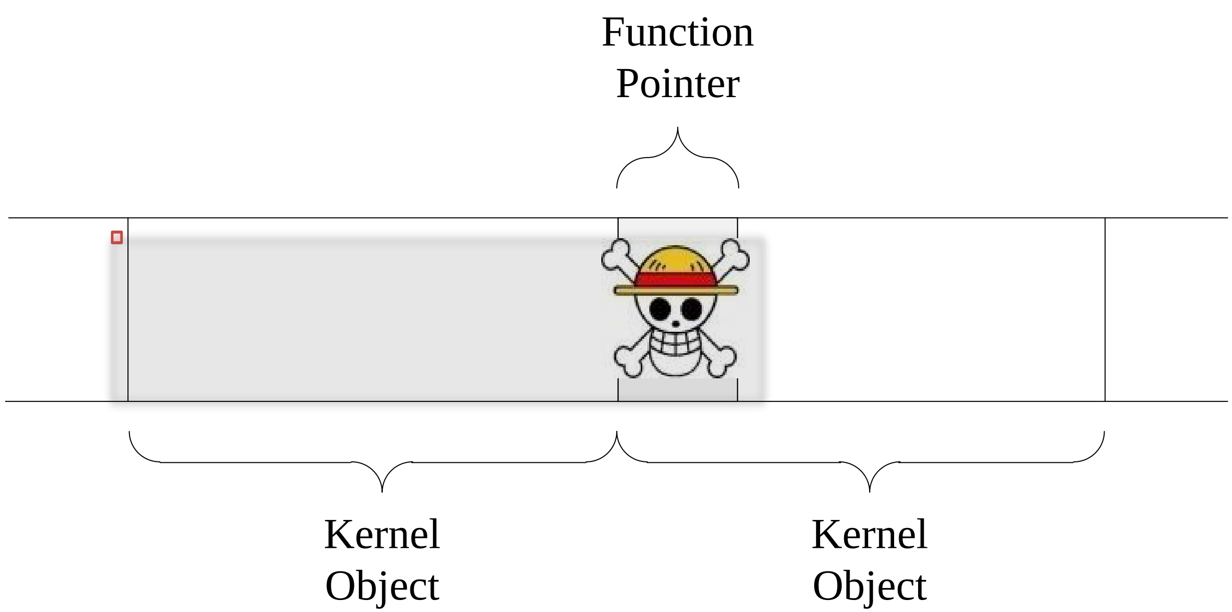
NDSS 2012

Kruiser: Semi-synchronized Non-blocking Concurrent Kernel Heap Buffer Overflow Monitoring

Donghai Tian[1,2], Qiang Zeng[2], Dinghao Wu[2], Peng Liu[2] and Changzhen Hu[1]

[1] Beijing Institute of Technology
[2] The Pennsylvania State University

Kernel Heap Buffer Overflow



Motivation

- There are more and more kernel buffer overflow exploits.

- To our knowledge, there are no practical mechanisms that have been widely deployed detecting kernel heap buffer overflows.

Current Methods: Limitations 1 & 2

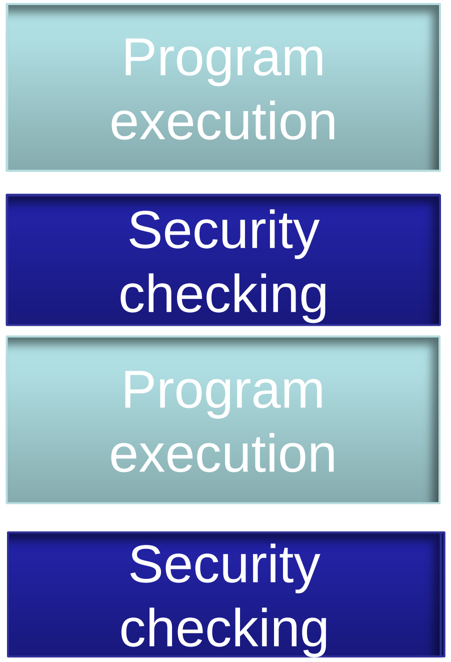
- Some approaches perform detection before each buffer write operation.
  [PLDI '04], [USENIX ATC '02], [NDSS '04]

High overhead!

