```
[<c219ec5f>] security_sk_free+0xf/0x
[<c2451efb>] __sk_free+0x9b/0x120
[<c25ae7c1>] ? _raw_spin_unlock_irqr
[<c2451ffd>] sk_free+0x1d/0x30
[<c24f1024>] unix_release_sock+0x174
```

SlimGuard: A Secure and Memory Efficient Heap Allocator

Artifacts Available acm

Artifacts Evaluated Functional acm

Beichen Liu, Pierre Olivier, Binoy Ravindran
ECE, Virginia Tech

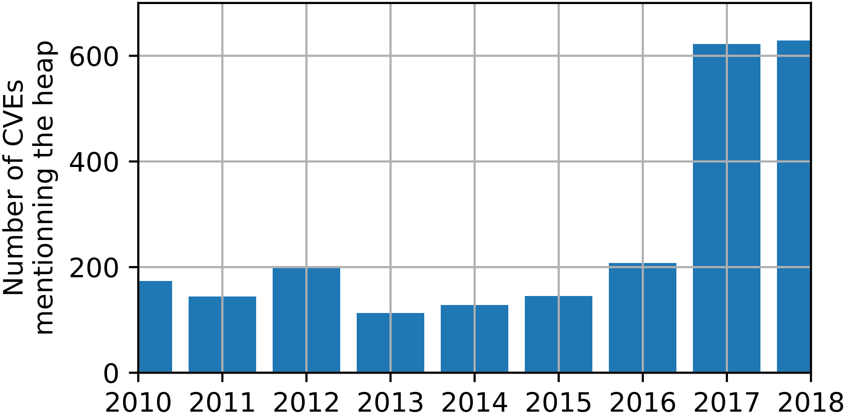
12/11/2019

ece Bradley Department of Electrical & Computer Engineering

VT VIRGINIA TECH™

Background and Motivation

Motivation

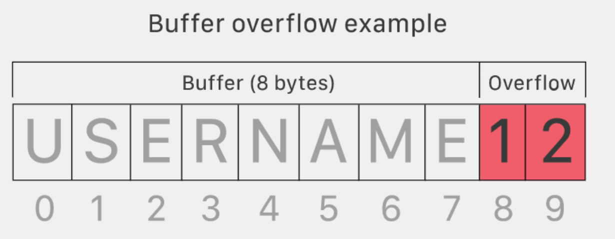


Number of heap-related CVE entries since 2010.

Background and Motivation

Heap Vulnerabilities

- buffer overflow, buffer overread



Buffer overflow example

- use-after-free/double free
- invalid free

Background and Motivation
Related Works

One solution for these heap vulnerabilities is to use a secure heap allocator, aka use a secure malloc algorithm.

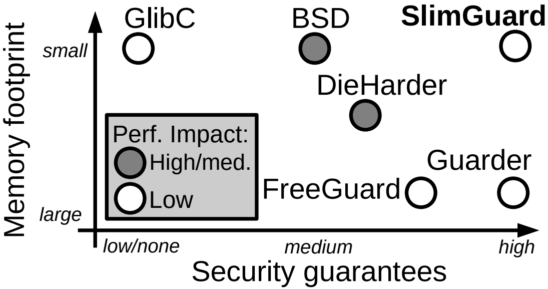
- OpenBSD
  - segragation of metadata
  - randomized allocation
  - security guarantees are not competitive
- DieHarder(PLDI'06)
  - randomized allocation
  - over-provisioning
  - memory overhead is large
- FreeGuard(CCS'17) and Guarder(USENIX'18)
  - high performance
  - high security guarantee
  - memory overhead is large

Background and Motivation

Related Works

Existing secure allocator are either not really secure or **not memory efficient**.

Design and Implementation



SlimGuard in the secure allocators design space.

Design Objective: **memory efficiency**, competitive security guarantees, and negligible performance overhead.

Design and Implementation

General Design

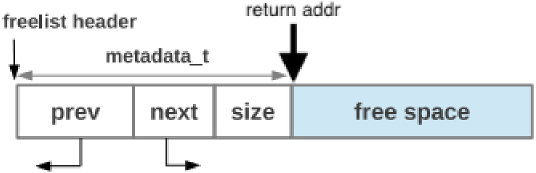
1. size class management
2. malloc path
3. free path
4. multithreading, binary compatibility

Design and Implementation

Size Classes Management

free-list(e.g., ptmallocv2)

- round size up to a multiple of 8 or 16 bytes
- store the size right before the heap object



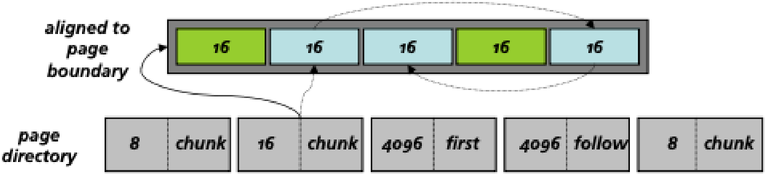
Inline metadata example

Design and Implementation

Size Classes Management

bibop style (e.g., OpenBSD, Guarder)

