
        }
```



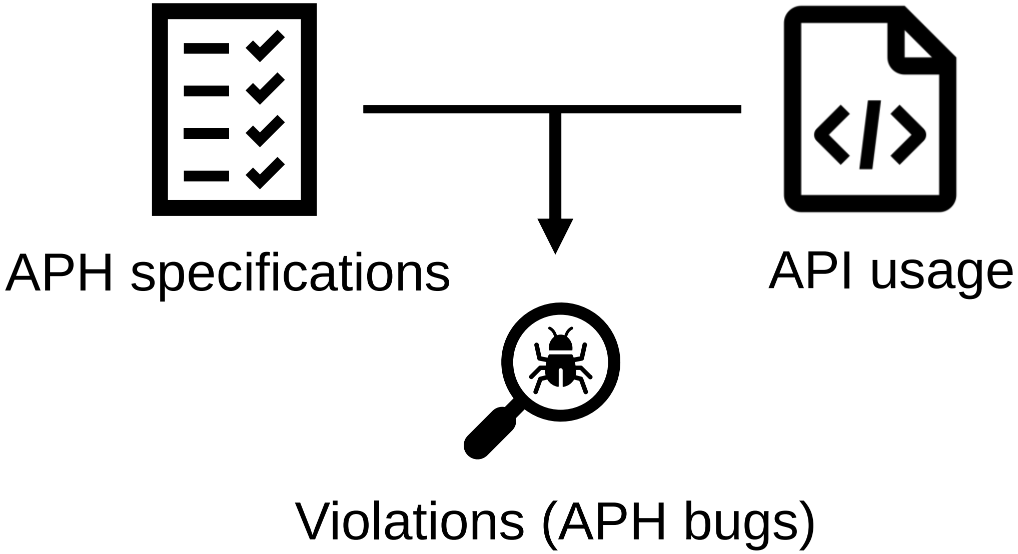
```
@@ -4152,8 +4152,10 @@ static struct usb_hcd *oxu_cre
        oxu->is_otg = otg;

        ret = usb_add_hcd(hcd, irq, IRQF_SHARED);
-       if (ret < 0)
+       if (ret < 0) {
+               usb_put_hcd(hcd);
                return ERR_PTR(ret);
+       }

        device_wakeup_enable(hcd->self.controller);
        return hcd;
```

API Post-handling is error-prone and detecting APH bugs is vital

How to detect?