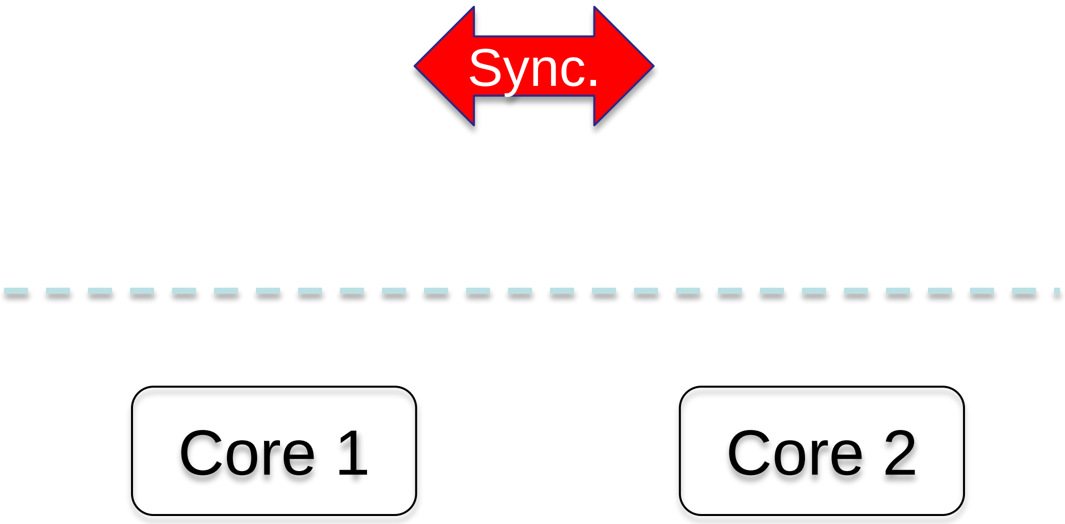
- Some approaches do not check heap buffer overflows until a buffer is de-allocated.
  [LISA '03], [BLACKHAT '11]

Large detection delay!

