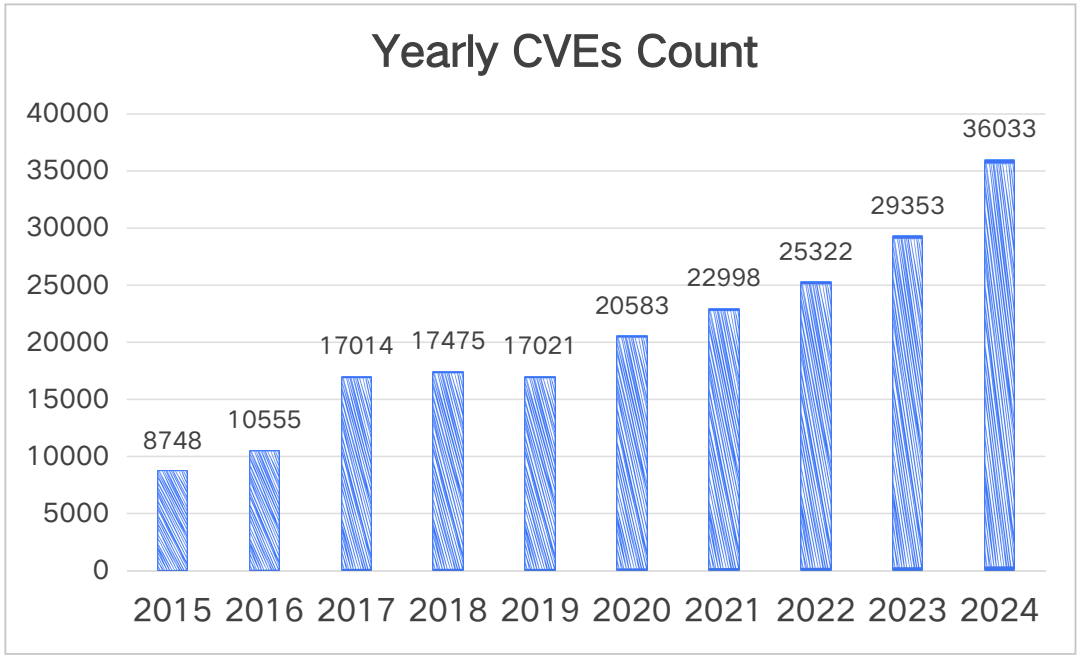
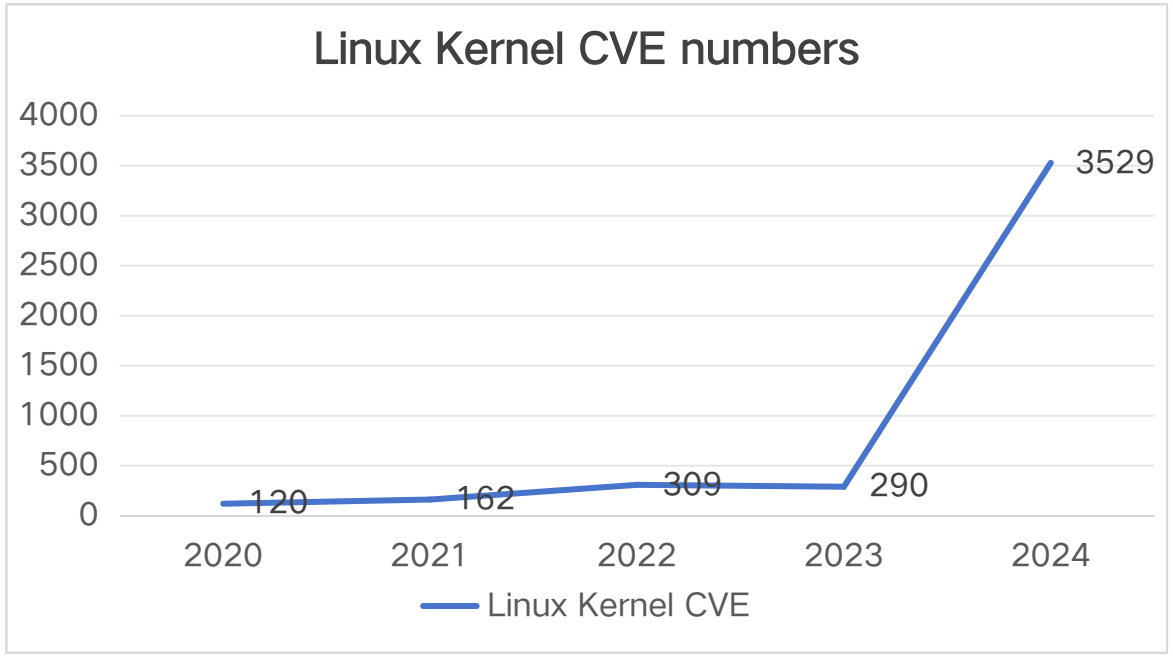
VulShield: Protecting Vulnerable Code Before Deploying Patches