Document of **kobject_init_and_add** in Linux kernel:

“If this function returns an error, kobject_put() must be called to properly clean up the memory associated with the object”



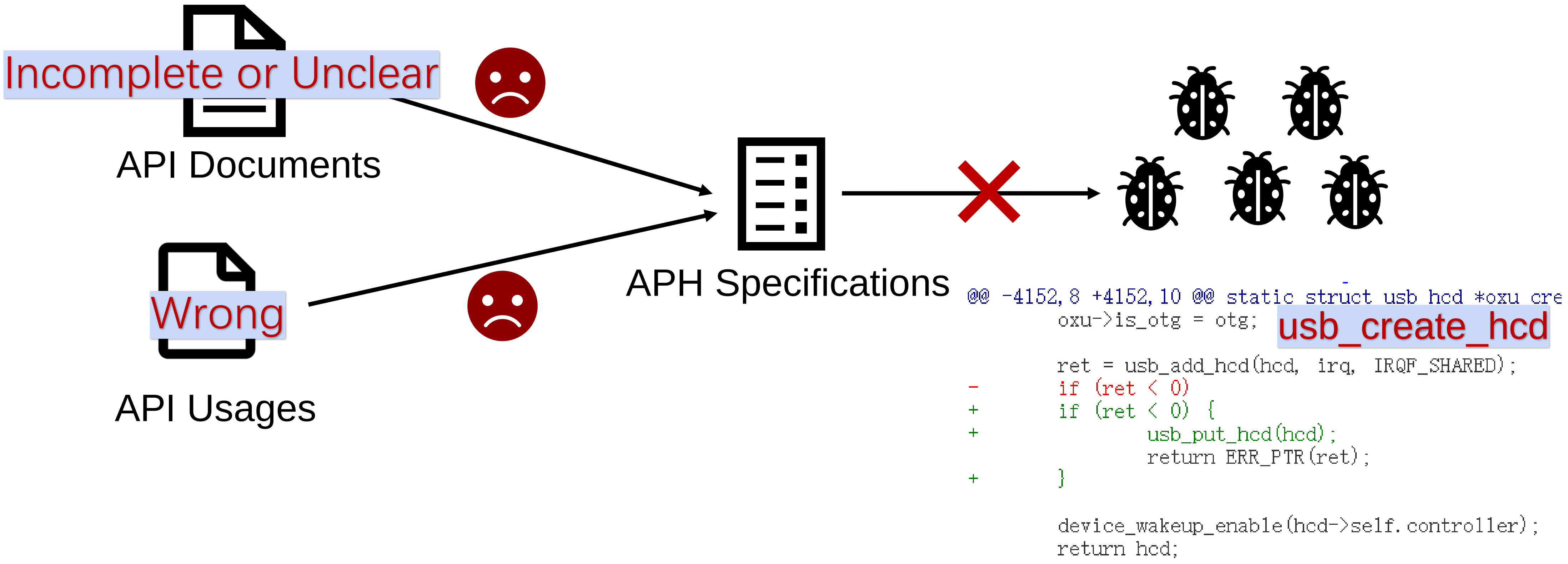
APH specifications

API usage

Violations (APH bugs)

APH Specifications are the key for detecting APH bugs

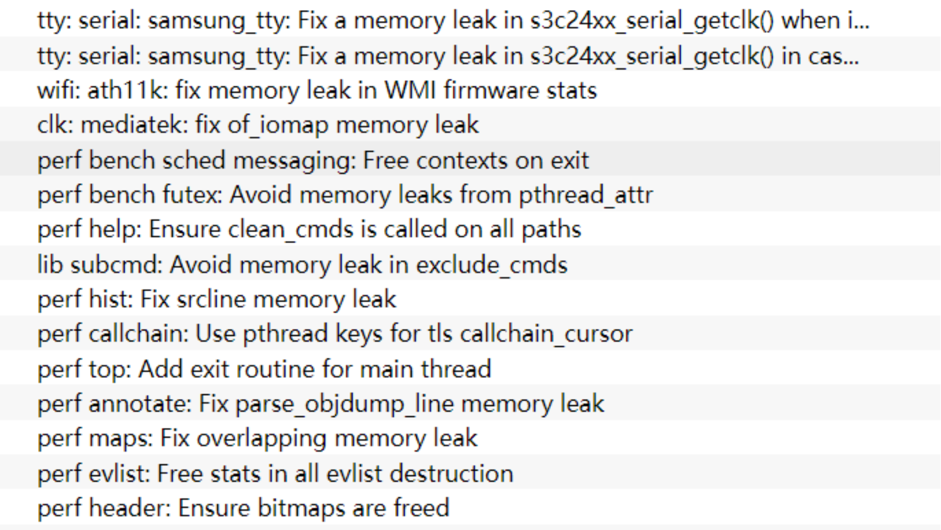
Limitations of Previous Work



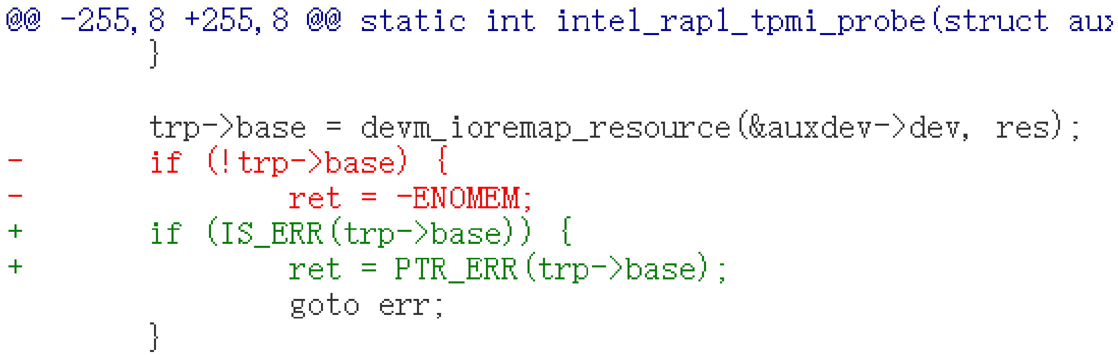
```
@@ -4152,8 +4152,10 @@ static struct usb hcd *oxu cre
         oxu->is_otg = otg;