Our Idea



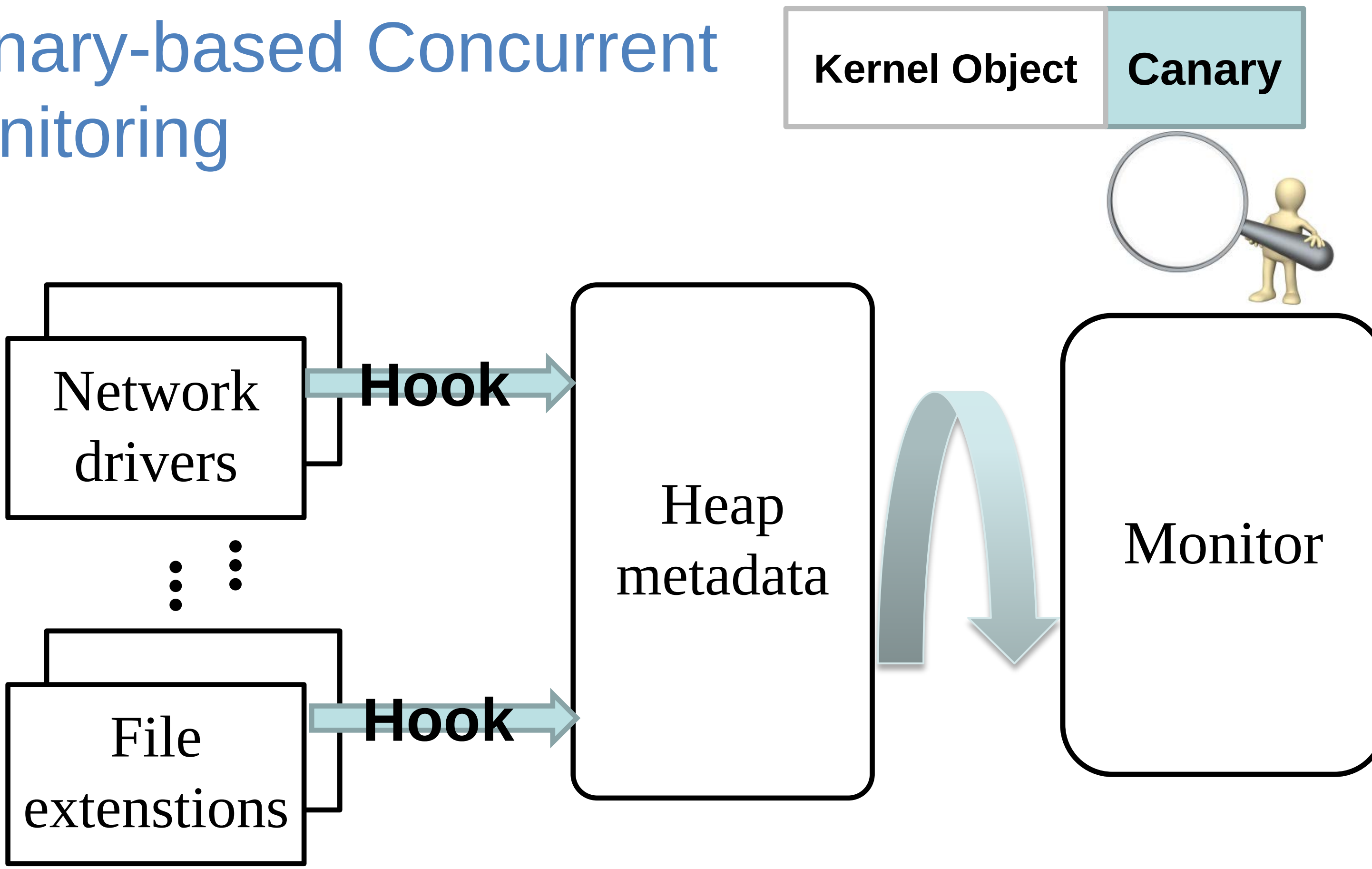
Inlined Checking



Concurrent checking

Basic Method

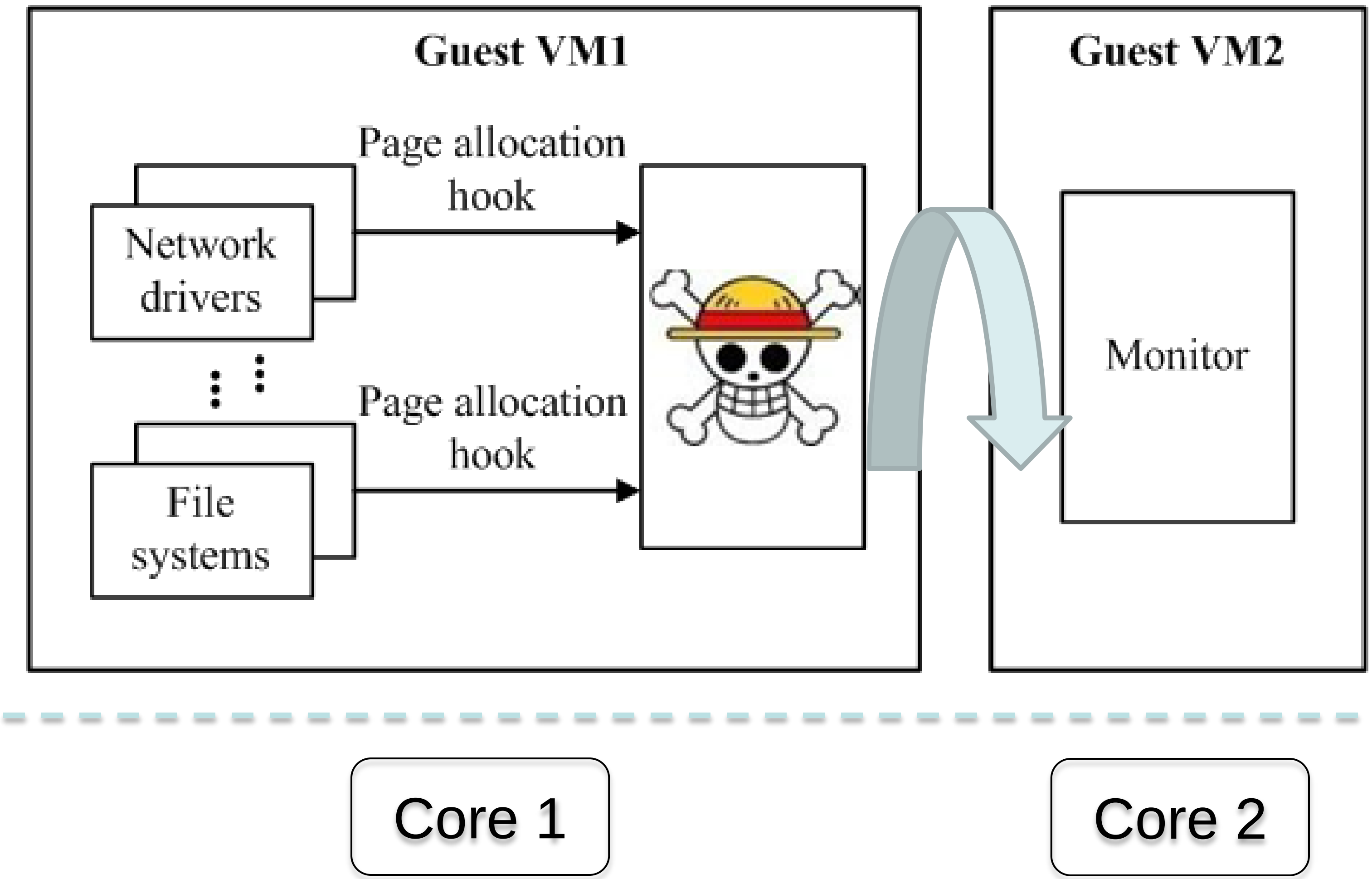
- Canary-based Concurrent Monitoring