Yuan Li, Chao Zhang✉, Jinhao Zhu, Penghui Li, **Chenyang Li**, Songtao Yang, Wende Tan

Massive Vulnerabilities



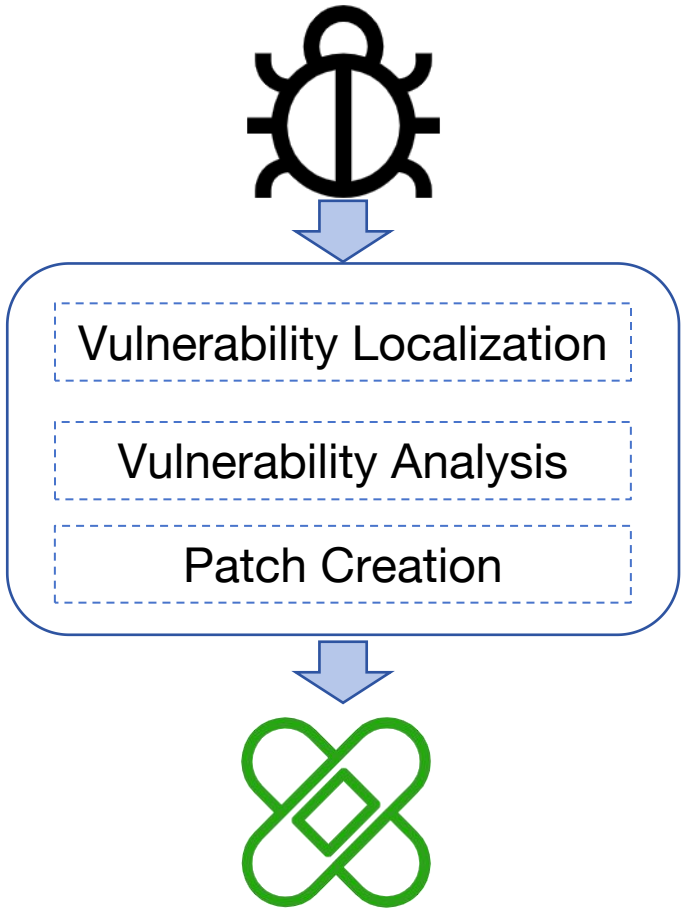
https://app.opencve.io/statistics

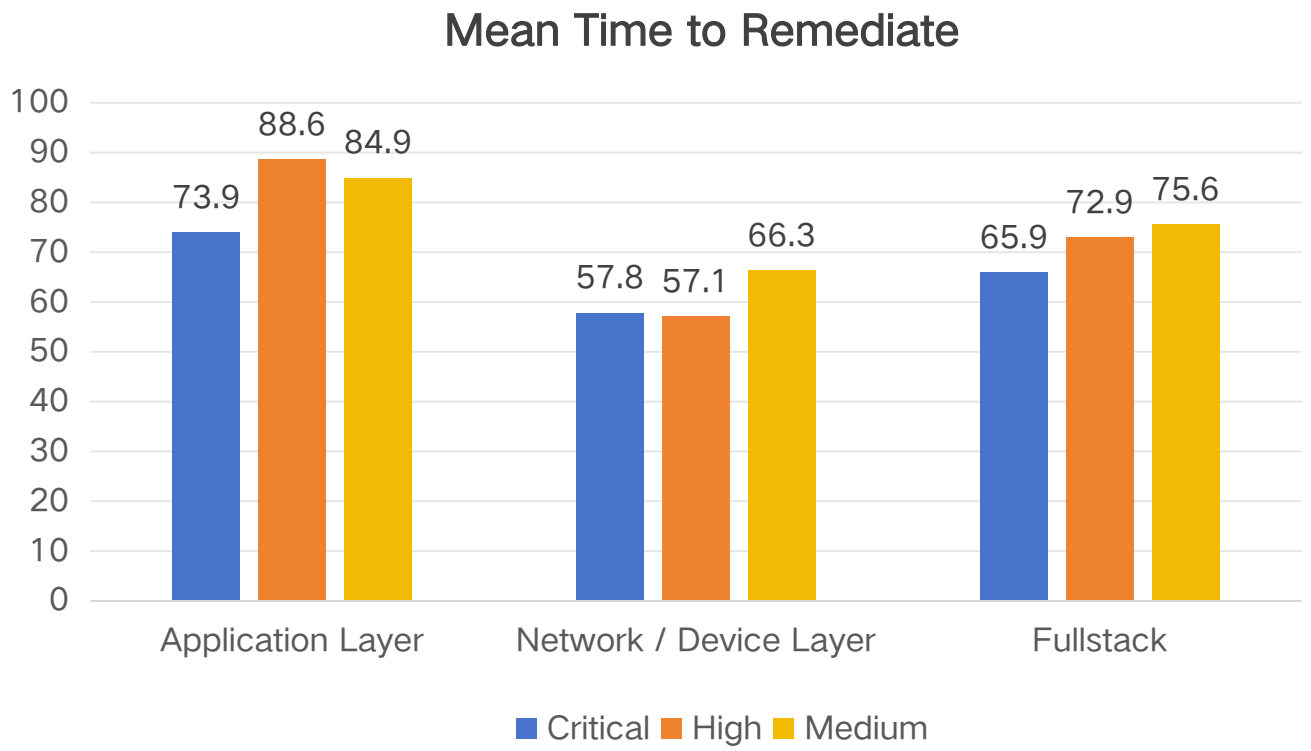


https://tuxcare.com/blog/the-linux-kernel-cve-flood-continues-unabated-in-2025/

The number of CVEs is vast and surging each year.