         ret = usb_add_hcd(hcd, irq, IRQF_SHARED);
-        if (ret < 0)
+        if (ret < 0) {
+                usb_put_hcd(hcd);
                 return ERR_PTR(ret);
+        }

         device_wakeup_enable(hcd->self.controller);
         return hcd;
```

Failure to extract specifications leads to **uncovered bugs!**

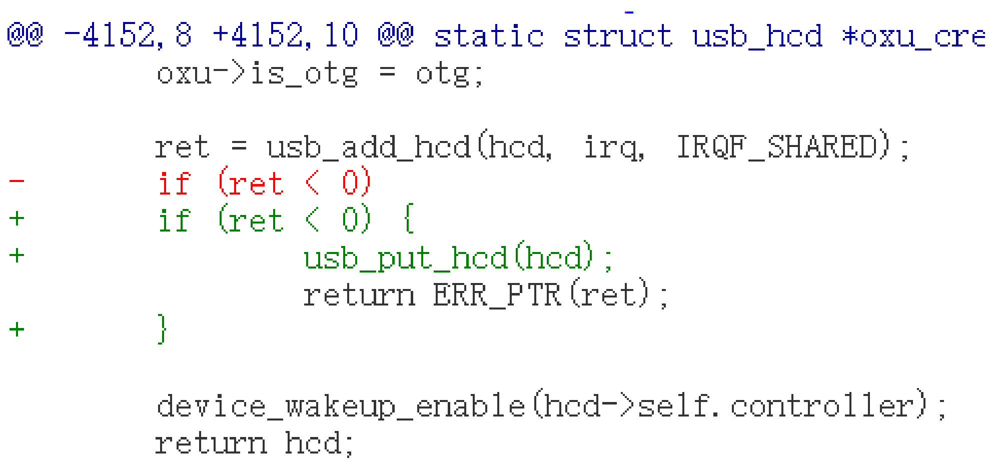
APH Bug Patches



tty: serial: samsung_tty: Fix a memory leak in s3c24xx_serial_getclk() when i...
tty: serial: samsung_tty: Fix a memory leak in s3c24xx_serial_getclk() in cas...
wifi: ath11k: fix memory leak in WMI firmware stats
clk: mediatek: fix of_iomap memory leak
perf bench sched messaging: Free contexts on exit
perf bench futex: Avoid memory leaks from pthread_attr
perf help: Ensure clean_cmds is called on all paths
lib subcmd: Avoid memory leak in exclude_cmds
perf hist: Fix srcline memory leak
perf callchain: Use pthread keys for tls callchain_cursor
perf top: Add exit routine for main thread
perf annotate: Fix parse_objdump_line memory leak
perf maps: Fix overlapping memory leak
perf evlist: Free stats in all evlist destruction
perf header: Ensure bitmaps are freed



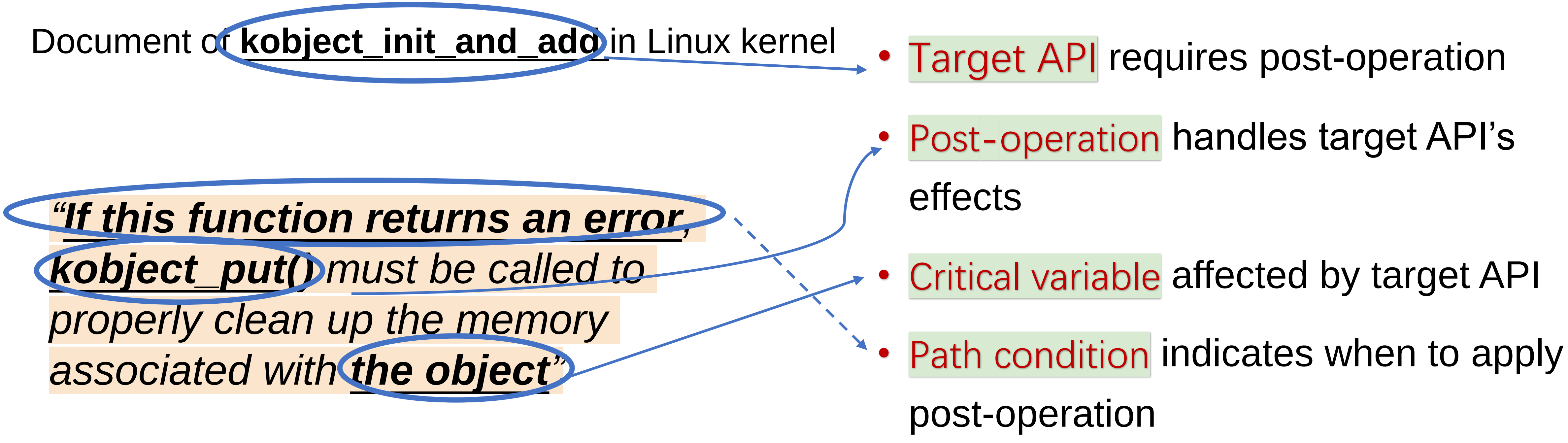
```