Challenges

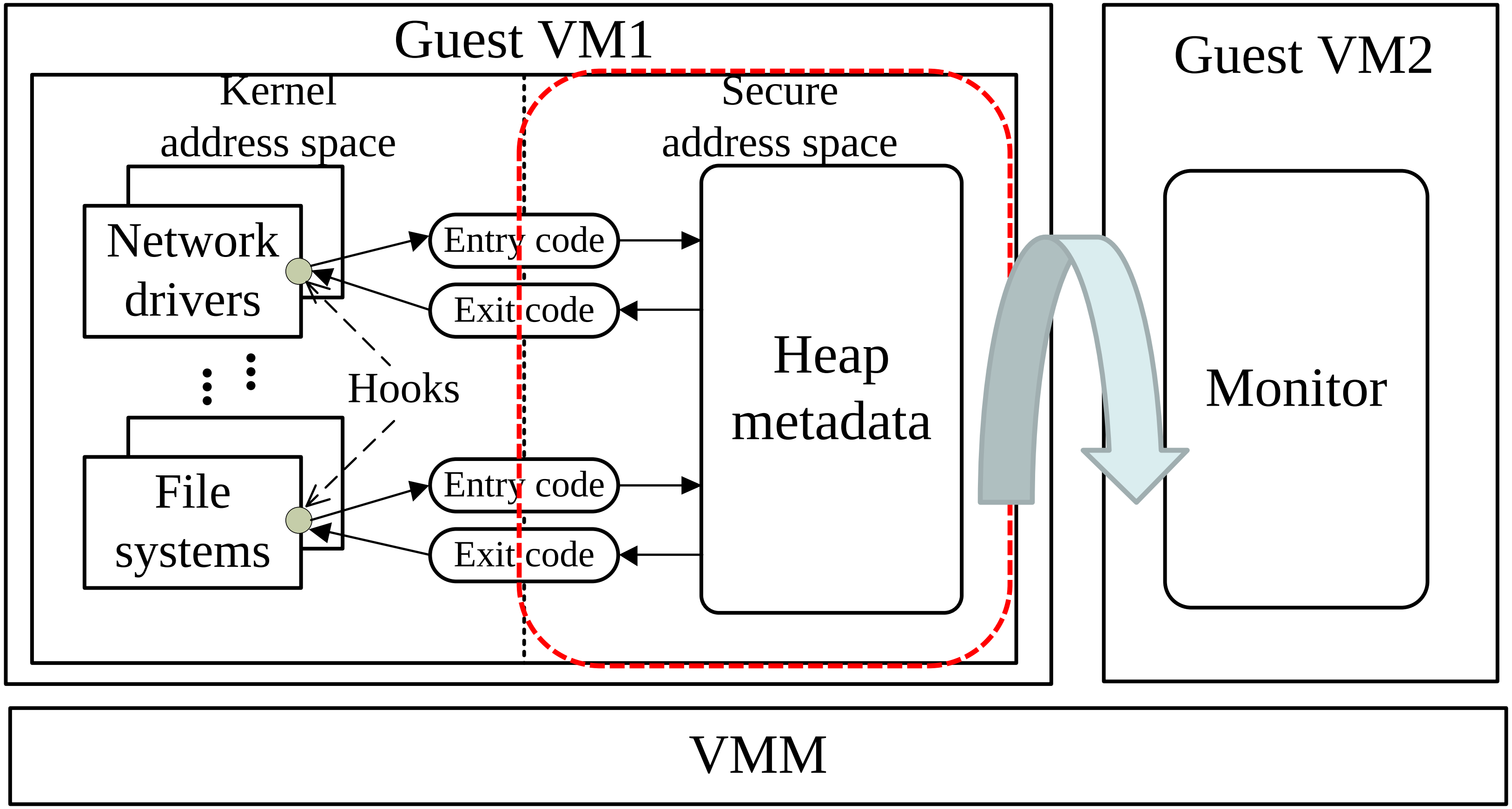
- Self-protection.
  - Monitor and the metadata
- Synchronization.
  - Races between hooks and monitor
- Compatibility.
  - OS and hardware