- power-of-2 class size
- a memory region are equally divided into classs size

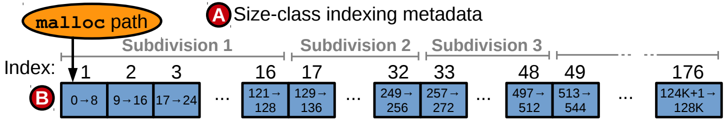


BIBOP heap example

Design and Implementation

SlimGuard Size Classed Management

SlimGuard: Fine-grained size class management

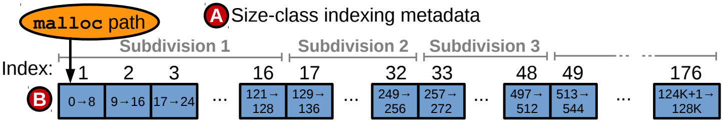


SlimGuard Design Space

Design and Implementation

SlimGuard Malloc Path

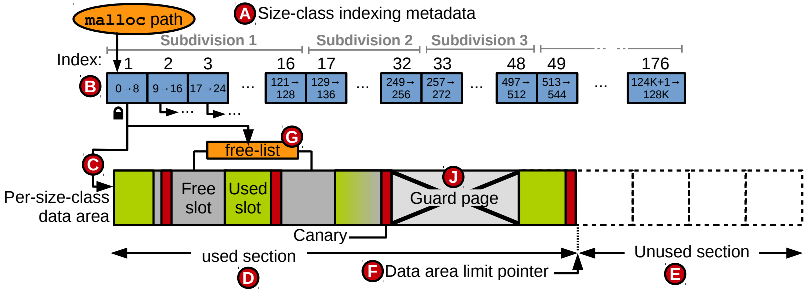
```
void *pointer = malloc(x);
```



SlimGuard Design Space

Design and Implementation
SlimGuard Malloc Path

```
void *pointer = malloc(x);
```

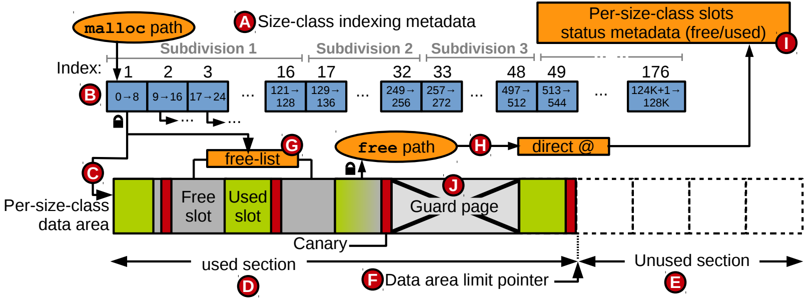


SlimGuard Design Space

Design and Implementation

SlimGuard Free Path

```
free(pointer);
```



SlimGuard Design Space

Design and Implementation

Other Features

- **Multithreading Support**
  thread-safe with fine-grained locks
- **Binary Compatibility**
  A binary Compatible dynamic library
  support malloc(), free(), realloc(), memalign()

Design and Implementation

Security Features

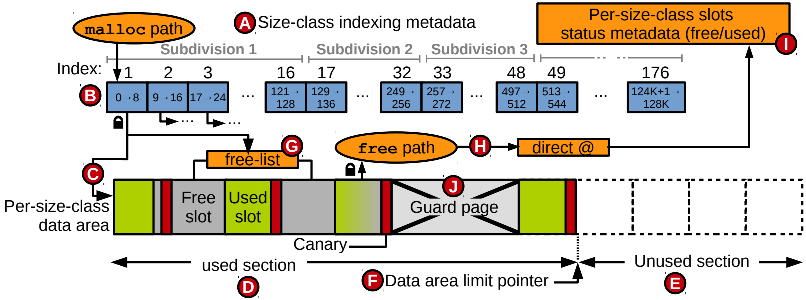
- randomization allocation
- metadata segradation
- dynamic canary
- guard page
- destroy-on-free, delayed memory reuse, etc.

Design and Implementation

Randomization Allocation

Goal: make it harder to guess where the next object is located

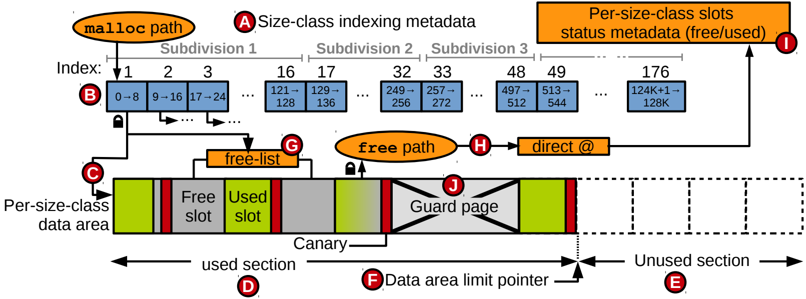
Solution: Build a large enough free array(G in figure)



SlimGuard Design Space

Design and Implementation
Metadata Segregation

Goal: prevent metadata to be overwritten
Solution: A separate region for metadata(A and B in figure)