@@ -255,8 +255,8 @@ static int intel_rapl_tpmi_probe(struct au
        }

        trp->base = devm_ioremap_resource(&auxdev->dev, res);
-       if (!trp->base) {
-               ret = -ENOMEM;
+       if (IS_ERR(trp->base)) {
+               ret = PTR_ERR(trp->base);
                goto err;
        }
```



```
@@ -4152,8 +4152,10 @@ static struct usb_hcd *oxu_cre
        oxu->is_otg = otg;

        ret = usb_add_hcd(hcd, irq, IRQF_SHARED);
-       if (ret < 0)
+       if (ret < 0) {
+               usb_put_hcd(hcd);
                return ERR_PTR(ret);
+       }

        device_wakeup_enable(hcd->self.controller);
        return hcd;
```

APH bug patches are good source for APH specifications

Insights



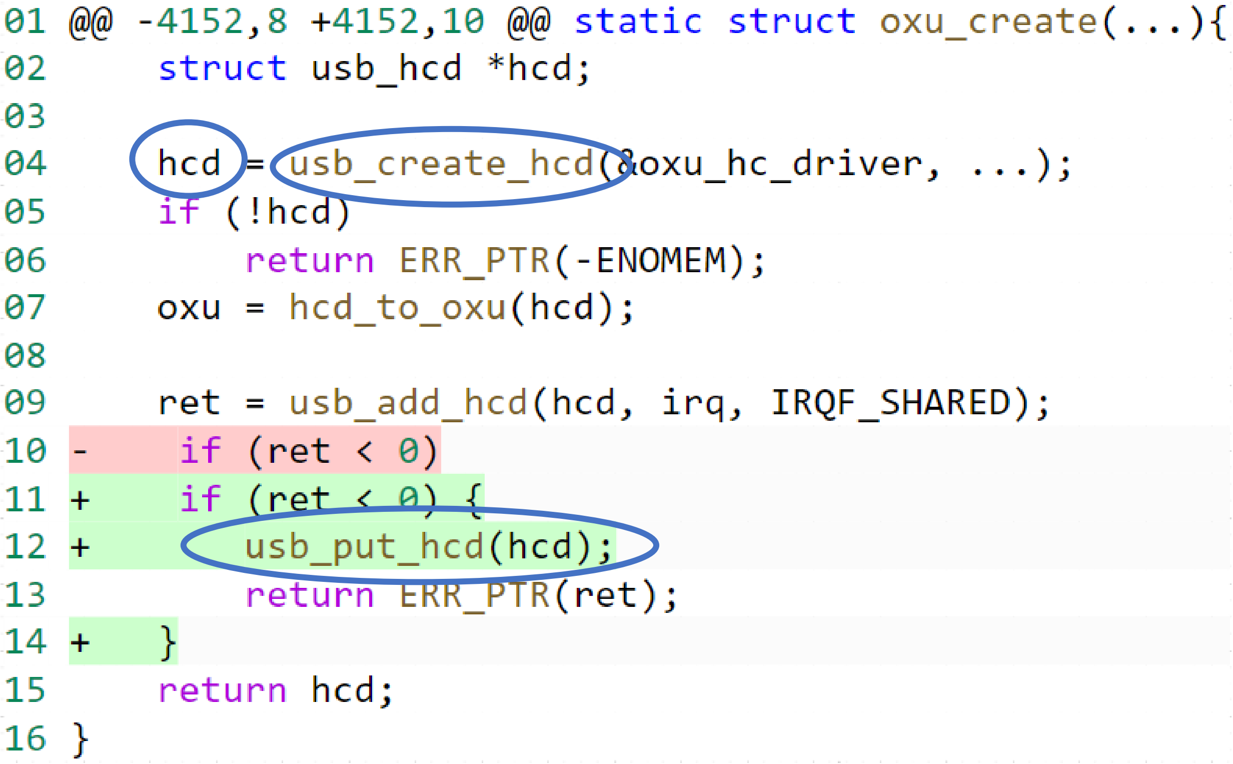
Document of **kobject_init_and_add** in Linux kernel

*“**If this function returns an error, kobject_put()** must be called to properly clean up the memory associated with **the object**”*

- Target API requires post-operation
- Post-operation handles target API’s effects
- Critical variable affected by target API
- Path condition indicates when to apply post-operation

☺ Define APH specifications as **four-tuples** with key elements

Motivating Example