Out-of-the-VM Architecture
(Our previous CCS submission - rejected)



Hybrid VM monitoring Architecture

(*NDSS submission - accepted*)



Now, Kernel Cruising

- How to gather canary location info?
- How to deal with the races between hooks and monitor?

Kernel Cruising

- Page Identity Array (PIA)
  - Heap buffer canary location information
  - Other information

- Race conditions
  - Concurrent updates by two hooks
  - Inconsistent reads by monitor
  - Time of check to time of use (TOCTTOU)

Semi-synchronized Non-blocking Cruising Algorithm

- Avoid Concurrent Entry Updates.
  - Put the PIA entry update operations into the critical section.

Resolve TOCTTOU

Hook:

if the page is moved to the heap page pool

flag = true;

else if the page is removed from the heap

flag = fal

true->false->true

A B A

Monitor:

if (the canary is ter ered) {

if (flag == true) { // the page is still in heap

report overflow!

}

ABA Hazard Solution

```
if the page is moved to the heap page pool
        version++;
else if the page is removed from the heap
        version++;

…
if (the canary is tampered) {
        if (version == original version) {
            report overflow!
        }
}
```

Secure Canary Generation

- R1) The canaries are not predictable.
- R2) The canary generation and verification algorithms should be efficient.
- Generate unpredictable canaries using RC4 from a per-virtual-page random value.

