Practical Byte-Granular Memory Blacklisting using Californs

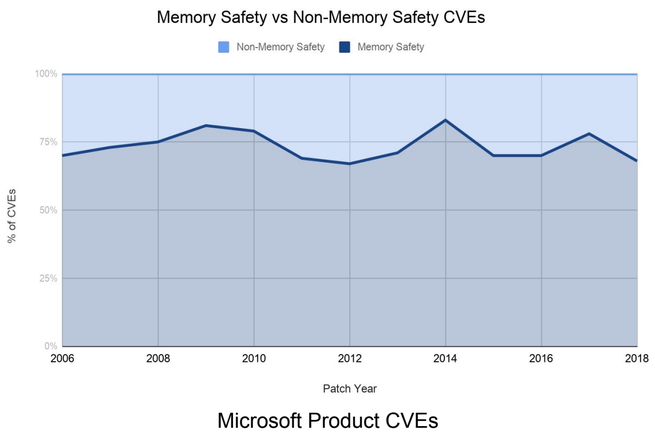
Hiroshi Sasaki, Miguel A. Arroyo, Mohamed Tarek Ibn Ziad,
Koustubha Bhat, Kanad Sinha, Simha Sethumadhavan
Columbia University

Video

Paper

THE PROBLEM

- Memory safety vulnerabilities are very easy for programmers to introduce unknowingly
- A need for a lower overhead and finer-grained (i.e. intra-object) level of memory safety



Microsoft Product CVEs

Source: *Microsoft Security Response Center (MSRC) - BlueHat 2019*

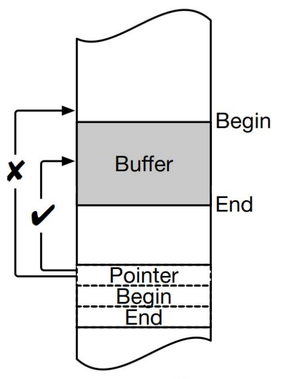
BACKGROUND

Disjoint Metadata Whitelisting

Metadata Overhead
2 words (base+size) per pointer

Memory Overhead
∝ # of pointers

Performance Overhead
∝ # of pointer dereferences

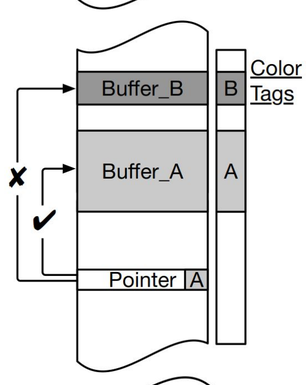


Cojoined Metadata Whitelisting

Metadata Overhead
256-bits per pointer

Memory Overhead
∝ # of pointers & phys mem.

Performance Overhead
∝ # of pointer operations

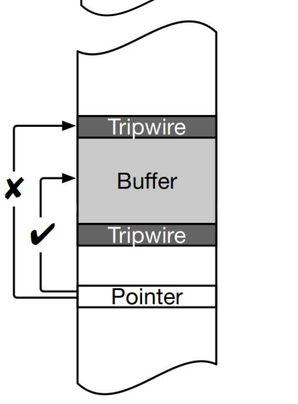


Inline Metadata Blacklisting

Metadata Overhead
1-bit per byte (naive)

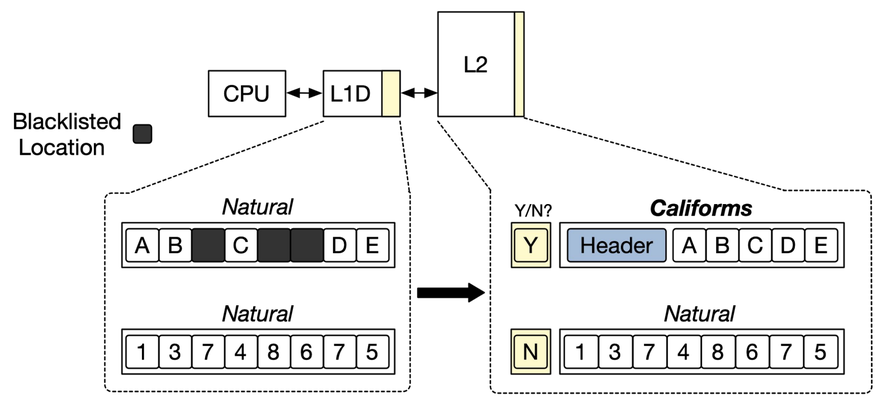
Memory Overhead
∝ blacklisted memory

Performance Overhead
∝ # of blacklist instructions



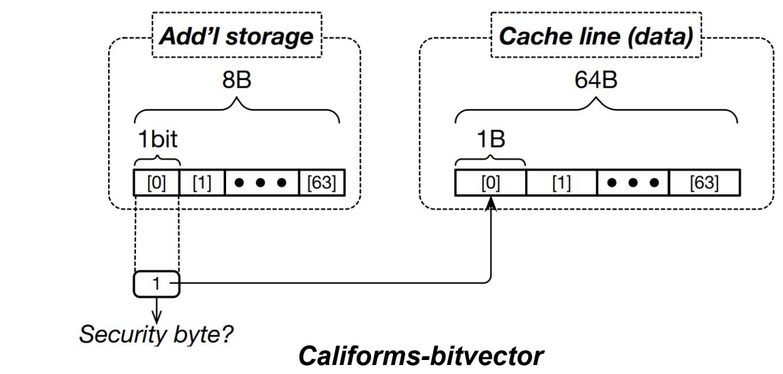
OUR CONTRIBUTIONS

Californs (Cache Line Formats)