```
@@ -4152,8 +4152,10 @@ static struct oxu_create(...){
    struct usb_hcd *hcd;

    hcd = usb_create_hcd(&oxu_hc_driver, ...);
    if (!hcd)
        return ERR_PTR(-ENOMEM);
    oxu = hcd_to_oxu(hcd);

    ret = usb_add_hcd(hcd, irq, IRQF_SHARED);
-    if (ret < 0)
+    if (ret < 0) {
+        usb_put_hcd(hcd);
        return ERR_PTR(ret);
+   }
    return hcd;
}
```

Target API: usb_create_hcd

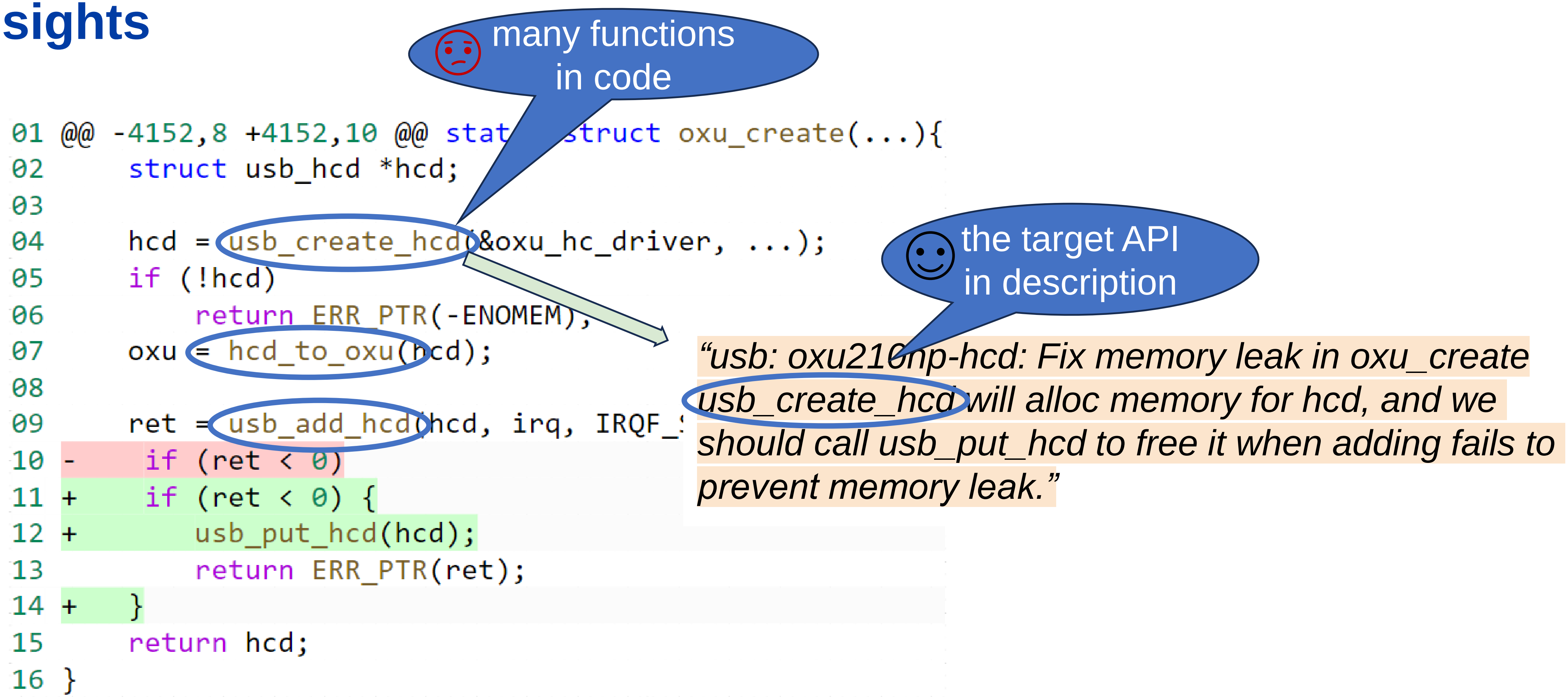
Critical variable: hcd

Post-operation: usb_put_hcd

Path conditions

Patch contains key elements defined in APH specification

Insights



many functions in code

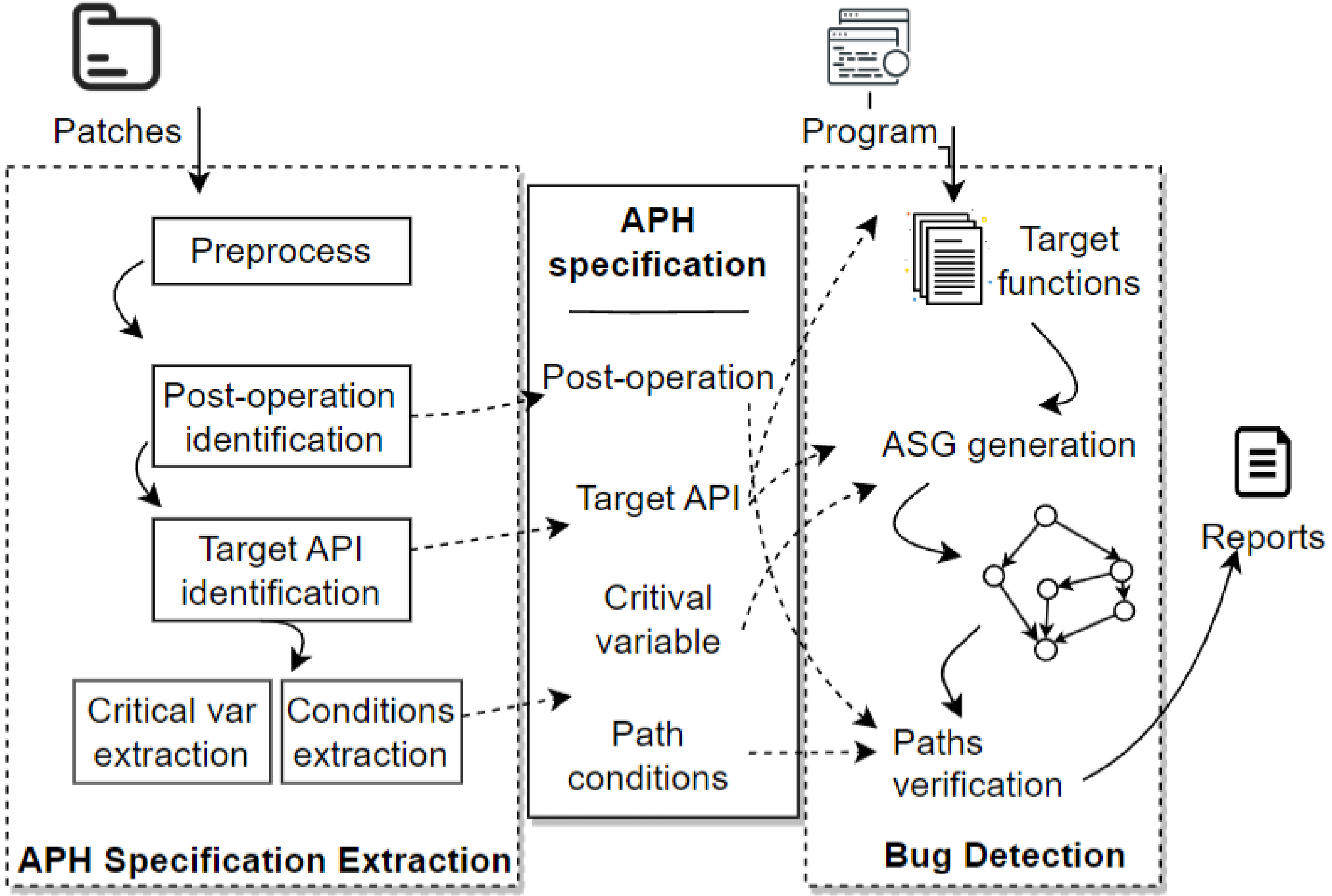
```
@@ -4152,8 +4152,10 @@ stat      struct oxu_create(...){
     struct usb_hcd *hcd;

     hcd = usb_create_hcd(&oxu_hc_driver, ...);
     if (!hcd)
         return ERR_PTR(-ENOMEM);
     oxu = hcd_to_oxu(hcd);