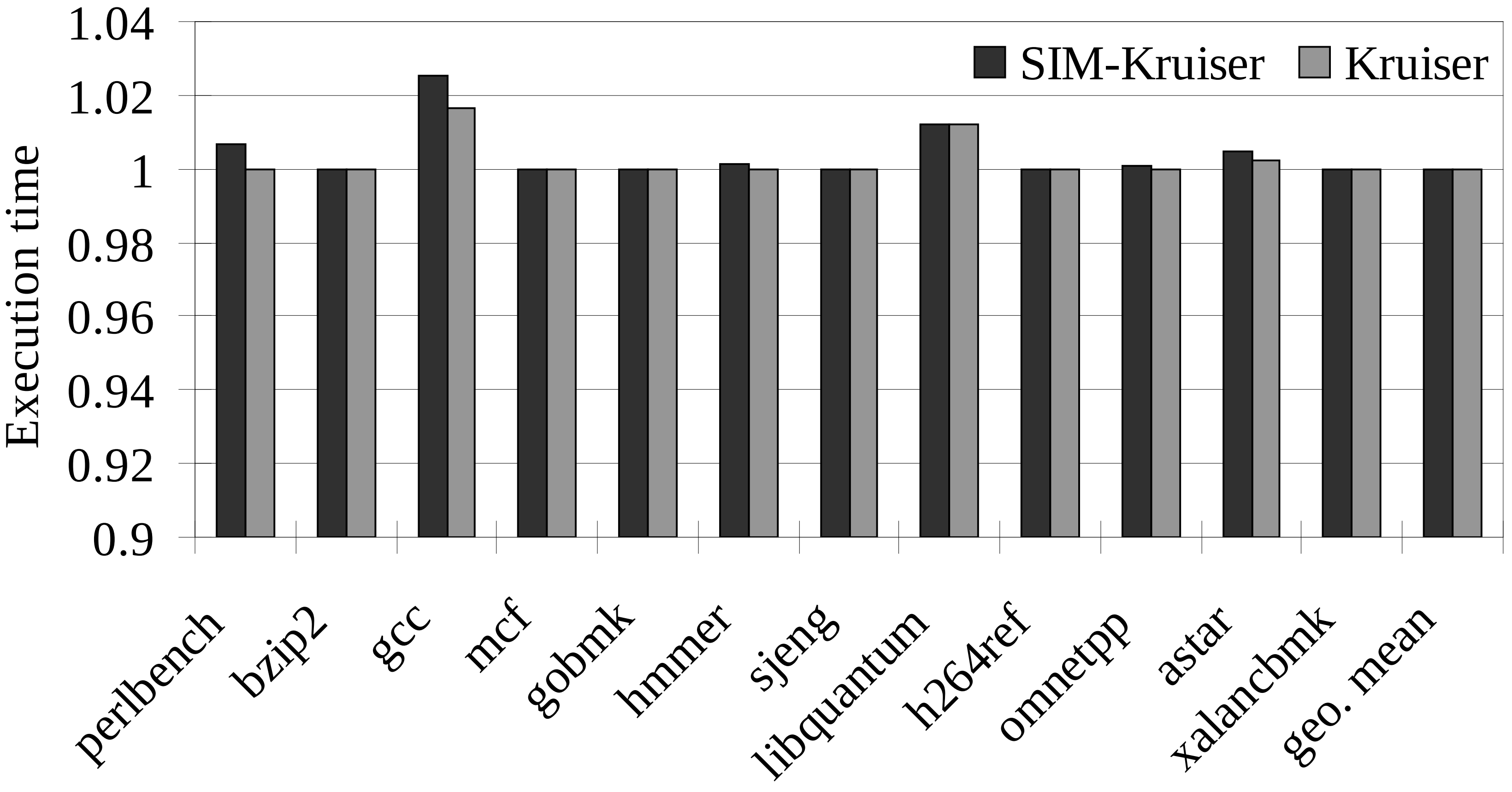
Outline

- Idea
- Architecture
- Kernel Cruising
- **Evaluation**
- Related Work
- Summary

Effectiveness

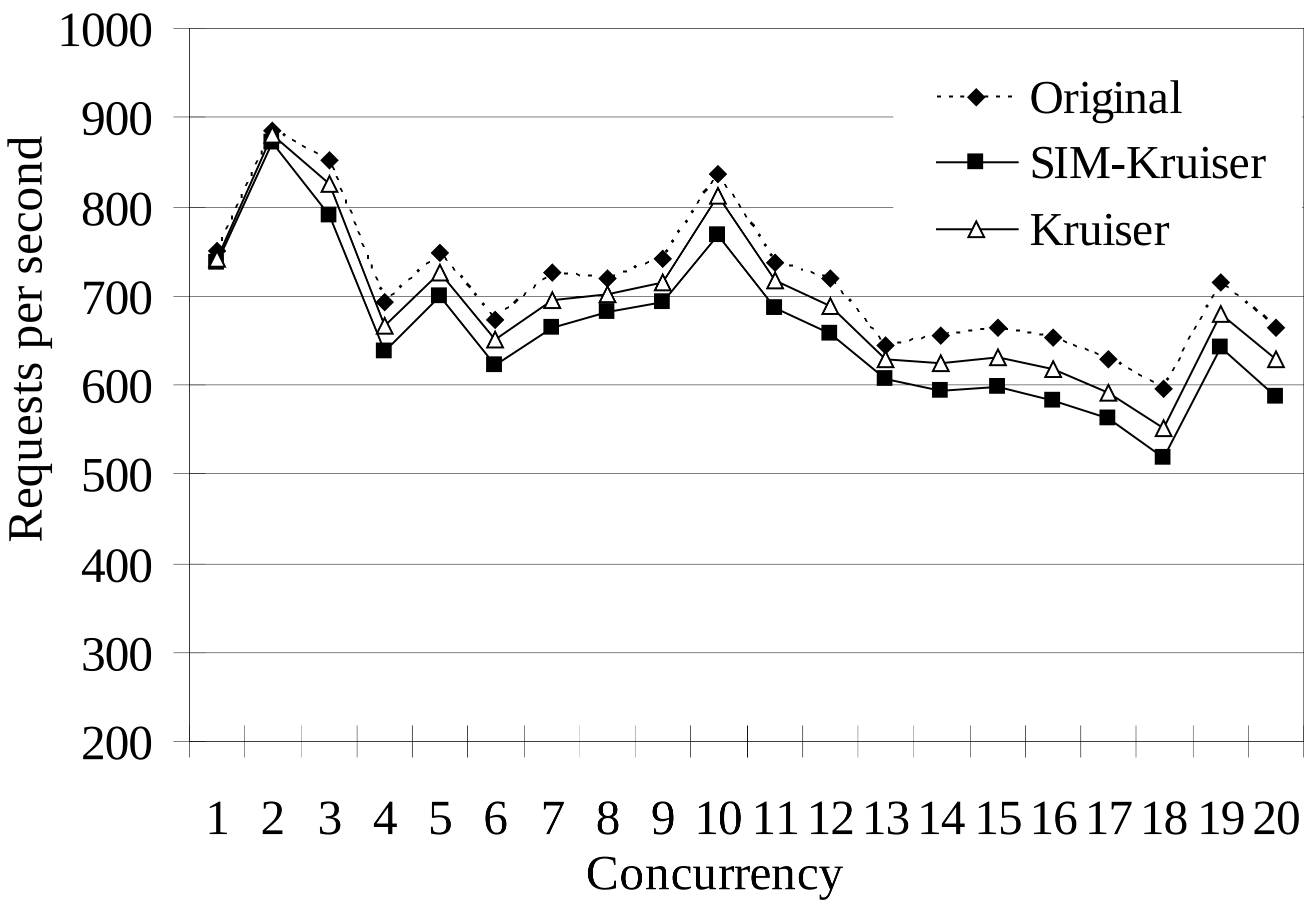
- We exploited five heap buffer overflow vulnerabilities in Linux, including three synthetic bugs and two real world vulnerabilities .
- All the overflows are successfully detected by *Kruiser*.

