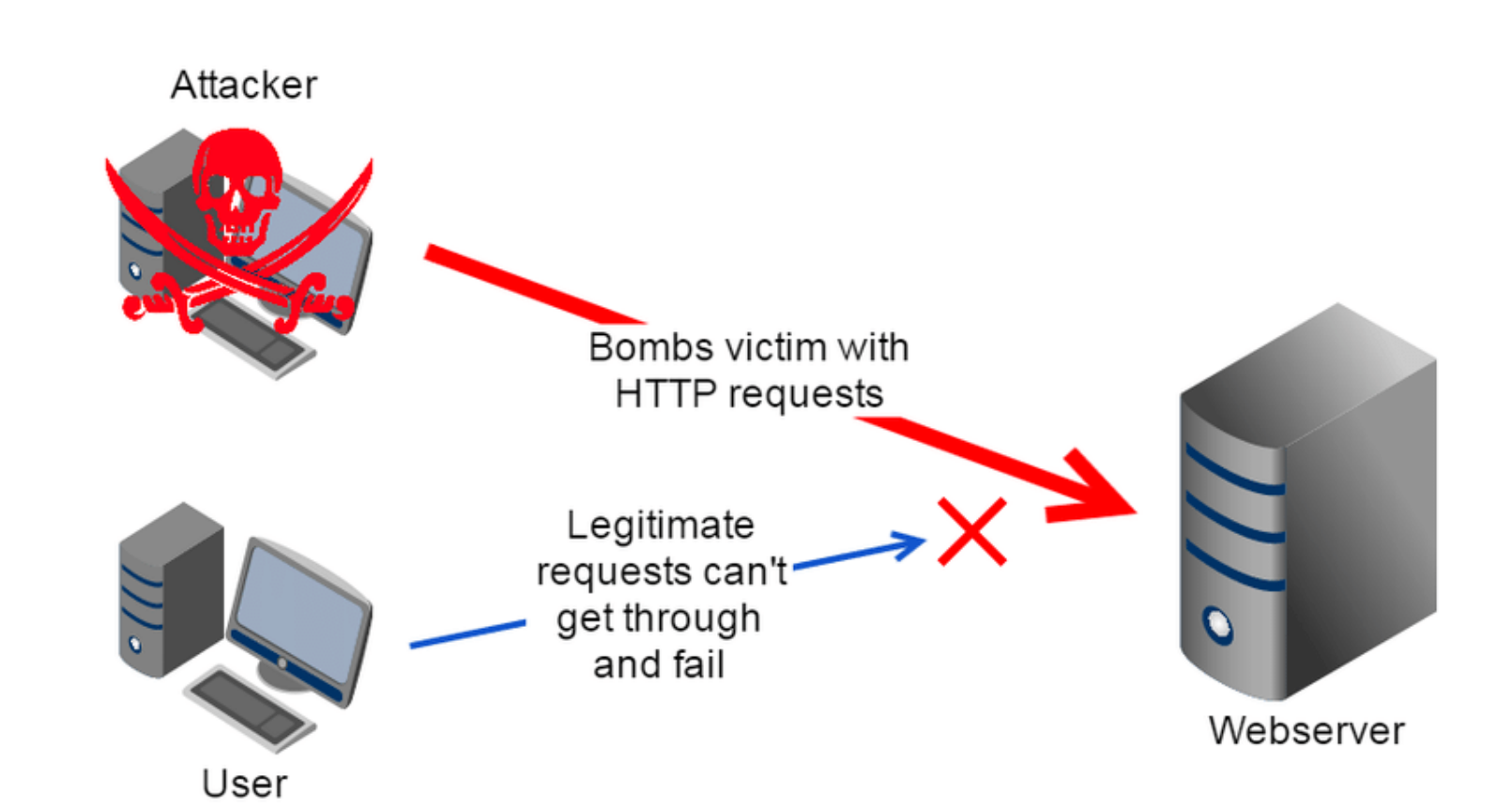
What IF Is Not Enough?
Fixing Null Pointer Dereference With Contextual Check