     ret = usb_add_hcd(hcd, irq, IRQF_
-     if (ret < 0)
+     if (ret < 0) {
+         usb_put_hcd(hcd);
         return ERR_PTR(ret);
+     }
     return hcd;
}
```

the target API in description

"usb: oxu210hp-hcd: Fix memory leak in oxu_create usb_create_hcd will alloc memory for hcd, and we should call usb_put_hcd to free it when adding fails to prevent memory leak."

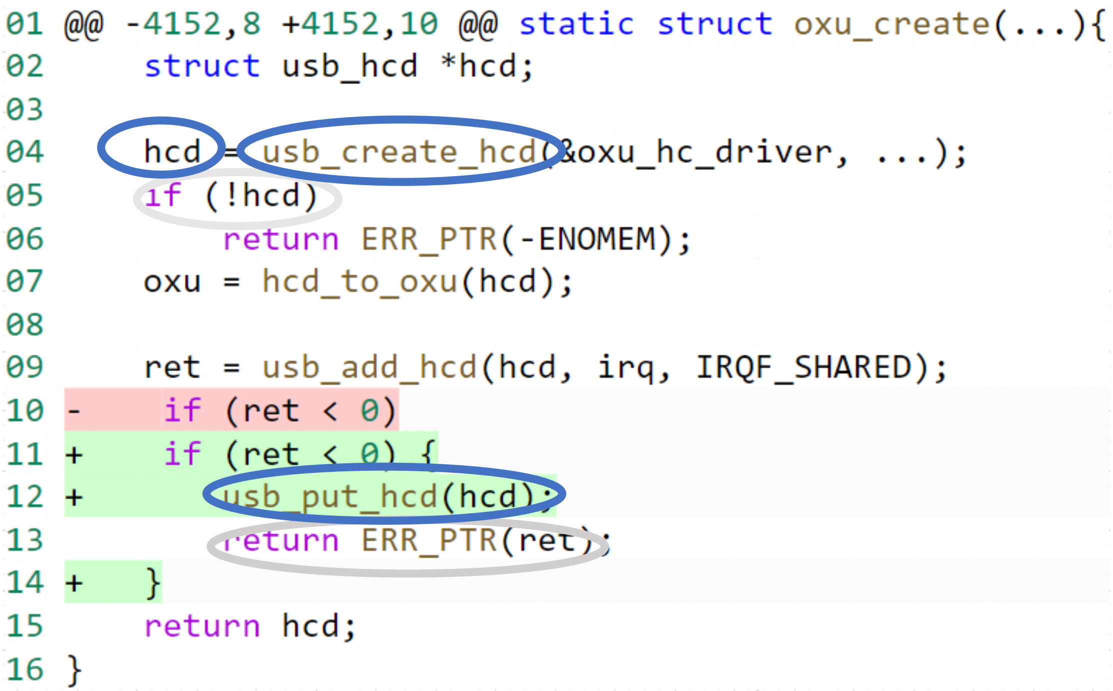
Extract specifications using **code and description** in patches

Overview of APHP: APH bugs detector using patches

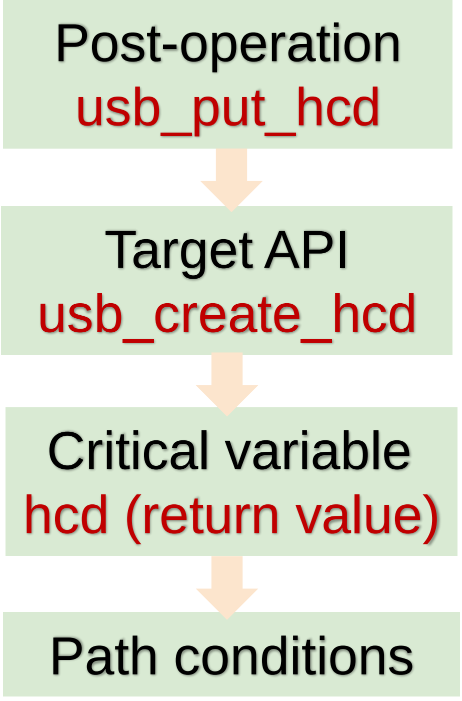


Specification Extraction: Using code and description

“usb: oxu210hp-hcd: Fix memory leak … usb_create_hcd will alloc memory for hcd, and we should call usb_put_hcd to free it when adding fails to prevent memory leak.”



```
@@ -4152,8 +4152,10 @@ static struct oxu_create(...){
     struct usb_hcd *hcd;

     hcd = usb_create_hcd(&oxu_hc_driver, ...);
     if (!hcd)
         return ERR_PTR(-ENOMEM);
     oxu = hcd_to_oxu(hcd);