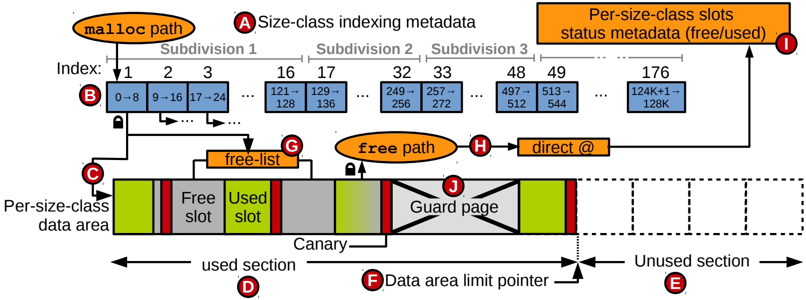
SlimGuard Design Space

Design and Implementation

Dynamic Canary

Goal: Detect buffer overflow

Solution: A hashed value based on return address(red block in data area)



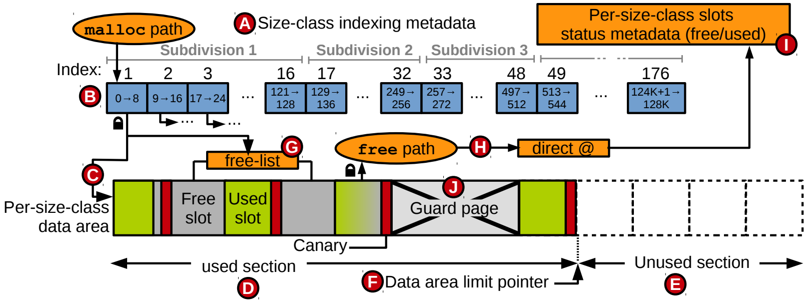
SlimGuard Design Space

Design and Implementation

Guard Pages

Goal: Detect buffer overflow immediately

Solution: fully on demand guard pages(J in figure)



SlimGuard Design Space

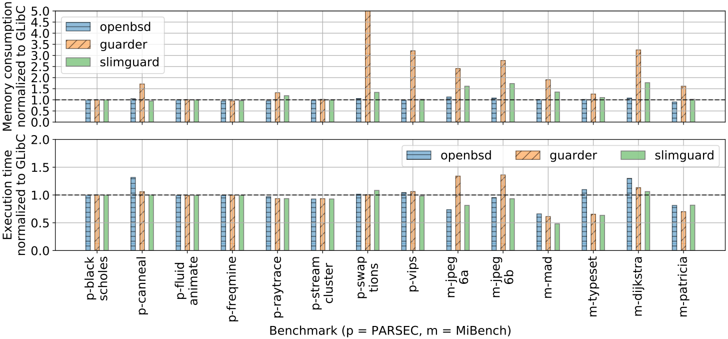
Design and Implementation

Other Security Features

- Destroy-on-Free
  zero out the free area
- delayed memory reuse
  a side-effect from our randomization

Results

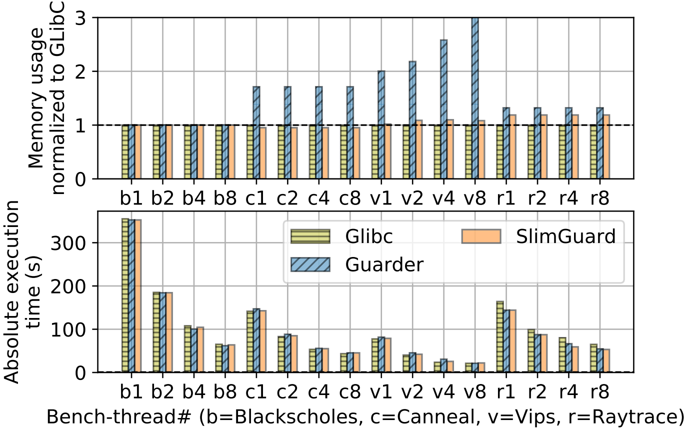
Performance and Memory Overhead Analysis



Memory footprint (top) and execution time (bottom) of SlimGuard, OpenBSD and Guarder for PARSEC (`p-*`) and MiBench (`m-*`) macro-benchmarks. All values are normalized to Glibc's memory footprint and execution time.

Results

Multitheading



Multithreading memory overhead (top) and performance (bottom) results.

Results

Security Analysis

SlimGuard's efficiency on real-world bugs

- gzip-1.2.4
- ncompress-4.2.4
- ed-1.14.1
- ImageMagic 7.0.4-1

SlimGuard successfully detects the exploit in all cases.

Conclusion

- Memory overhead is not acceptable in the state-of-the-art memory allocator
- Slimguard: memory-efficient and secure allocator
- Memory overhead is much smaller than state-of-the-art allocator

SlimGuard's Source Code

SlimGuard[1]is an open-source project of the Systems Software Research Group at Virginia Tech.
Source Available at **https://ssrg-vt.github.io/SlimGuard/**
Docker with benchmarks available at
https://hub.docker.com/repository/docker/bcliu430/slimguard

[1] SlimGuard is supported in part by ONR under grants N00014-16-1-2104 and N00014-18-1-2022.