Performance Overhead



SPEC CPU2006 performance (normalized to the execution time of original Linux).

Scalability



Throughput of the Apache web server for varying numbers of concurrent requests.

Detection Latency

Different cruising cycle for different applications in the SPEC CPU2006 benchmark

Benchmark	Maximum cruising number	Minimum cruising number	Average cruising number	Average cruising cycle(μs)
perlbench	107,824	105,145	106,378	39,259
bzip2	79,085	76,325	76,682	27,662
gcc	78,460	76,810	77,413	27,774
mcf	82,885	79,328	79,540	28,156
gobmk	80,761	80,345	80,519	28,606
hmmer	81,278	80,435	80,591	28,635
sjeng	81,437	80,259	80,535	28,610
libquantum	80,911	80,317	80,407	28,493
h264ref	80,756	80,337	80,480	28,572
omnetpp	82,109	80,796	81,088	28,836
astar	81,592	81,022	81,097	28,897
xalancbmk	99,436	82,747	88,454	30,190

10 of 12 applications have less than 29ms (for scanning the kernel heap).

Outline

- Idea
- Architecture
- Kernel Cruising
- Evaluation
- **Related work**
- Summary

Related Work

- Countermeasures Against Buffer Overflows
  - StackGuard [USENIX Security '98]
  - Heap Integrity Detection [LISA '03]
  - Cruiser [PLDI '11]
  - DieHard [PLDI '06] and DieHarder [CCS '10]
- VM-based Methods
  - SIM [CCS '09]
  - OSck [ASPLOS '11]

Summary

- *Kruiser* can achieve *concurrent monitoring* against kernel heap buffer overflows.
  - *Non-blocking*
  - *Semi-synchronized*
  - *NO false positive*

- The *hybrid VM monitoring* scheme provides high efficiency without sacrificing the security guarantees.

Thank you!

Questions?

Outline

- Background and Idea
- Architecture
- Kernel Cruising
- Evaluation
- Related Work
- Summary

Non-blocking Cruising Algorithm

```
Monitor(){
uint ver1, ver2;
for (int page = 0; page < ENTRY NUMBER; page++){
    ver1 = PIA[page].version;
    if (The page is non-heap page)
        continue; // Bypass non−heap page
    Read the metadata stored in PIA[page];
    ver2 = PIA[page].version;
    if (ver1 != ver2)
        continue; // Metadata was updated
    for (each canary within the page){
        if (the canary is tampered){
            DoubleCheckOnTamper(page, ver1);
        }
    }
}
```

Avoid Read Inconsistency!

Is the page still used by the heap?