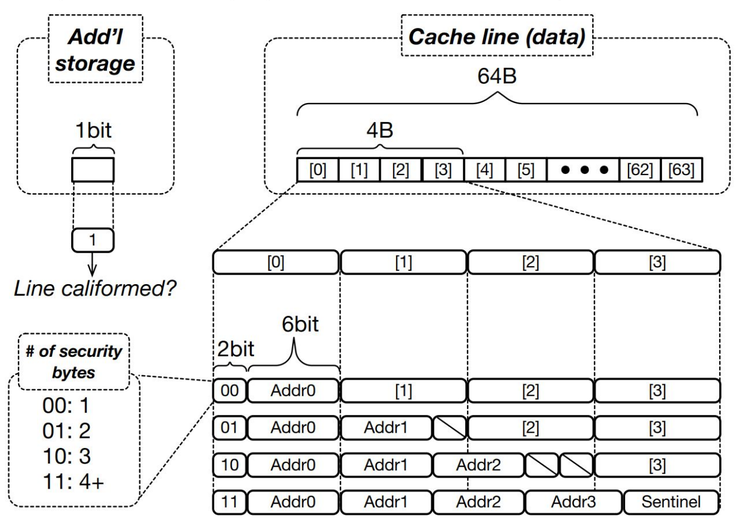
- Californs offers memory safety by detecting accesses to dead bytes in memory. Blacklisted locations are not stored beyond the L1 data cache and are identified using a special header in the L2 cache (and beyond) resulting in very low overhead.
- The conversion between these formats happens when lines are filled or spilled between the L1 and L2 caches. The absence of blacklisted locations results in the cache lines stored in the same natural format across the memory system.

Californs Schemes



Californs-bitvector

L1 Californ implementation using a bit vector that indicates whether each byte is a security byte. HW overhead of 8B per 64B cache line.



Californs-sentinel

L2+ Californ implementation that stores a bit vector in security byte locations. HW overhead of 1-bit per 64B cache line.

SECURITY BENEFITS

- Provides intra-object (i.e. field level) memory safety at low overheads (~1.02x-1.16x)
- Californs is agnostic of architecture width and can be deployed over a diverse device environment

Security Policies

```
struct
A_opportunistic {
  char c;
  char tripw[3];
  int i;
  char buf[64];
  void (*fp)();
}
```

(1) Opportunistic

```
struct A_full {
  char tripw[2];
  char c;
  char tripw[1];
  int i;
  char tripw[3];
  char buf[64];
  char tripw[2];
  void (*fp)();
  char tripw[1];
}
```

(2) Full

```
struct
A_intelligent {
  char c;
  int i;
  char tripw[3];
  char buf[64];
  char tripw[2];
  void (*fp)();
  char tripw[3];
}
```

(3) Intelligent

HARDWARE

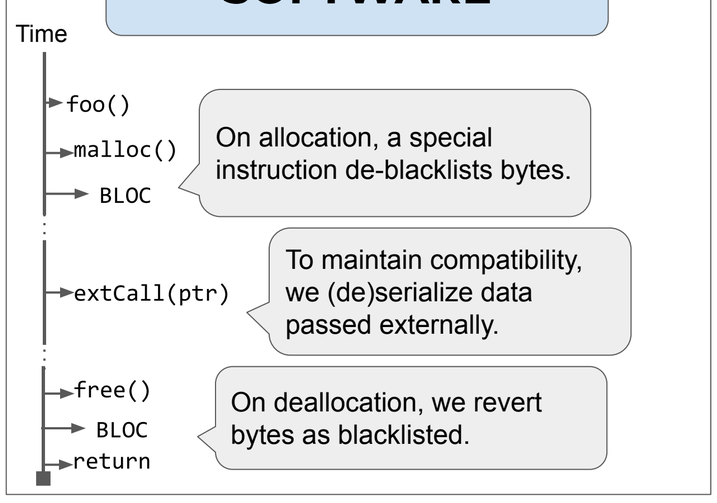
L1⇒L2 Conversion Algorithm

```
1: Read the Californs metadata for the evicted line and OR them
2: if result is 0 then
3:     Evict the line as is and set Californs bit to zero
4: else
5:     Set Californs bit to one
6:     if num security bytes (N) < 4 then
7:         Get locations of first N security bytes
8:         Store data of first N bytes in locations obtained in 7
9:         Fill the first N bytes based on Figure 7
10:    else
11:        Get locations of first four security bytes
12:        Scan least 6-bit of every byte to determine sentinel
13:        Store data of first four bytes in locations obtained in 11
14:        Fill the first four bytes based on Figure 7
15:        Use the sentinel to mark the remaining security bytes
16:    end
17: end
```