Delayed Patching

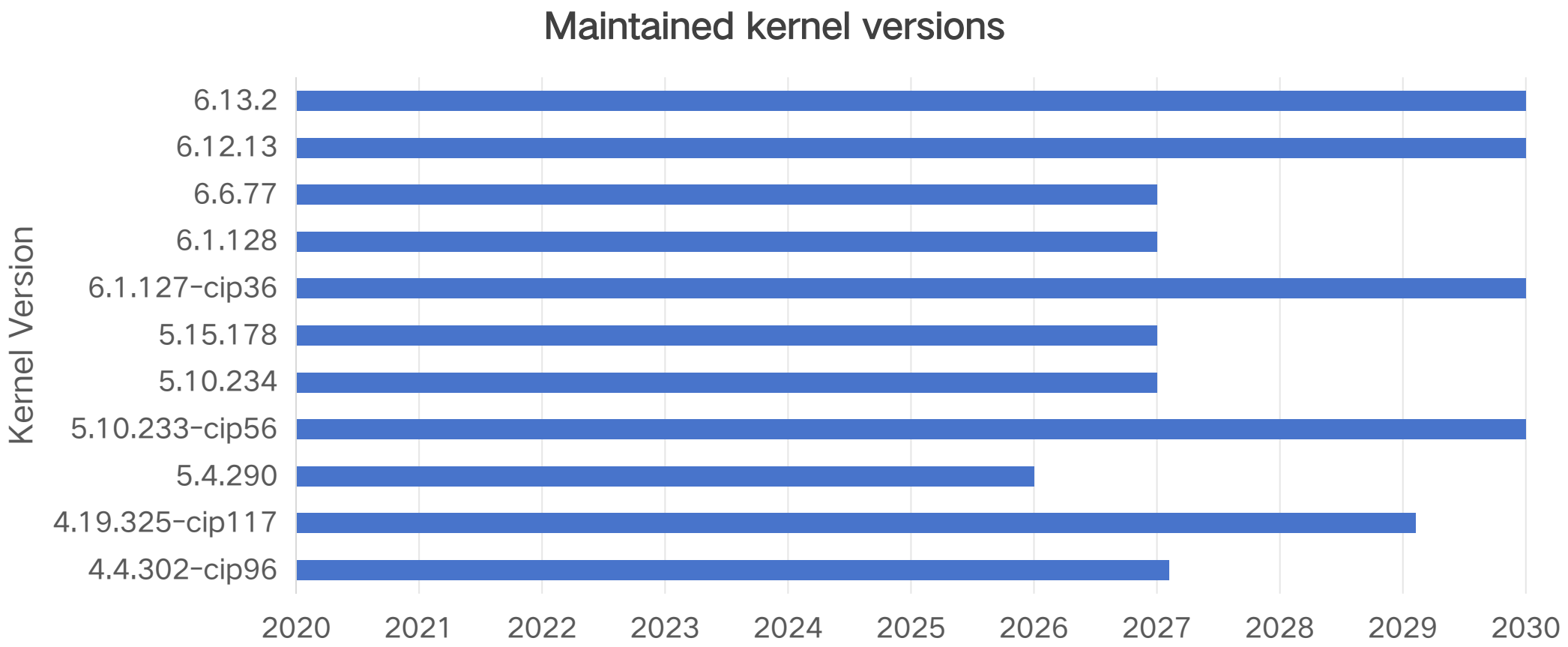




https://www.edgescan.com/wp-content/uploads/2024/03/2023-Vulnerability-Statistics-Report.pdf

Vulnerability fixes take long, leaving attackers a window of dozens of days.

Diverse software/kernel Distributions and Versions in Use



https://en.wikipedia.org/wiki/Linux_kernel_version_history

For example, the Linux kernel encompasses numerous distributions and versions, reflecting its extensive adoption and customization.

Motivation

A rapid and effective mitigation mechanism can cover various vulnerabilities before patch release while being non-intrusive and backward-compatible.

Observation

No.	Category	Percentage
1	add input sanitization checks	43.5%
2	change input sanitization checks	25.1%
3	add data structures	6.1%
4	change data structure definitions	6.5%
5	change data structure references	22.3%
6	change function parameters	10.9%
7	add or change function calls	15.3%
8	add functions	4.7%
9	change functions	7.6%

Statistics on different memory corruption patch types from **PatchScope** [1]

Code Pattern	Total Patches #	Total Patches ratio to total
if-exit	25	2.4%
if-return	437	42.4%
if-goto	147	14.3%
if-body	229	22.2%
Misc.	193	18.7%
Total	1031	100%

Statistics of patches patterns [2]

Many patches follow the pattern of constraint expressions.

