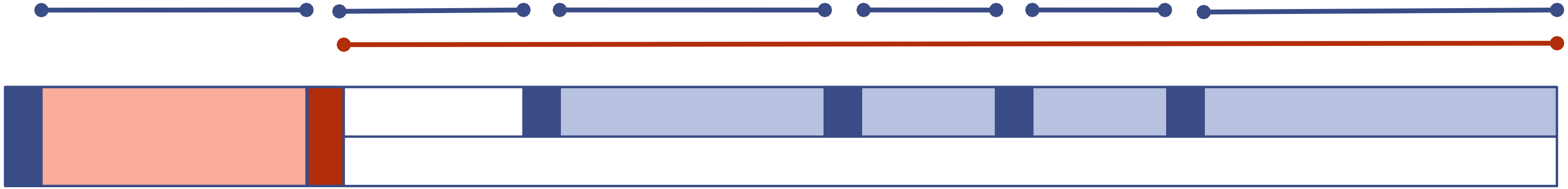
Heap Management

Recap

- Freelist-based allocators
- Bug elimination techniques:
  - Red zones
  - Poison values
  - Shadow memor
- Address Sanitizer

Freelists and overflows



- Attacker writes

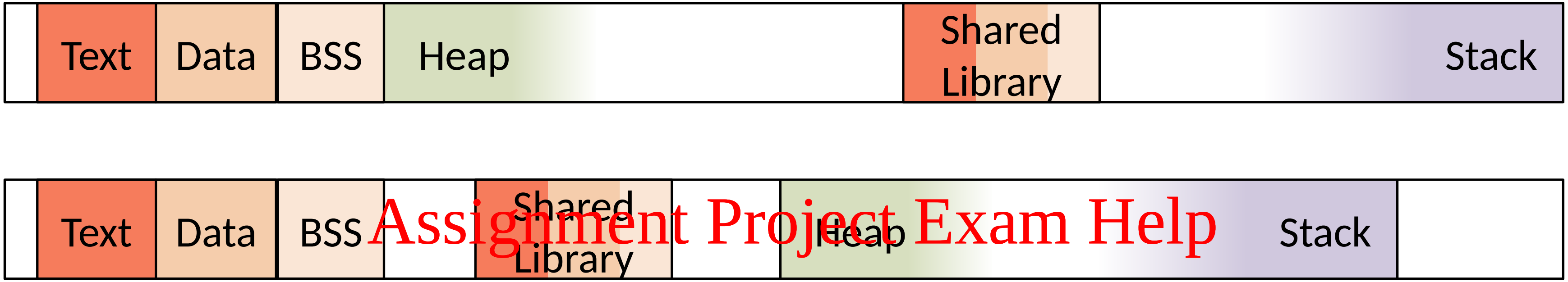
- Exploits a (sma verwrite the header of the next chunk

- Causing inconsistencies
- Which can be exploited

Securing the heap

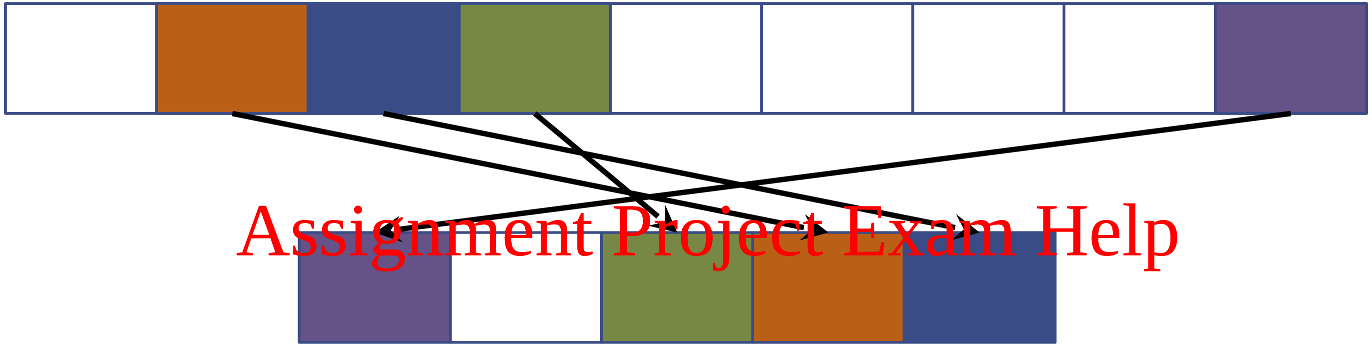
- Canaries in metadata
  - Detects (some) overflows
- Moving metadata to the shadow memory
  - Prevents exploi
- Randomise allocation
  - Avoids deterministic layout
- Use *guard pages*
  - Catches buffer overflows

Techniques – ASLR



- Address Space L
- Allocate segments of the proc m addresses
- The attacker does not know the virtual addresses of the data and code
    - But sometimes can learn it
    - Low entropy on 32 bit machines