     ret = usb_add_hcd(hcd, irq, IRQF_SHARED);
-     if (ret < 0)
+     if (ret < 0) {
+         usb_put_hcd(hcd);
          return ERR_PTR(ret);
+     }
     return hcd;
}
```



Post-operation
usb_put_hcd

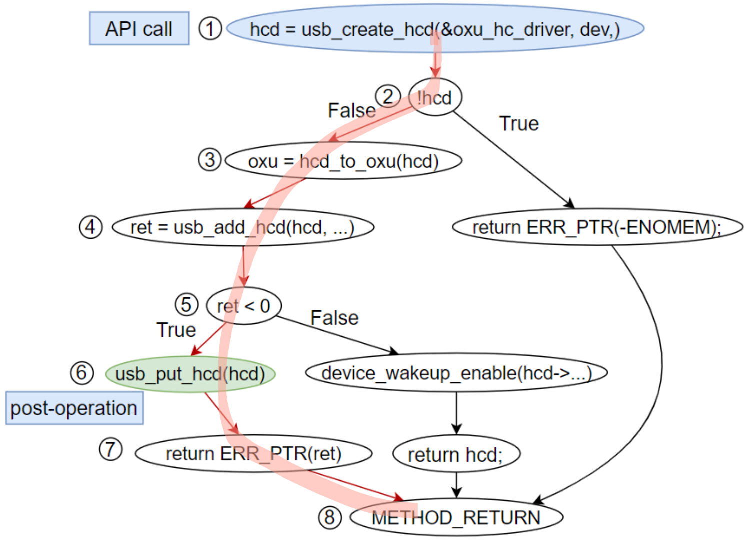
Target API
usb_create_hcd

Critical variable
hcd (return value)

Path conditions

AST difference
Path-level difference

Combine code and textual semantics



Bug Detection: Partial path-sensitive analysis

APH specification

Target API
usb_create_hcd → get callers of target API

Path conditions → get paths to be checked

Critical variable
hcd(return value) → focus on the variable

Post-operation
usb_put_hcd → check if the operation exist

🐞 If not exist, report it

APH specification-based graph (ASG)

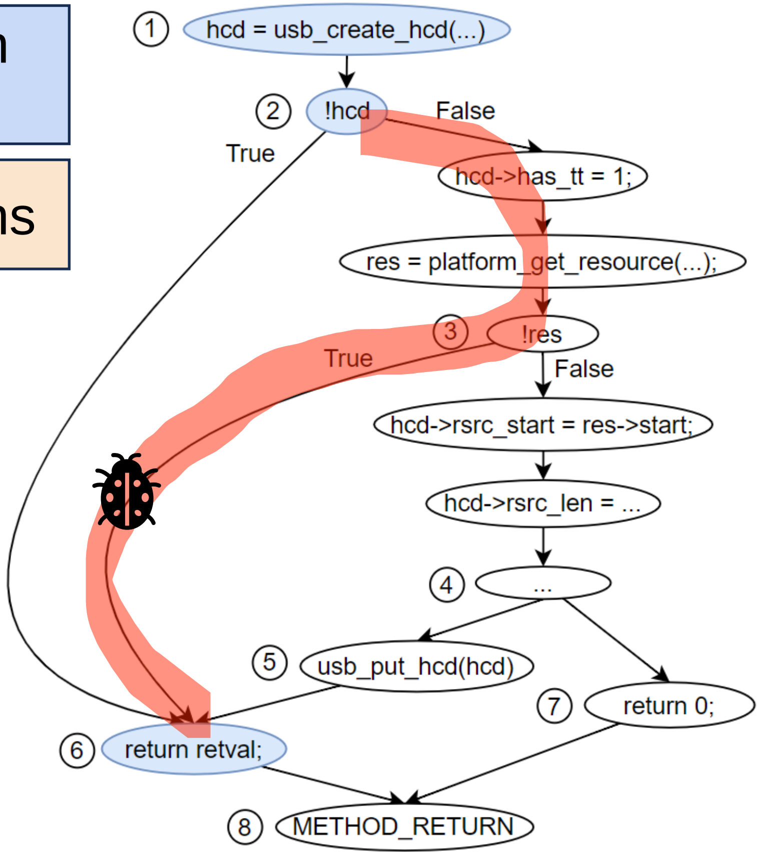
☹ extensive code with numerous paths

☺ analyze partial paths

ASG generation

↓

Path verification



ASG of function dwc2_hcd_init

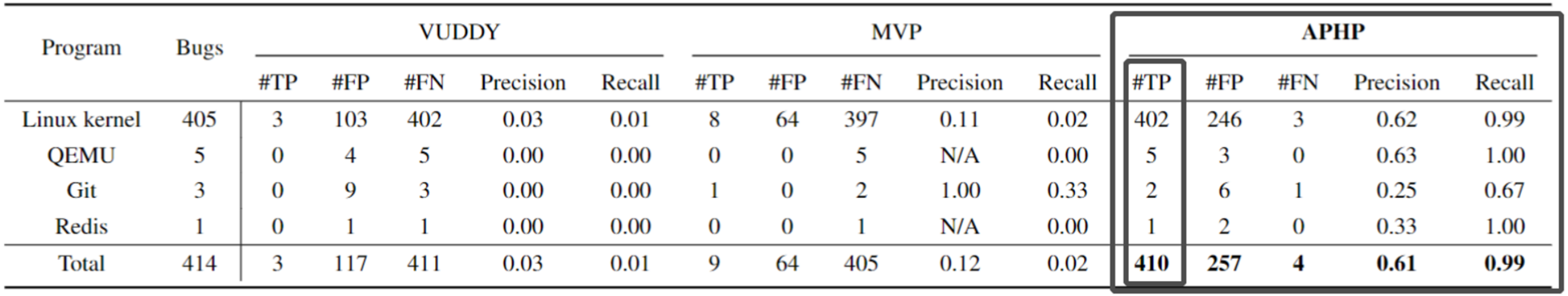
Evaluation Results: APHP Effectiveness

- Dataset
  - Four popular open-source programs: **Linux kernel, QEMU, Git and Redis**
- Results
  - Detected **410 new bugs**, 216 confirmed by developers
  - Bugs exist for **a long time**, on average more than **5 years**
  - **Various security impacts** such as resource leaks, NULL pointer dereference.

APHP detects numerous bugs on popular programs

Evaluation Results: Comparisons with SOTAs

- Comparators
  - Patch-based: VUDDY[S&P’17], MVP[Security’20]
  - Document-based: Advance[CCS’20]
  - Source code-based: IPPO[CCS’21]

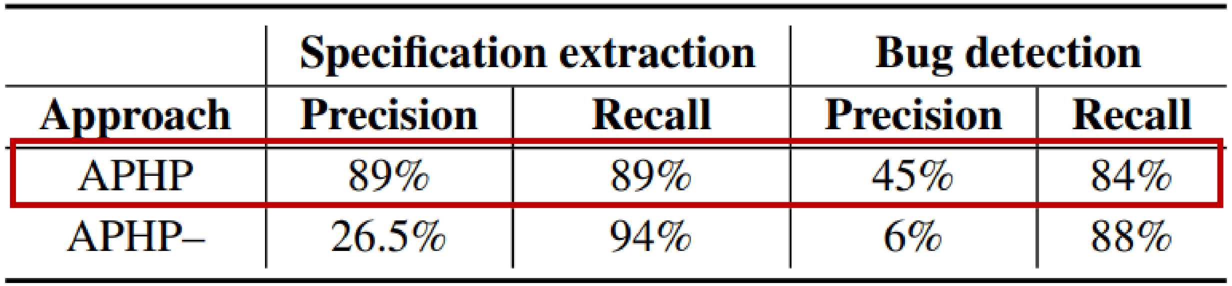


Program	Bugs	VUDDY					MVP					**APHP**				
		#TP	#FP	#FN	Precision	Recall	#TP	#FP	#FN	Precision	Recall	#TP	#FP	#FN	Precision	Recall