[1] Zhao, Lei, et al. "PatchScope: Memory object centric patch diffing." Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 2020.
[2] Yang, Songtao, et al. "1dFuzz: Reproduce 1-Day Vulnerabilities with Directed Differential Fuzzing." Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. 2023.

Intuition

Constraint expressions to describe most vulnerability conditions.

Cover various vulnerabilities.

Intuition

Constraint expressions to describe most vulnerability conditions.

Existing probing mechanisms can collect runtime information without requiring intrusive modifications.

Non-intrusive and backward-compatible.

Intuition

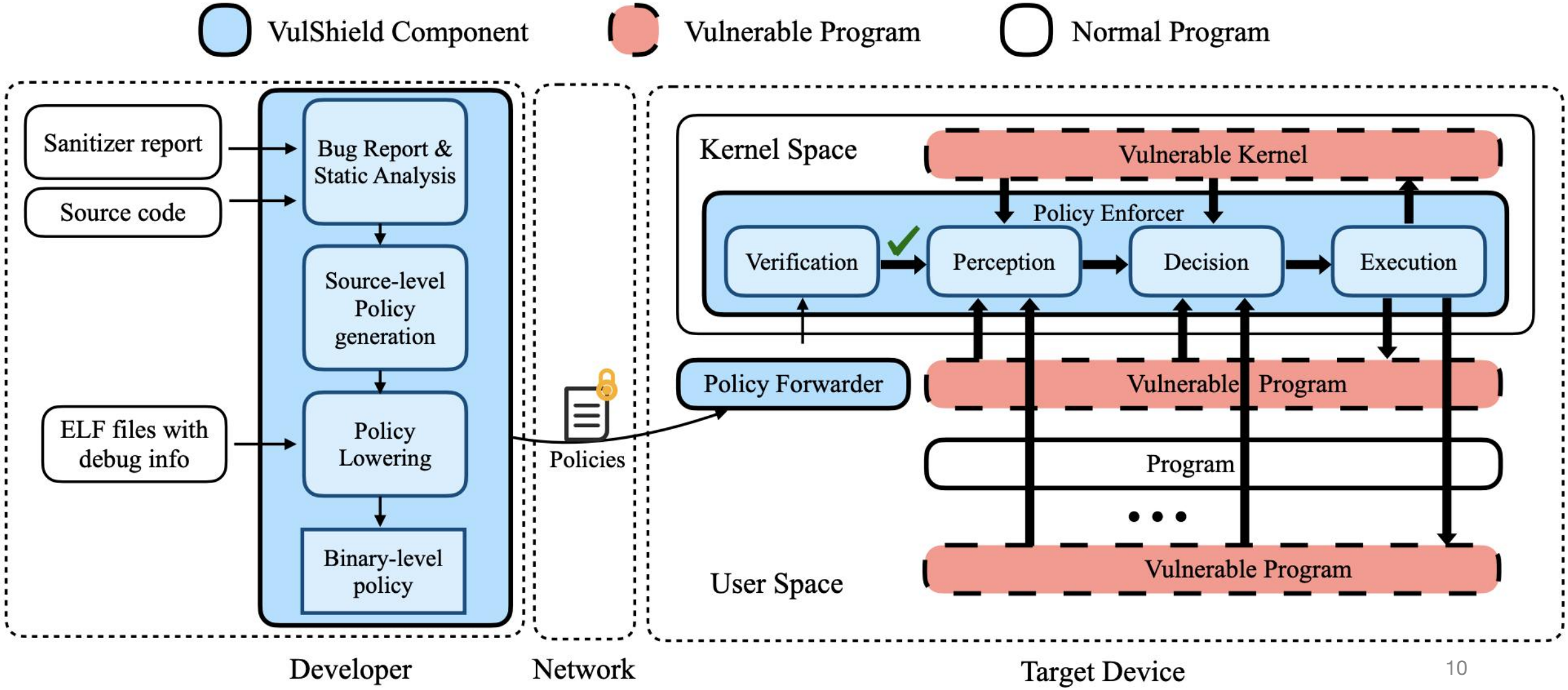
Constraint expressions to describe most vulnerability conditions.

Existing probing mechanisms can collect runtime information without requiring intrusive modifications.

Automatically generate mitigation policy based on vulnerability error reports.

Rapid and effective.

Overview


VulShield Component
Vulnerable Program
Normal Program
Sanitizer report
Source code
ELF files with debug info
Bug Report & Static Analysis
Source-level Policy generation
Policy Lowering
Binary-level policy
Policies
Kernel Space
Vulnerable Kernel
Policy Enforcer
Verification
Perception
Decision
Execution
Policy Forwarder
Vulnerable Program
Program
Vulnerable Program
User Space
Developer
Network
Target Device

Source-level Policy Generation

UBSAN: array-index-out-of-bounds in drivers/net/hamradio/6pack.c:845:16

index 400 is out of range for type 'unsigned char [400]'

.....