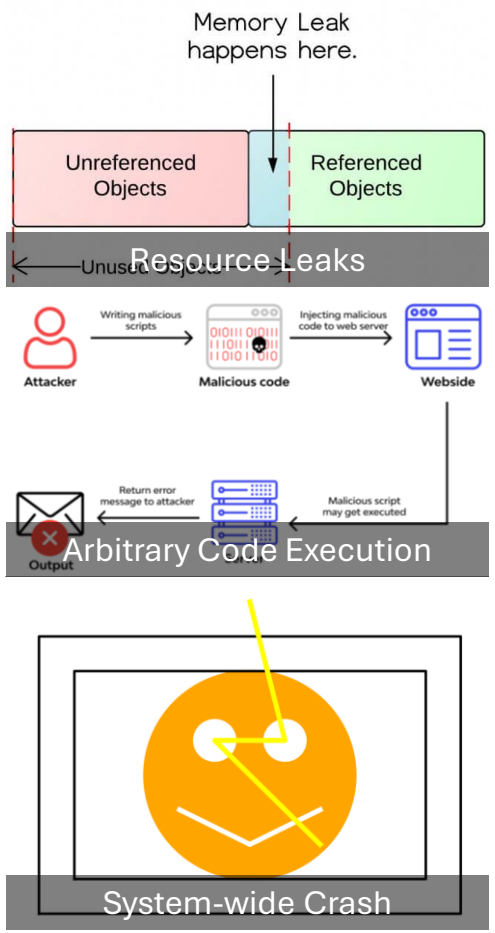
Yunlong Xing, Shu Wang, Shiyu Sun, Xu He, Kun Sun, Qi Li

USENIX Security 2024

Threats of Null Pointer Dereference (NPD)



DoS Attack



How SOTA Approaches Address NPD?

- Selecting Repair Locations
  - e.g., path congestion calculation in VFix*
- Applying Repair Operations
  - General repair framework:

```
if (variable == NULL)
    return;
normal execution;
```

OR

```
if (variable != NULL)
    normal execution;
```

However, the valuable contextual information is ignored by all SOTA approaches, resulting in incorrect patches.

* VFix: Value-Flow-Guided Precise Program Repair for Null Pointer Dereferences, in ICSE 2019.

Motivating Example 1

Intraprocedural State Retrogression

```
1 void buggy(param1, param2){
2     spin_lock(&sl->lock);
3 +   if(sl->tty == NULL){
4 +
5 +       return;
6 +   }
7     function(sl->tty, … );
8 }
```

Fixed by SOTA approach

```
1 void buggy(param1, param2){
2     spin_lock(&sl->lock);
3 +   if(sl->tty == NULL){
4 +       spin_unlock(&sl->lock);
5 +       return;
6 +   }
7     function(sl->tty, … );
8 }
```

Fixed by our method

SOTA approaches ignore the valuable intraprocedural information, such as memory freeing and lock releasing.

Motivating Example 2

Interprocedural State Propagation (Function Argument Resetting)

```
1  bool buggy(int *r, … ){
2      *r = -1;
3      if(condition1){
4  +       if (src == NULL){
5  +
6  +           return true;
7  +       }
8          // return 0 if discarded
9          *r = func(src->vcpu, …);
10         return true;
11     }
12 }
13 int caller( … ){
14     int r = -1;
15     if(buggy(&r, … ))
16         return r;
17 }
18 int caller_caller( … ){
19     if(caller( … ))
20         schedule_work();
21 }
```

Fixed by SOTA approach

```
1  bool buggy(int *r, … ){
2      *r = -1;
3      if(condition1){
4  +       if (src == NULL){
5  +           *r = 0;
6  +           return true;
7  +       }
8          // return 0 if discarded
9          *r = func(src->vcpu, …);
10         return true;
11     }
12 }
13 int caller( … ){
14     int r = -1;
15     if(buggy(&r, … ))
16         return r;
17 }
18 int caller_caller( … ){
19     if(caller( … ))
20         schedule_work();
21 }
```

Fixed by our method

Failing to reset variable ***r*** could lead to an incorrect program status.

Motivating Example 3

Interprocedural State Propagation (Call Chain Assessment)

```
1  void buggy(param1, param2,… ){
2
3      struct *new_ts;
4      new_ts = kzalloc(sizeof());
5  +   if(new_ts == NULL)
6  +       return;
7      new_ts->ts = ts;
8  }
9  int caller(param1, param2){
10     int ret;
11     if(error)
12         return -EAGAIN;
13     buggy( … );
14
15
16
17     …
18     return 0;
19 }
20 void caller_caller(param) {
21     if(caller( … ) < 0)
22         break;
23 }
```

Fixed by SOTA approach

```
1  - void buggy(param1, param2,… ){
2  + int buggy(param1, param2, … ){
3      struct *new_ts;
4      new_ts = kzalloc(sizeof());
5  +   if(new_ts == NULL)
6  +       return -ENOMEM;
7      new_ts->ts = ts;
8  }
9  int caller(param1, param2){
10     int ret;
11     if(error)
12         return -EAGAIN;
13 -   buggy( … );
14 +   ret = buggy( … );
15 +   if(ret)
16 +       return ret;
17     …
18     return 0;
19 }
20 void caller_caller(param) {
21     if(caller( … ) < 0)
22         break;
23 }
```

