Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
SERICS SECURITY AND RIGHTS IN THE CYBERSPACE

QMSan: Efficiently Detecting Uninitialized Memory Errors During Fuzzing

Matteo Marini (Sapienza)
Daniele Cono D'Elia (Sapienza)
Mathias Payer (EPFL)
Leonardo Querzoni (Sapienza)

UUM Errors

```
void foo(){
    char buf[4], a;
    read(0, buf, 4);
    a = buf[0];
    if(a==MAGIC_BYTE)
        puts("Hello world!");
}
```

- Will this program print "*Hello world!*"?

Obvious answer: it depends on the first char of the buffer!

But what if **nothing is read**?

Use-of-Uninitialized-Memory (UUM) error!

UUM Errors - Detection

- Define a **shadow memory**
  - Contains Initialization status of memory

- **Propagate** the shadow memory
  - Propagation rules

- Check shadow memory on **sensitive operations**
  - Pointer derefentiation
  - Conditional branches
  - Data in system calls

?

Loading uninitialized data **is allowed…**

…As long as **<u>its content is not used</u>** in sensitive operations

Memory Sanitizer (MSan)

- State-of-the-Art UUM detection
  - Compile-time solution

Pros:

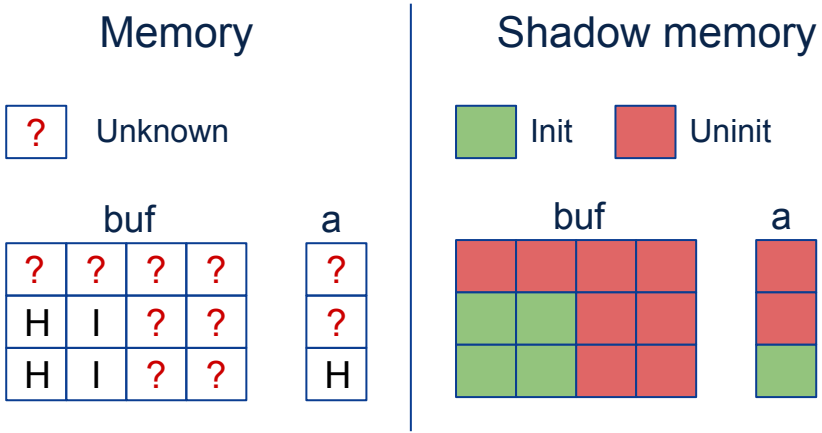
- Accurate
- Fuzzing-compatible
- Acceptable Slowdown (2-3x)

Cons:

- Requires **recompilation**
- **All code** must be instrumented
  - Libraries
- LLVM only

MSan - Workflow

```
void foo(){
    char buf[4], a;
    read(0, buf, 4);
    a = buf[0];
    if(a==MAGIC_BYTE)
        puts("Hello world!");
}
```



Binary Detection

- Detect UUM errors at the binary level
  - Similar workflow as MSan
  - Much more instrumentation

Pros:

- More generic
  - No recompilation
  - Closed-source software

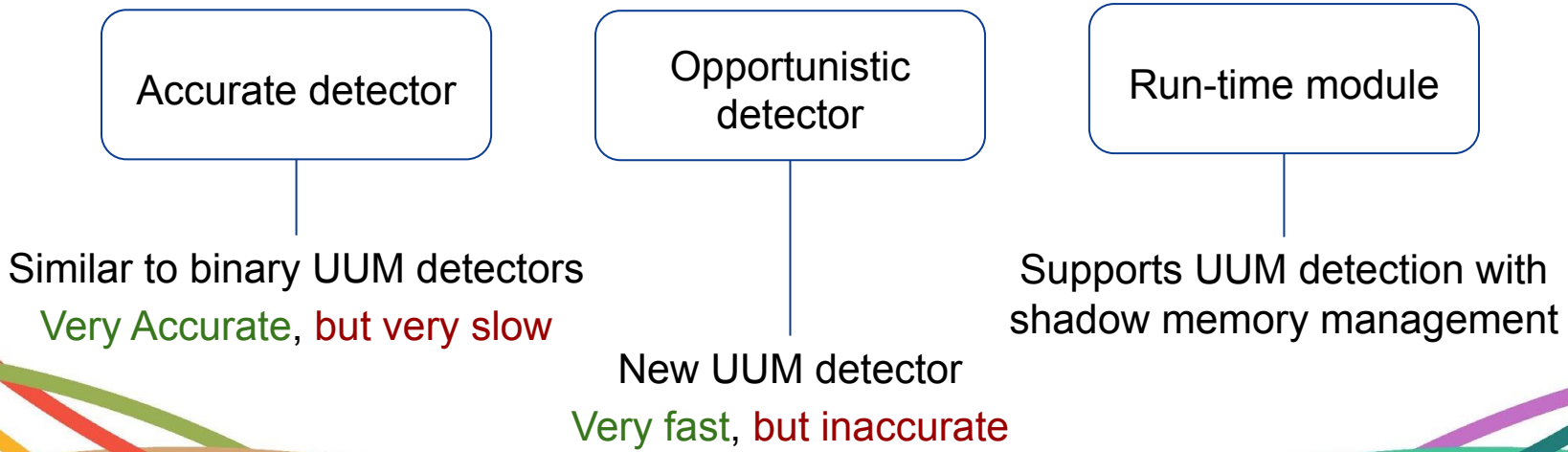
Cons:

- Slow (10-20x slowdown)
  - Shadow propagation is **much** harder
- No fuzzing compatibility

QMSan - overview

- Binary-based multi-layered solution to detect UUM errors
  - based on the QEMU emulator
  - fuzzing-compatible

- Three main components:



QMSan - Opportunistic Detector

Key intuition: Most loads of uninitialized memory are safe…
We don't need to check them every time!

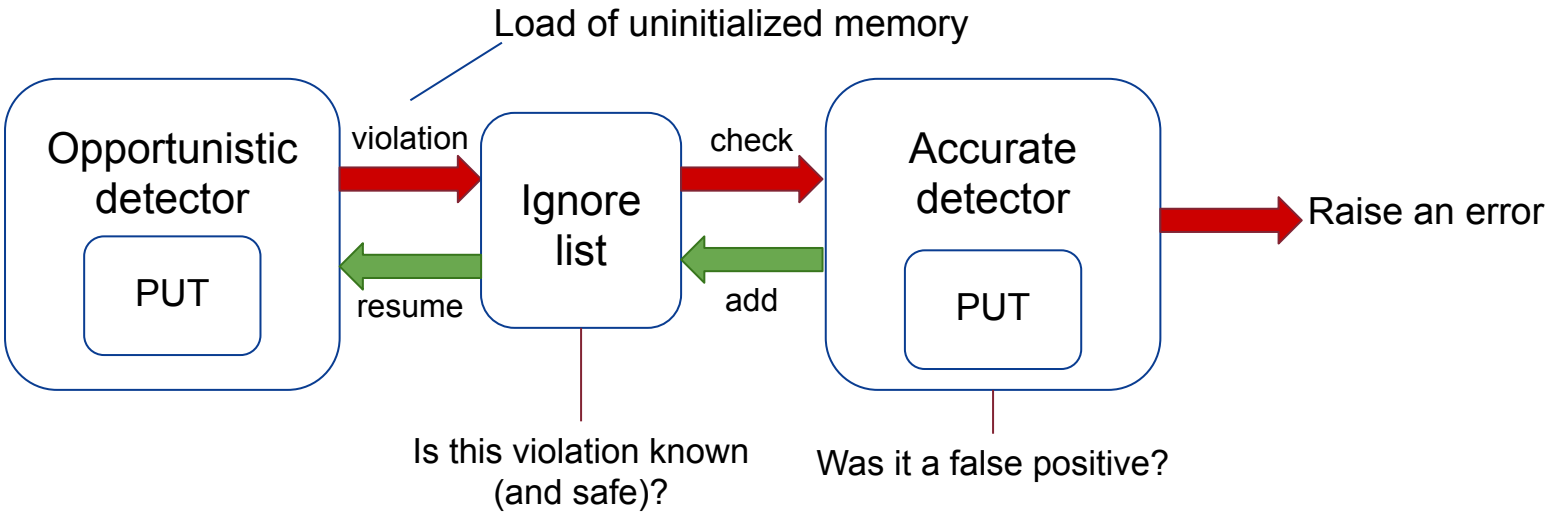
Opportunistic detection:

- Only check memory accesses (R/W)
  - Write: initialize shadow
  - Read: check shadow
- When a **violation** occurs:

Known: keep executing

Not Known: Use propagation to check and **remember for next time**

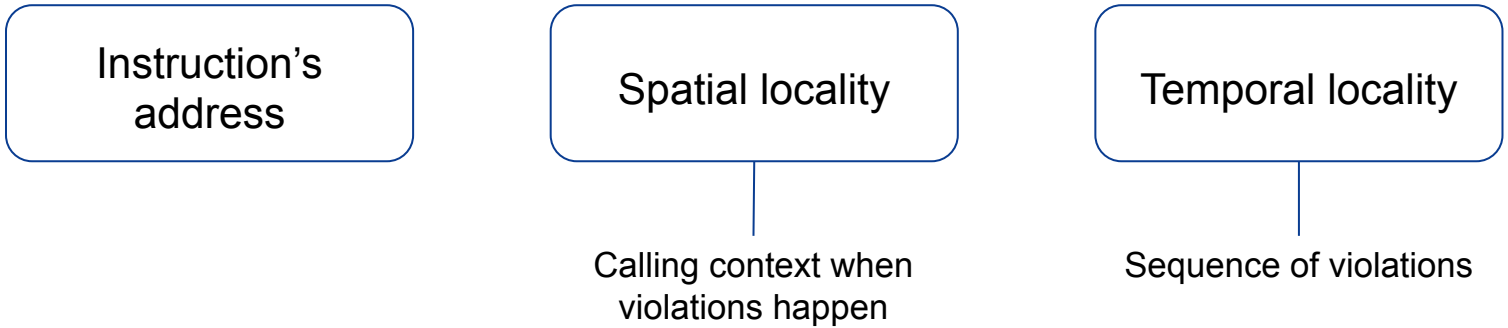
QMSan - Workflow



QMSan - Ignore list

- Used to remember violations
  - How to identify a violation?

- Three properties:



Evaluation - Bugs

Dataset:

- 9 closed-source binaries
  - 5 projects, multiple versions
- 10 open-source programs (from OSS-Fuzz)

Subject	Vendor	Version	Bugs
cuobjdump	NVIDIA	12.3	2
cuobjdump	NVIDIA	12.4	0
nconvert	XnView Software	7.136	5
nconvert	XnView Software	7.155	4
nvdisasm	NVIDIA	12.3	7
nvdisasm	NVIDIA	12.4	3
pngout	Ken Silverman	Jan 15 2020	2
rar	rarlab	6.11	1
rar	rarlab	7.0	3
Total			27

Methodology:

- 72 hour runs
- 3 runs

Subject	Version	Bugs
libredwg	763d702	3
gpac	205bfe3	1
assimp	b71b8f7	2
libdwarf	6178ba8	2
serenity	7914383	1
opensc	fe2c1c8	5
ntopng	8786f06	1
upx	3495d1a	2
radare2	cfe5806	0
libucl	5c58d0d	0
Total		17

Evaluation - Performance

Dataset:

- 8 common fuzzing benchmarks (from Google's FTS)

Methodology:

- 24 hour runs
- 3 runs

Project	QMSan		
Name	vs AFL-cc	vs MSan	vs QEMU
c-ares	2,20	1,05	1,04
guetzli	3,17	1,24	1,41
json	2,69	1,24	1,12
libxml2	3,41	0,90	1,42
openssl	19,84	8,24	4,68
pcre2	3,18	1,42	1,40
re2	3,35	1,48	1,48
woff2	2,86	1,34	1,20
geomean	3,75	1,55	1,51

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
SERICS SECURITY AND RIGHTS IN THE CYBERSPACE

Conclusions

- Detecting UUM errors is a challenging task that tends to either introduce **high slowdowns** or have **compatibility issues**.
- We presented a new design that **increases compatibility** with software while still incurring in a reasonable slowdown.
  - 44 new bugs (4 CVEs)
  - 1.51x slowdown over QEMU

https://github.com/Heinzeen/qmsan