Call Trace:

dump_stack+0x107/0x163

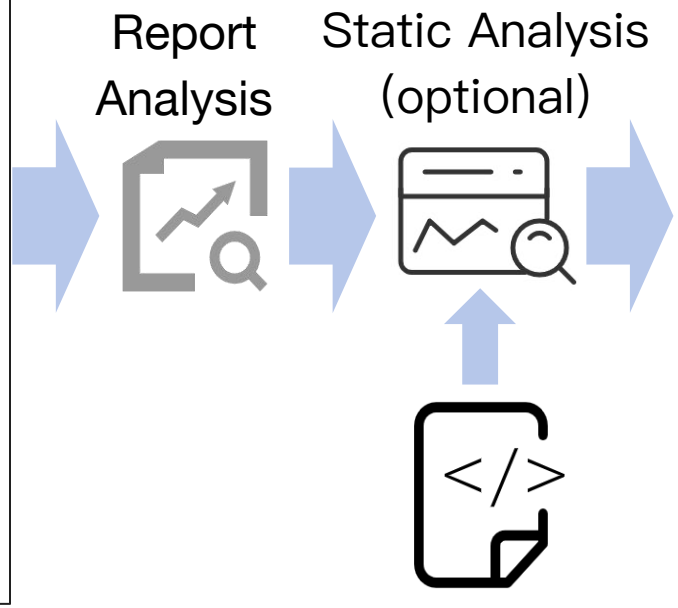
ubsan_epilogue+0xb/0x5a

__ubsan_handle_out_of_bounds.cold+0x62/0x6c

decode_data.part.0+0x2c8/0x2e0

sixpack_receive_buf+0xcb1/0x1320

The sanitizer report for the **instrumented** binary




```
---
VulType: array-index-out-of-bounds
Variables:
  - {Name: rx_count_cooked, Type: sub-object,
    Index:10, ObjType: S.struct.sixpack, Value: {
    Name: sp, Type: pointer, Value: 0th parameter}
    }
Constraint: sp->rx_count_cooked + 2>=400
Decision point:
  - {FuncName: decode_data, Type: func_entry,
    Source: drivers/net/hamradio/6pack.c:832}
Mitigation action: return
---
```

The source–level policy of the **uninstrumented** binary

Policy Lowering for Binary-level Policy

```
---
VulType: array-index-out-of-bounds
Variables:
  - {Name: rx_count_cooked, Type: sub-object,
    Index:10, ObjType: S.struct.sixpack, Value: {
    Name: sp, Type: pointer, Value: 0th parameter}
    }
Constraint: sp->rx_count_cooked + 2>=400
Decision point:
  - {FuncName: decode_data, Type: func_entry,
    Source: drivers/net/hamradio/6pack.c:832}
Mitigation action: return
---
```

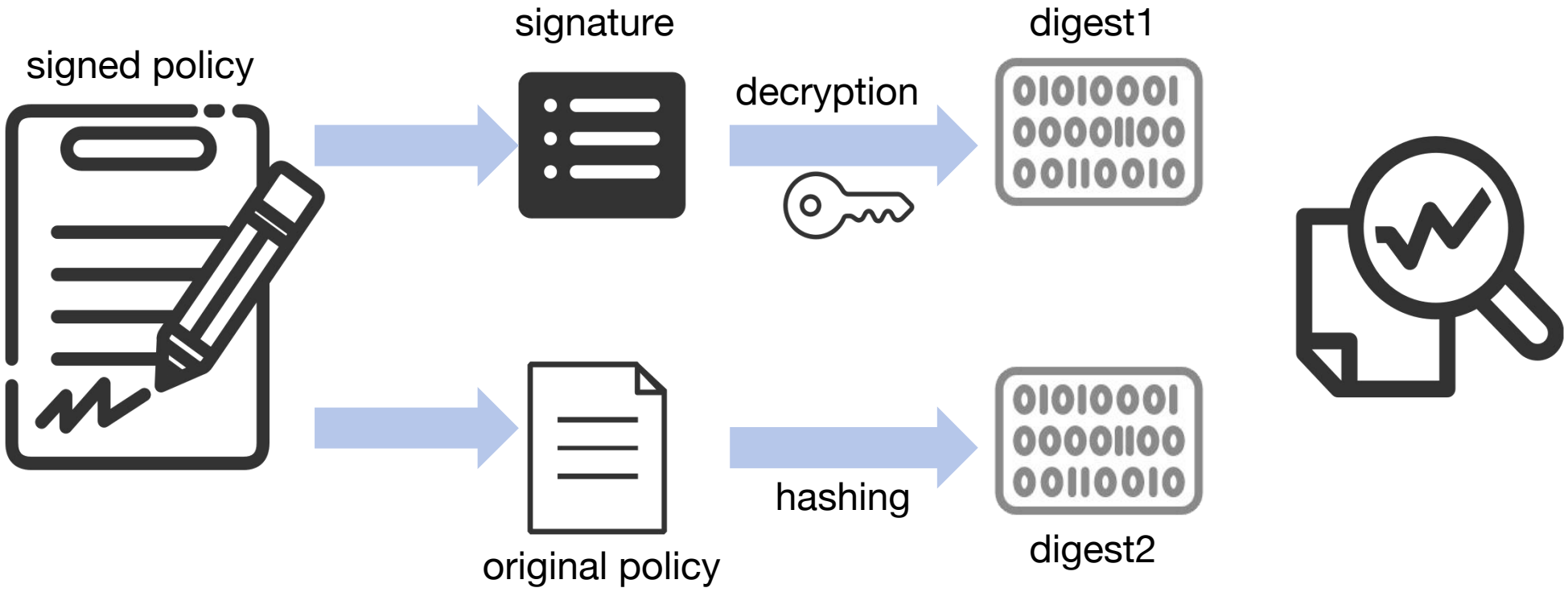
```
0x0521de10:   DW_TAG_subprogram
              DW_AT_abstract_origin   (0x0521b972
"decode_data")
              DW_AT_low_pc    (0xffffffff84e6acb0)
              DW_AT_high_pc   (0xffffffff84e6af81)
.....
0x0521b972:   DW_TAG_subprogram
              DW_AT_name      ("decode_data")
              DW_AT_decl_file
("/home/clive/linux_kernel/linux-5.11-
rc1/drivers/net/hamradio/6pack.c")
              DW_AT_decl_line (832)
              DW_AT_decl_column       (0x0d)
.....
```