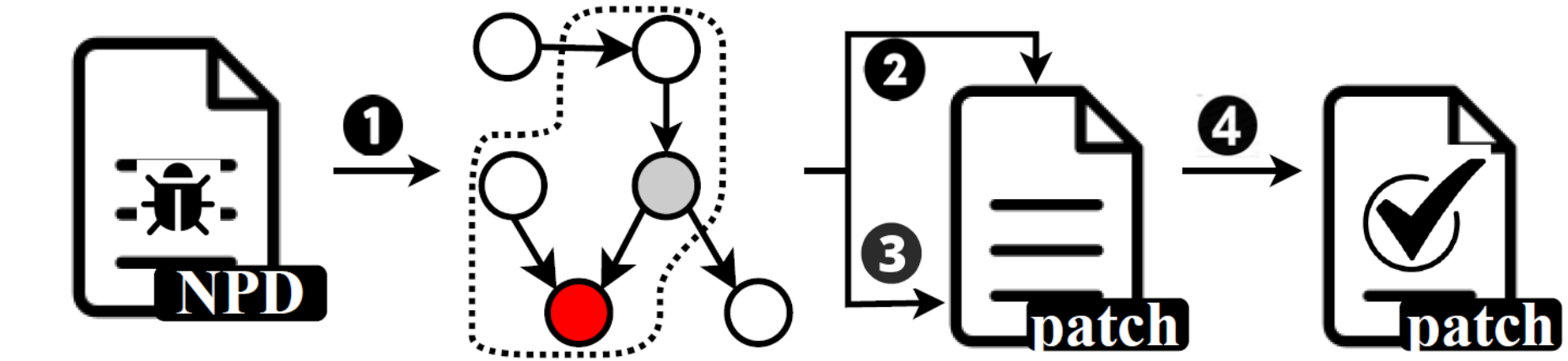
Fixed by our method

Function type modification and **call chain assessment** are required to fix this NPD issue.

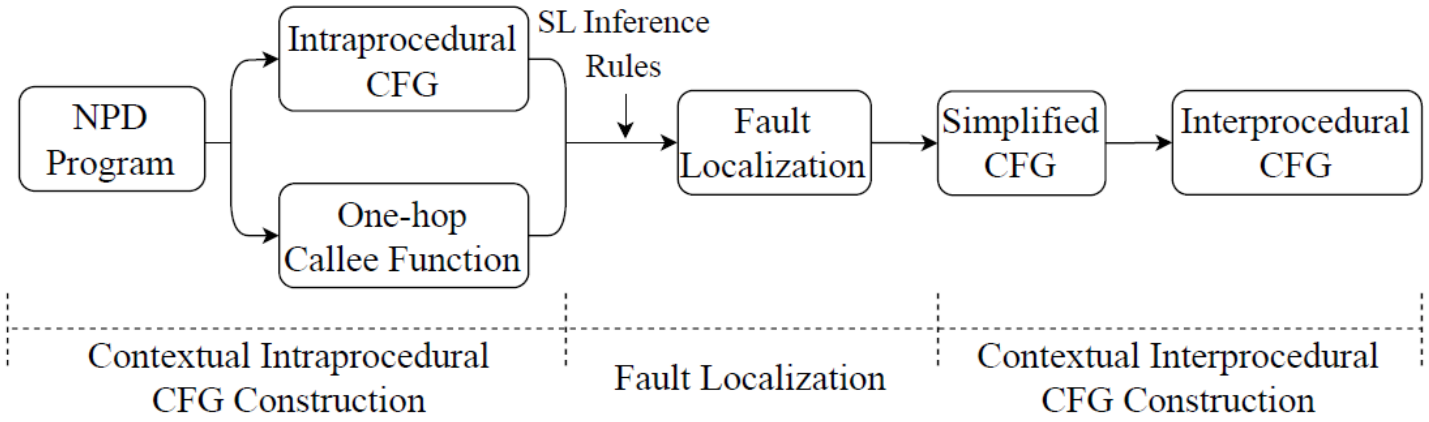
Our Contribution

We propose **CONCH** to generate accurate patches for NPD errors by considering the contextual information, including *Intraprocedural State Retrogression*, *Function Argument Resetting*, and *Call Chain Assessment*.