Linux kernel	405	3	103	402	0.03	0.01	8	64	397	0.11	0.02	402	246	3	0.62	0.99
QEMU	5	0	4	5	0.00	0.00	0	0	5	N/A	0.00	5	3	0	0.63	1.00
Git	3	0	9	3	0.00	0.00	1	0	2	1.00	0.33	2	6	1	0.25	0.67
Redis	1	0	1	1	0.00	0.00	0	0	1	N/A	0.00	1	2	0	0.33	1.00
Total	414	3	117	411	0.03	0.01	9	64	405	0.12	0.02	**410**	**257**	**4**	**0.61**	**0.99**

These tools fail to detect most APH bugs found by APHP

Evaluation Results: Ablation study

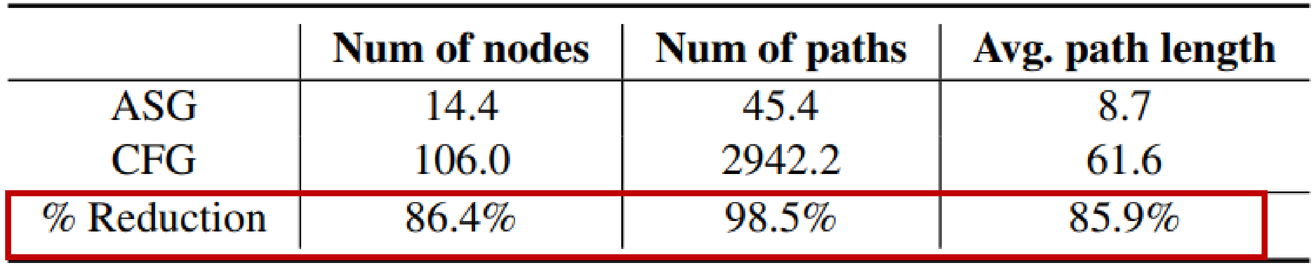
- Contribution of patch descriptions



	Specification extraction		Bug detection	
Approach	**Precision**	**Recall**	**Precision**	**Recall**
APHP	89%	89%	45%	84%
APHP–	26.5%	94%	6%	88%

Patch descriptions enhance the precision

- Contribution of APH specification-based graph (ASG)



	Num of nodes	**Num of paths**	**Avg. path length**
ASG	14.4	45.4	8.7
CFG	106.0	2942.2	61.6
% Reduction	86.4%	98.5%	85.9%

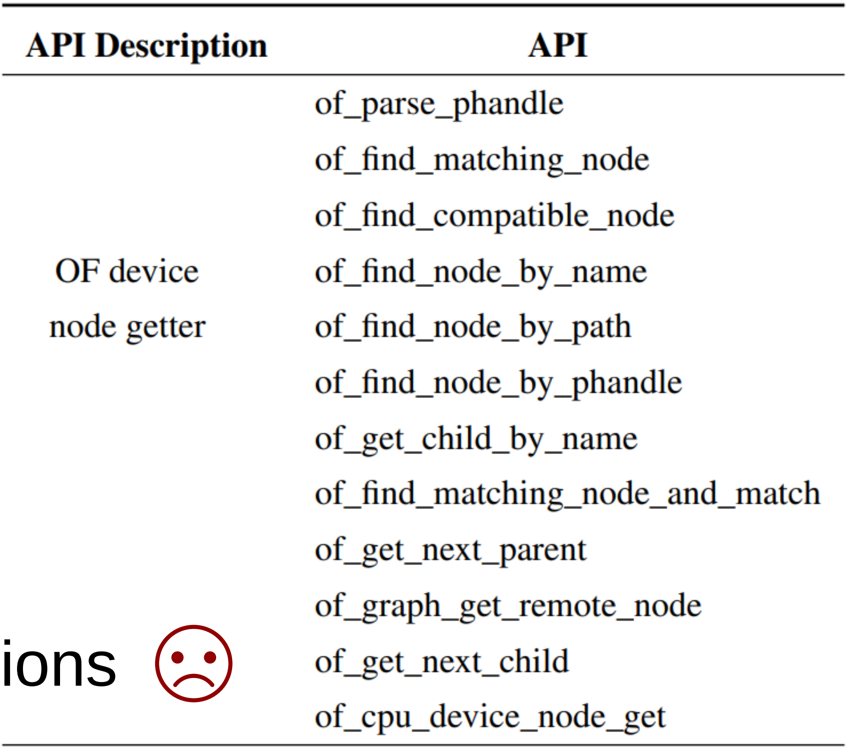
ASG reduce the amount of code analyzed

Key Findings from Detected APH Bugs

- Error-prone APIs ☹

- Implicit APH specifications ☹

- Specifications deviating from default conventions ☹



API Description	API
OF device node getter	of_parse_phandle
	of_find_matching_node
	of_find_compatible_node
	of_find_node_by_name
	of_find_node_by_path
	of_find_node_by_phandle
	of_get_child_by_name
	of_find_matching_node_and_match
	of_get_next_parent
	of_graph_get_remote_node
	of_get_next_child
	of_cpu_device_node_get

Conclusion: APHP

- Novel approach to detect APH bugs using **code and descriptions in patches**
- Detect **410 new bugs** in popular programs such as Linux Kernel, Qemu
- Valuable **knowledge gain** for bug hunters and developers
- https://github.com/Yuuoniy/APHP

Thank You

Q&A

linmiaoqian@iie.ac.cn