Policy Lowering

```
---
Decision point:
  - Func: decode_data
    offset: 0x0
    value:
      - {val: [rdi+0x1cc], name: count}
Constraint: count+ 2>=400
Mitigation action: return
---
```

Target binaries often lack debuginfo, requiring policy lowering to the binary level.

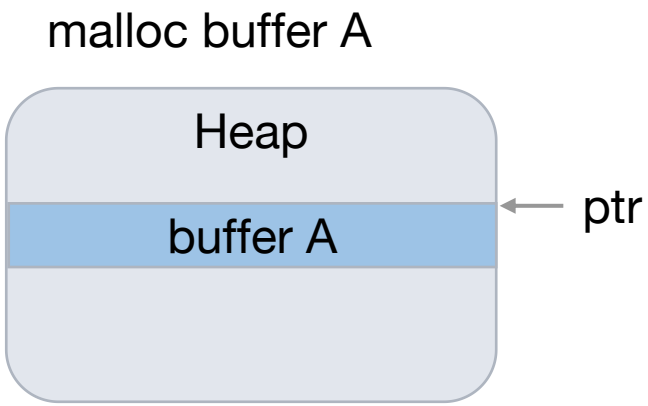
Policy Enforcer - Verification



Policy Enforcer - Perception (optional)

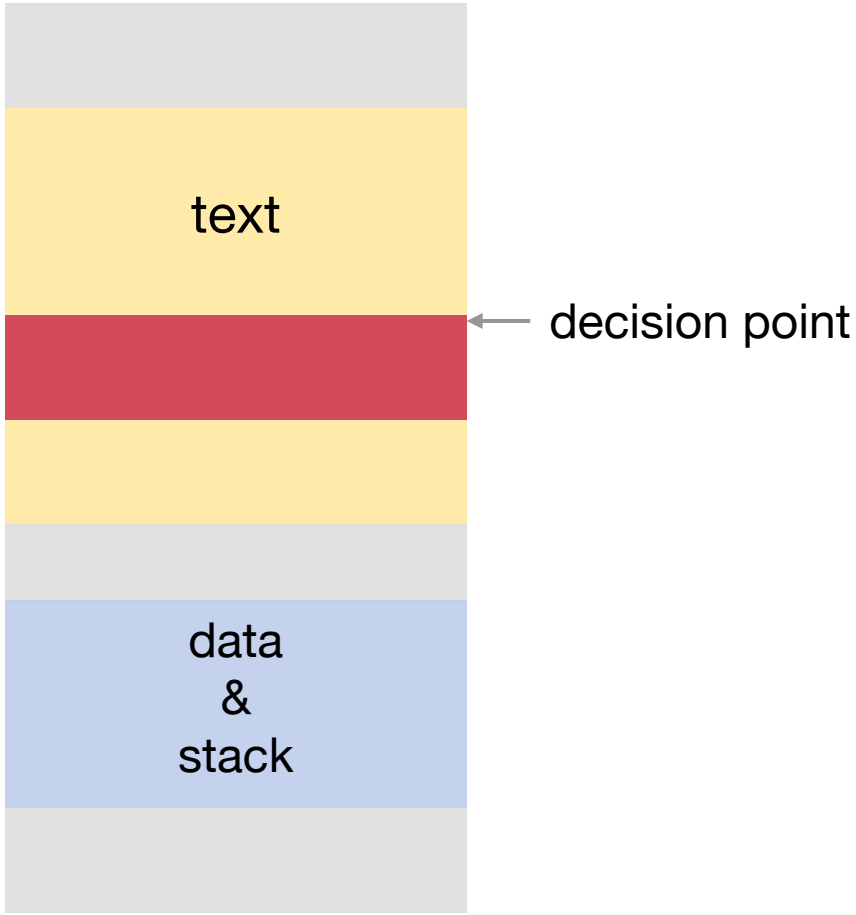
```
##malloc buffer A
char *ptr = malloc(Len);
......
data[i] = L'\0';
```

← Perception point



Collecting necessary runtime information before making decisions.