CONCH Design



NPD Context Graph Construction



Three steps to construct NPD context graph

$\textsc{LoadErr}: \{y \nrightarrow\} x := [y] \{err: y \nrightarrow\}$

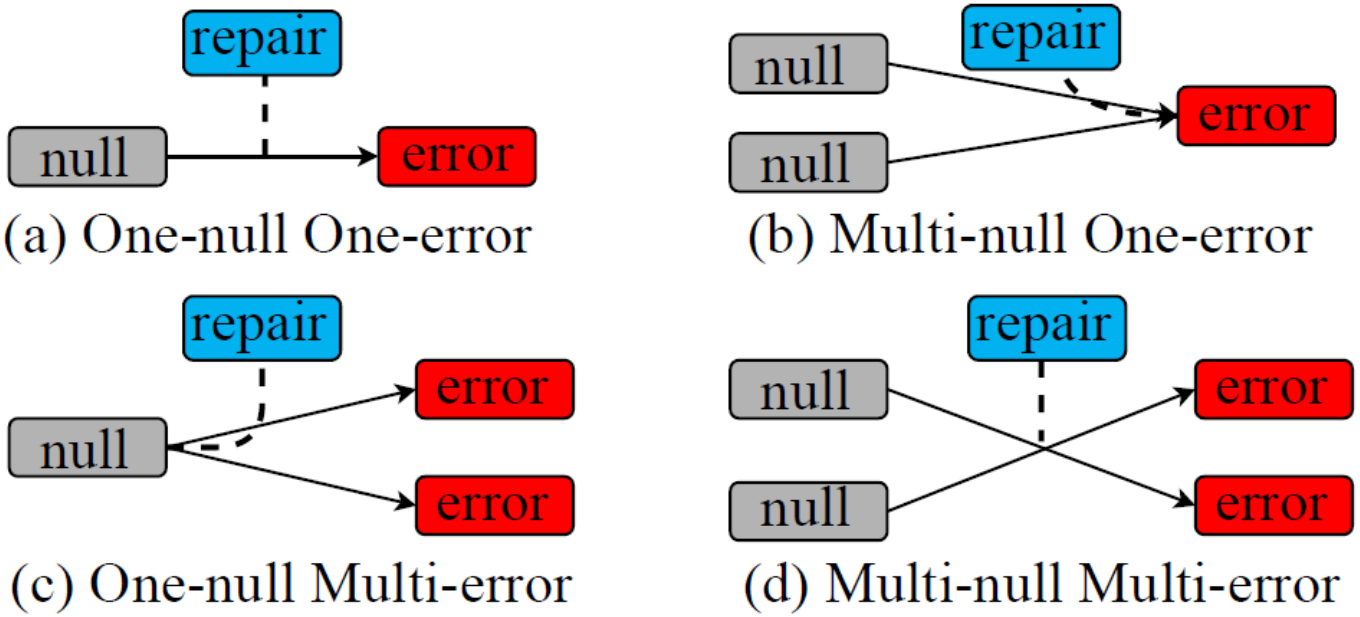
$\textsc{LoadNull}: \{y = null\} x := [y] \{err: y = null\}$

$\textsc{StoreErr}: \{x \nrightarrow\} [x] := y \{err: x \nrightarrow\}$

$\textsc{StoreNull}: \{x = null\} [x] := y \{err: x = null\}$

Separation Logic rules to localize the NPD errors

Path-sensitive Fixing Position Selection



Intraprocedural State Retrogression

- If Condition Construction

```
1      pcpu_sum = kvmalloc_array(param1, param2, param3)
2  +   if(pcpu_sum == null)
3  +       return;
4      this_sum = &pcpu_sum[cpu];
```

Null check for
CVE-2022-3107

```
1  -   amvdev_add_ts( … );  // return neg when fails
2  +   int ret = amvdec_add_ts( … );
3  +   if(ret)
4  +       return ret;
```

Exception value check for
CVE-2022-3112

```
1  +   if(info->st_info_list != NULL){
2          clist_foreach(info->st_info_list, NULL);
3          clist_free(info->st_info_list);
4  +   }
5      free(info);
```

Not-Null check for
CVE-2022-4121

Intraprocedural State Retrogression

- Local Resource Retrogression

```
1      rcu_read_lock();
2      slave = rcu_dereference(bond->curr_active_slave);
3  +   if(!slave){
4  +       rcu_read_unlock();
5  +       return -ENODEV;
6  +   }
7      xs->xso.real_dev = slave->dev;
```

Lock releasing for CVE-2022-0286

```
1      not_checked = kmalloc(sizeof(*not_checked) * 2);
2      checked = kmalloc(sizeof(*checked) * 2);
3  +   if(!not_checked || !checked){
4  +       kfree(not_checked);
5  +       kfree(checked);
6  +       return;
7  +   }
8      checked->data[] = …
9      not_checked->data[] = …
```