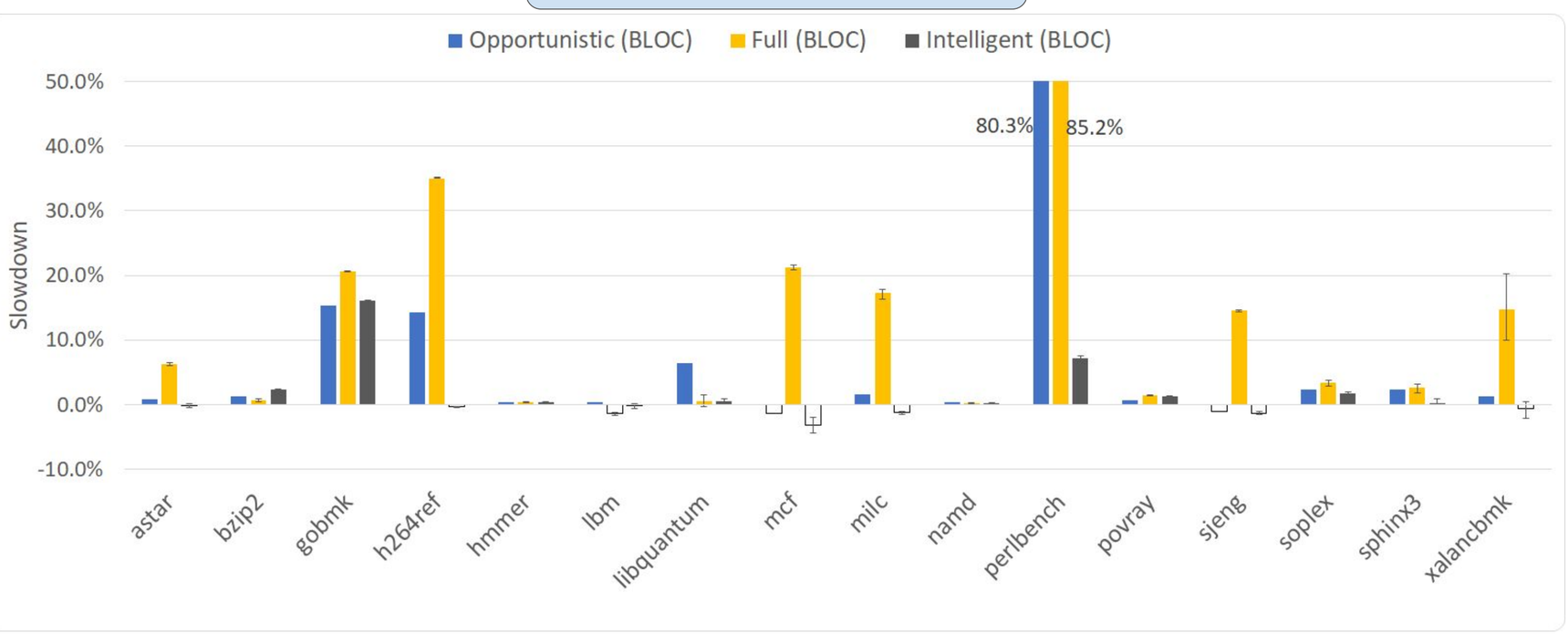
L2⇒L1 Conversion Algorithm

```
1: Read the Californs bit for the inserted line
2: if result is 0 then
3:     Set the Californs metadata bit vector to [0]
4: else
5:     Check the least significant 2-bit of byte 0
6:     Set the metadata of byte[Addr[0-3]] to one based on 5
7:     Set the metadata of byte[Addr[byte==sentinel]] to one
8:     Set the data of byte[0-3] to byte[Addr[0-3]]
9:     Set the new locations of byte[Addr[0-3]] to zero
10: end
```

SOFTWARE



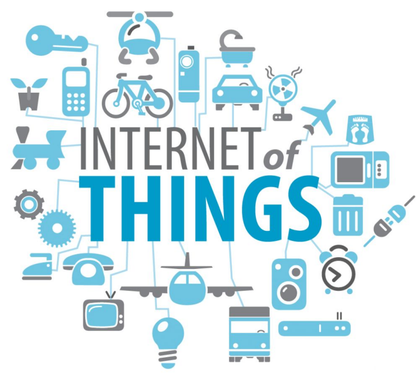
RESULTS



Blacklisting Overheads

IMPACT

- Scalable
  - Architecture width agnostic
  - Low overheads



- Californs have applications other than memory safety
  - Information Flow Tracking