Policy Enforcer - Decision & Execution

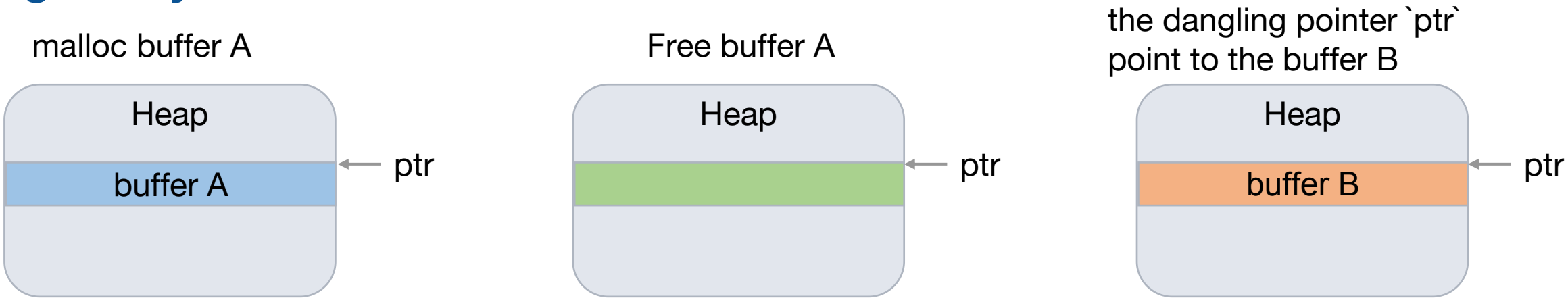



```
##buffer overflow
memcpy(target_buffer,shellcode,vul_size)
……
goto target
```

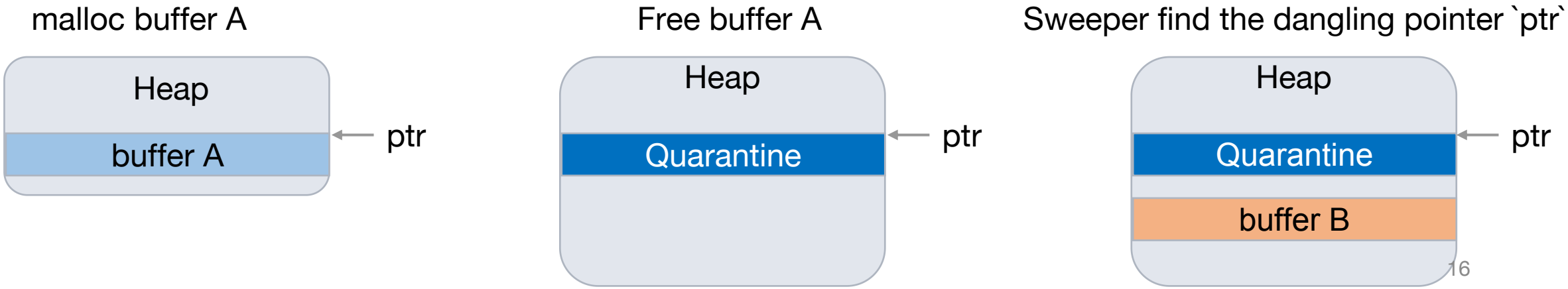
```
## Do checks before the vulnerability code
If (exp)
    error_handler()

##buffer overflow
memcpy(target_buffer,valid_content,valid_size)
……
goto target
```

Policy Enforcer - Other mitigation actions

Regular System



Quarantine and Sweeper



Effectiveness

Support both user-space programs and kernel.

Cover 9 different types of vulnerabilities.