Memory freeing for CVE-2022-3104

Intraprocedural State Retrogression

- Return Statement Construction

```
1      if(IstensorIdControlling(tensor_id))
2          return false;
3      input_node = graph.GetNode(tensor_id.node());
4  +   if(input_node == nullptr)
5  +       return false;
6      return IsSwitch(*input_node);
```

Return false for CVE-2022-23589

```
1      if(imx_keep_uart_clocks){
2          imx_uart_clocks = kcalloc(clk_count, … );
3  +       if(!imx_uart_clocks)
4  +           return;
5          if(!of_stdout)
6              return;
7      }
```

Return nothing for CVE-2022-3114

```
1      if(rettv->vval.v_object == NULL)
2          return FAIL;
3      cl = rettv->vvval.v_object->obj->class;
4  +   if(cl == NULL)
5  +       return FAIL;
6      if(get_func_argument(…) == FAIL)
7          return FAIL;
```

Return macro for CVE-2023-1355

```
1      while(scanindent(s)){
2          var = scanname(s);
3  +       if(!val)
4  +           continue;
5          if(strcmp(var, "command") == 0)
6      }
```

Continue for CVE-2021-30219

Interprocedural State Propagation

- Global variable and function argument resetting
  - Global variable and function argument identification
  - Inferring the expected value from the data flow in the caller function
- Call chain assessment
  - Assessing the *void* function type that may execute normally when failing

Evaluation

- Datasets
  - 80 real-world NPD vulnerabilities, 18 NPD errors in Defects4j
- Other repair methods
  - VFix (SOTA approach), NPEfix (NPD repair), SimFix (general repair)
- System runtime
  - Intel i7 CPU and 16GB memory, running Ubuntu 22.04 with FBinfer 1.1.0

Performance on CVE Dataset

- CONCH can generate 68 correct and 12 incorrect patches

	Same Fixing	Semantic Equivalence	Incorrect Patches	Proportion
VFix	29	18	33	58.75%
NPEfix	15	4	61	23.75%
SimFix	18	8	54	32.5%
CONCH	**36**	**32**	**12**	**85%**

Performance on CVE Dataset

- Incorrect patches and their reasons

Category	CVE ID	If Condition	Generated Patches	Why CONCH Cannot Generate Correct Patches
Unobtainable Member	CVE-2022-1674	rmp->regprog != NULL	rmp != NULL	member *regprog* cannot be obtained in context
	CVE-2022-1620	rmp->regprog != NULL	rmp != NULL	member *regprog* cannot be obtained in context
	CVE-2016-2782	serial->num_bulk_in < 2	serial != NULL	member *num_bulk_in* cannot be obtained in context
	CVE-2014-0101	!net->sctp.auth_enable	net == NULL	member *sctp.auth_enable* cannot be obtained in context
	CVE-2013-0313	inode->i_op->removexattr != NULL	inode->i_op != NULL	`removexattr` is not a function in context
Unobtainable Relation	CVE-2022-2874	cctx->ctx_skip != SKIP_YES	cctx != NULL	relation with SKIP_YES cannot be obtained in context
	CVE-2018-1092	ino == EXT4_ROOT_INO	ino == 0	relation with EXT4_ROOT_INO cannot be obtained in context
	CVE-2012-6647	uaddr == uaddr2	uaddr && uaddr2	relation that *uaddr* is equal to *uaddr2* cannot be obtained in context
Special Function	CVE-2022-3621	nilfs_is_metadata_file_inode(inode)	-	special function for sanity check
	CVE-2022-2302	JFS_IP(ipimap)->i_imap	-	special function for validation check
	CVE-2013-5634	!kvm_vcpu_initialized(vcpu)	-	special function for initializing vCPU
	CVE-2013-4119	!SecIsValidHandle(handle)	-	special function for validation check

Performance on Defects4j

- CONCH can generate 16 correct patches and 2 incorrect patches

Project	#NPD	Fixed by VFix			Fixed by CONCH		
		Same	Semantic	Incorrect	Same	Semantic	Incorrect
Chart	7	5	0	2	5	2	0
Closure	6	2	1	3	2	2	2
Lang	2	1	0	1	2	0	0
Math	2	1	1	0	1	1	0
Time	1	1	0	0	1	0	0
Total	18	10	2	6	11	5	2

Conclusion

- We propose CONCH to fix NPD errors with contextual checks, ensuring a more effective and complete vulnerability control
- We are the first to address local resource retrogression and reset global variable and function argument in NPD fixing
- The experimental results show that CONCH outperforms the SOTA approaches

Q & A