OS Memory Management

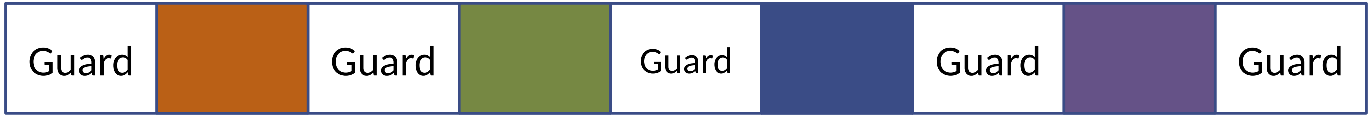


- The virtual address space co ed-size *pages*
- Pages map to physical *frames*
- Pages are associated with a *backing store*
  - Determines where contents is paged out to

The mmap interface

- Associates a virtual address with a backing store
- Program execution
  - Associates code
- Heap and stack
  - Associate pages with the swap
- Shared libraries
- sbrk

Guard pages



- Non consecutiv
  - Possibly at rand

- Limits overflow length

- No need to check for overflow
- ElectricFence
  - Allocates one object per page.

Techniques - BiBOP

- Big Bag of Pages
- Prevents exploitation (also used for performance)
- Dedicate a separate region of memory for each supported chun
- Use a separate hich chunk is available

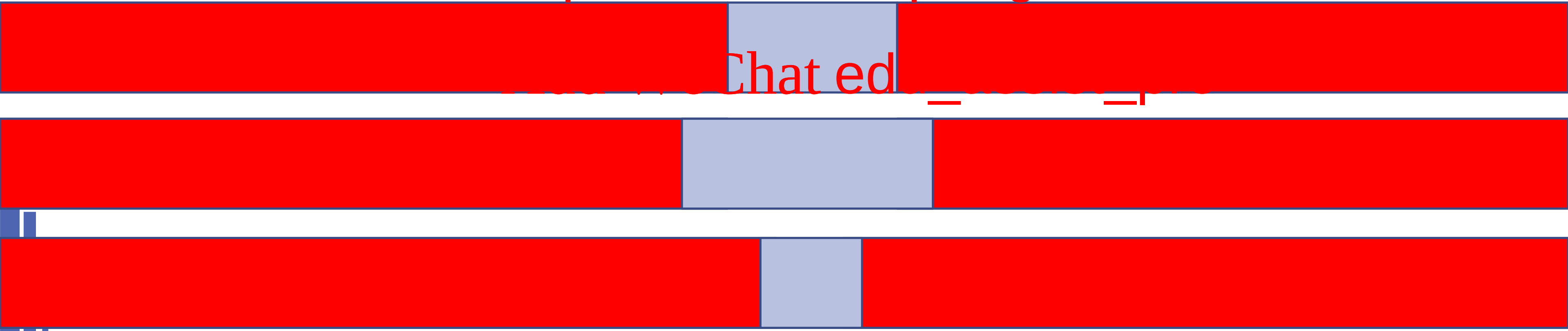
- Separates metadata from the malloc arena
  - Protects against metadata manipulation

Techniques - randomisation

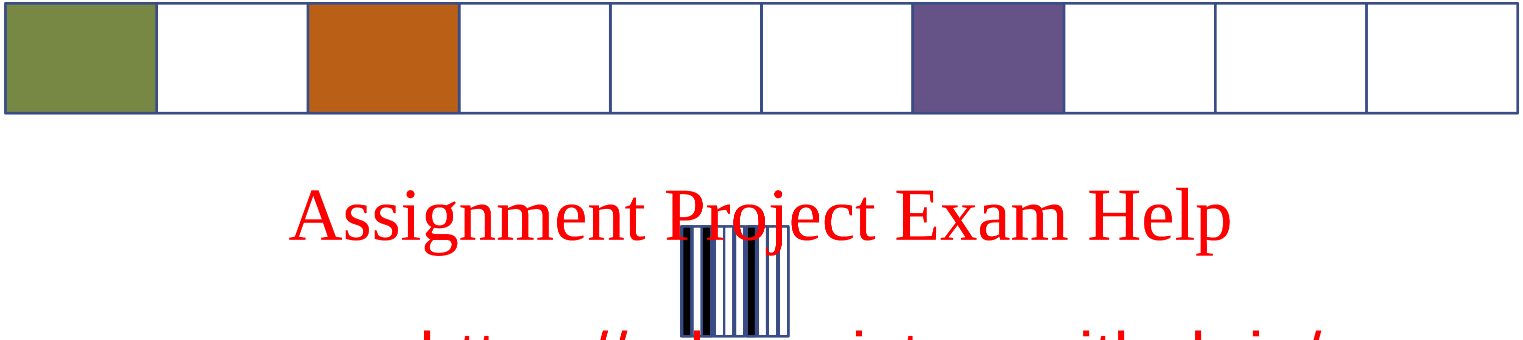
- Prevents exploitation – not good for debugging
- Address Space Layout randomization (ASLR) initialises the break at a random location.
- Randomise cho e
  - Easy with BiBOP
  - Limited support with freelist

Diehard – idea

- Heap structure that solves all memory bugs
- Allocate chunks with infinite length red zones
- Never free/reuse memory
- Secure, but the



Diehard – realisation



- Suppose we need M chunks
- Get space for αM chunks, for s
- A bit-map of size αM bits keeps track of free chunks.
- Allocation: pick a random free chunk – mark as used
- Free: mark as free