Target Binary		CVE No.	Vulnerability Type	Policy Type	Is it effective?
Magma	libpng	CVE-2013-6954	NULL Pointer Dereference	Expression-based	✓
	libtiff	CVE-2016-10270	Out-of-bounds Access	Expression-based	✓
		CVE-2019-7663	Invalid Address dereference	Expression-based	✓
	libxml	CVE-2017-9047	Buffer Overflow	Expression-based	✓
		CVE-2016-1836	Use After Free	Expression-based	✓
	openssl	CVE-2016-2108	Buffer Underflow	Expression-based	✓
		CVE-2016-6309	Use After Free	Expression-based	✓
		CVE-2017-3735	Out-of-bounds Access	Expression-based	✓
	php	CVE-2018-14883	Out-of-bounds Access	Expression-based	✓
		CVE-2019-11034	Out-of-bounds Access	Expression-based	✓
	poppler	CVE-2019-14494	Divide By Zero	Expression-based	✓
		CVE-2019-9959	Integer Overflow	Expression-based	✓
		CVE-2019-10873	NULL Pointer Dereference	Expression-based	✓
		CVE-2019-10872	Out-of-bounds Access	Expression-based	✓
		CVE-2019-7310	Out-of-bounds Access	Expression-based	✓
		CVE-2018-13988	NULL Pointer Dereference	Expression-based	✓
		CVE-2018-10768	NULL Pointer Dereference	Expression-based	✓
		CVE-2017-14617	Divide By Zero	Expression-based	✓
	sqlite3	CVE-2015-3414	NULL Pointer Dereference	Expression-based	✓
libIEC61850		CVE-2018-18957	Buffer Overflow	Expression-based	✓
LibreDWG		CVE-2020-21813	Buffer Overflow	Expression-based	✓
Nginx		CVE-2013-2028	Integer Overflow	Expression-based	✓
gpac		CVE-2020-19488	NULL Pointer Dereference	Expression-based	✓
libxml2		CVE-2021-3518	Use After Free	Expression-based	✓
Linux kernel		CVE-2017-18344	Out-of-bounds Access	Expression-based	✓
		CVE-2021-42008	Out-of-bounds Access	Expression-based	✓
		CVE-2022-2588	Use After Free	QS	✓
		CVE-2021-3492	UAF+Double Free+Mising Release	QS	✢
		6dc9ae7 [50]	Use After Free	QS	✓
		ca4463b [36]	Race condition+UAF	QS	✓
		a834b99 [51]	Data race	DEC	✓

✓: effectively detect and prevent the triggering of reported vulnerabilities. ✢: only support partial the reported vulnerabilities

Performance Evaluation - Nginx

- Implementing the policy of CVE-2013-2028
- n - total number of requests
- c - number of concurrent requests
- Each request triggers the policy execution

Nginx	w/o VulShield		w/ VulShield	
	Time per request		Time per request	
	a concurrent group	a single request	a concurrent group	a single request
n=100000, c=10	0.130 ms	0.013 ms	0.140 ms	0.014 ms
n=100000, c=100	1.313 ms	0.013 ms	1.411 ms	0.014 ms
n=100000, c=1000	15.075 ms	0.015 ms	15.827 ms	0.016 ms
n=100000, c=10000	136.712 ms	0.014 ms	136.238 ms	0.014 ms

Only +0.001ms/req

Performance Evaluation - UnixBench

Test Items	0 policy		1 policy			2 policies			3 policy
	w/o VulShield	w/ VulShield	Expression-based	QS	DEC	Expression-based + QS	Expression-based + DEC	QS + DEC	Expression-based + QS + DEC
Dhrystone 2 using register variables	6502.0	6436.1	6411.4	6419.2	6416.2	6413.2	6364.9	6311.3	6419.9
Double-Precision Whetstone	2028.8	2029.6	2028.8	2029.2	2030.0	2029.0	2030.3	2028.8	2030.2
Execl Throughput	2262.6	2270.1	2272.2	2261.6	2267.8	2257.4	2263.9	2263.5	2266.3
File Copy 1024 bufsize	7101.5	7097.5	7016.9	7086.5	7060.4	7063.7	7054.6	7111.6	6995.6
File Copy 256 bufsize	4687.1	4717.2	4723.8	4709.9	4729.6	4722.2	4728.0	4715.8	4718.3
File Copy 4096 bufsize	12954.1	12467.8	11974.8	11915.7	12513.3	12031.6	11824.6	12317.2	11646.3
Pipe Throughput	3356.5	3354.4	3369.3	3379.8	3361.5	3381.5	3366.6	3365.0	3397.7
Context Switching	1277.8	1293.6	1293.0	1297.2	1290.0	1290.0	1290.7	1293.4	1284.7
Process Creation	2612.1	2665.5	2632.6	2642.2	2642.7	2584.2	2542.9	2597.7	2612.6
Shell Scripts (1 concurrent)	4338.5	4344.3	4348.1	4340.8	4334.4	4332.4	4333.8	4327.6	4337.6
Shell Scripts (8 concurrent)	4054.0	4058.4	4058.5	4052.8	4053.8	4045.4	4044.5	4017.0	4043.3
SysCall Overhead	2478.1	2478.4	2476.9	2478.4	2479.6	2477.2	2478.6	2476.7	2471.6
System Benchmarks Index Score	3685.7	3684.3	3665.4	3667.3	3679.7	3659.6	3647.1	3662.6	3651.2
Overhead	-	0.038%	0.551%	0.499%	0.163%	0.708%	1.047%	0.627%	0.936%

The combined overhead of the three mitigation mechanisms is 1.047%.

Conclusion

- Policy-Based Protection
  - Utilizes a policy-driven approach for targeted and effective mitigation.
- Broad Coverage
  - Demonstrated effectiveness in mitigating at least 9 different types of vulnerabilities.
- Low Overhead
  - Nginx Performance: +0.001ms/req.
  - UnixBench Performance: up to **1.047%** overhead.

Thanks!

Any questions?