Google

GWP-ASan

Zero-Cost Detection of Memory Safety Bugs in Production

Matt Morehouse, Vlad Tsyrklevich, Mitch Phillips, Kostya Serebryany

October 2019

What is GWP-ASan?

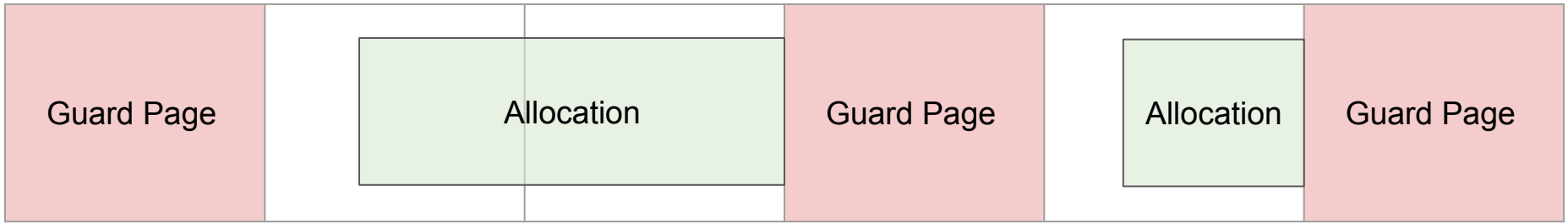
What is GWP-ASan?

- “**G**WP-ASan **W**ill **P**rovide **A**llocation **San**ity”

- Probabilistic memory safety error detector (heap only)
  - Detects heap buffer overflow, use-after-free.

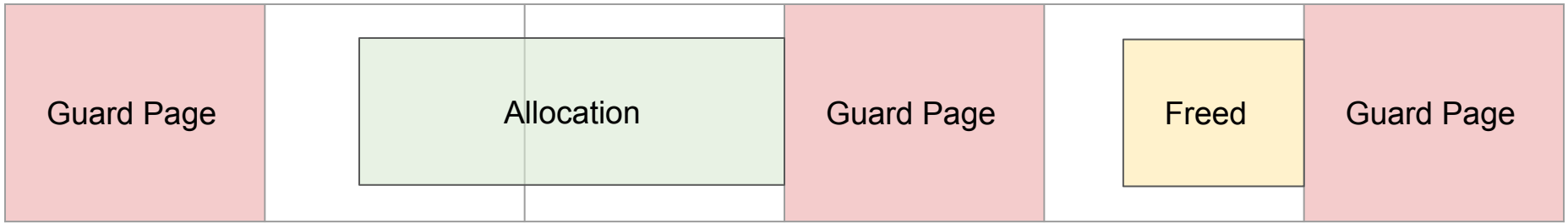
Background: Electric Fence

- Detects heap-buffer-overflows using **guard pages.**



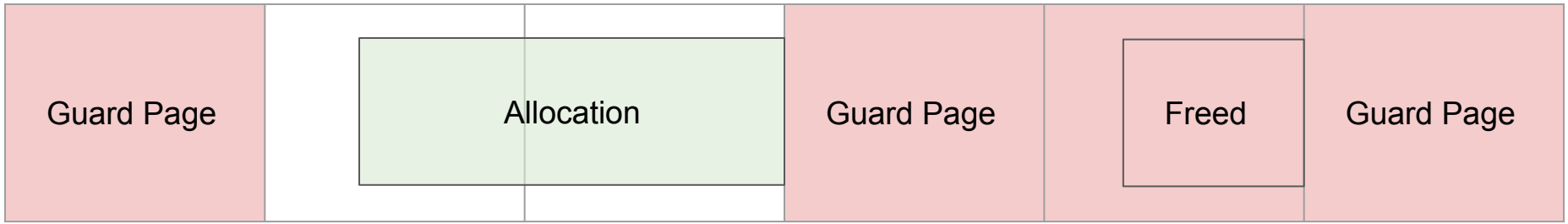
Background: Electric Fence

- Detects heap-buffer-overflows using **guard pages.**
- Detects use-after-frees by *mprotect*-ing freed memory**.**



Background: Electric Fence

- Detects heap-buffer-overflows using **guard pages.**
- Detects use-after-frees by *mprotect*-ing freed memory**.**



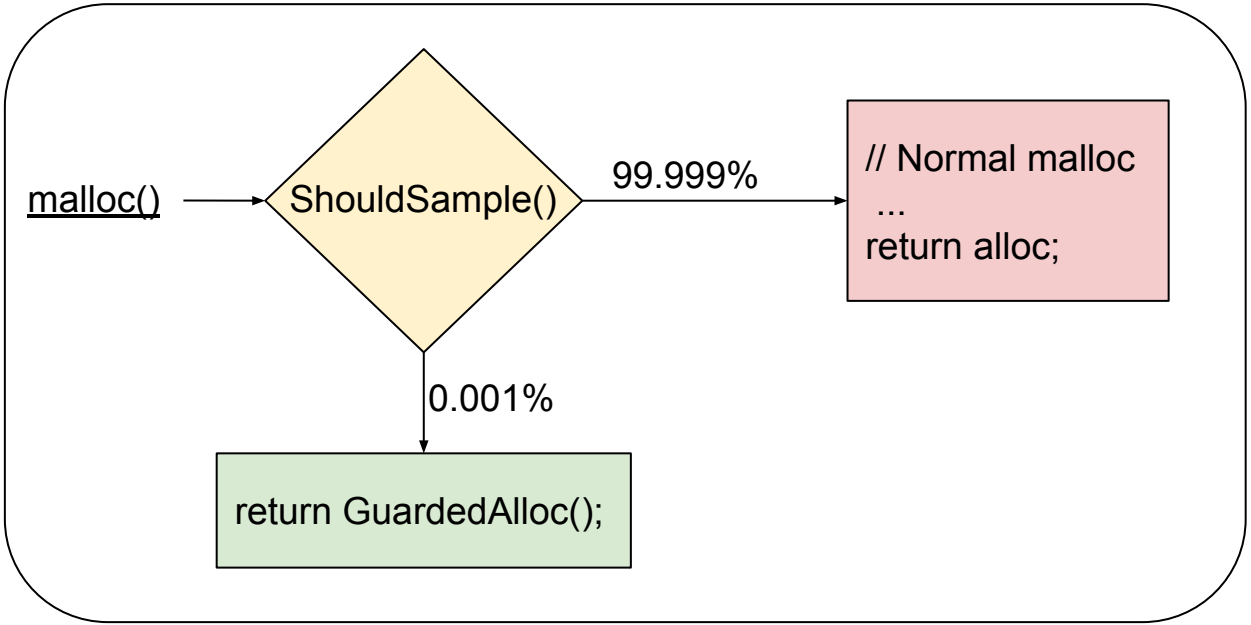
Background: Electric Fence

- + Detects bugs!
- + No compiler instrumentation.
  - Can be enabled at runtime.

- - Really expensive.
  - **Heap fragmentation (~100x)**
    - Each allocation needs a full 4KB page for buffer overflow detection.
  - **Slow (~100x)**
    - Most mallocs and frees require a system call to *mmap* or *mprotect*.

GWP-ASan = Electric Fence + Sampling

- Randomly guard a tiny fraction of allocations (e.g. 1/100,000).
  - Make overhead as low as we want.



GWP-ASan at Google

Deployment Status

- Chrome: **on-by-default**
  - for Windows and macOS only

- Google server-side applications: **on-by-default**

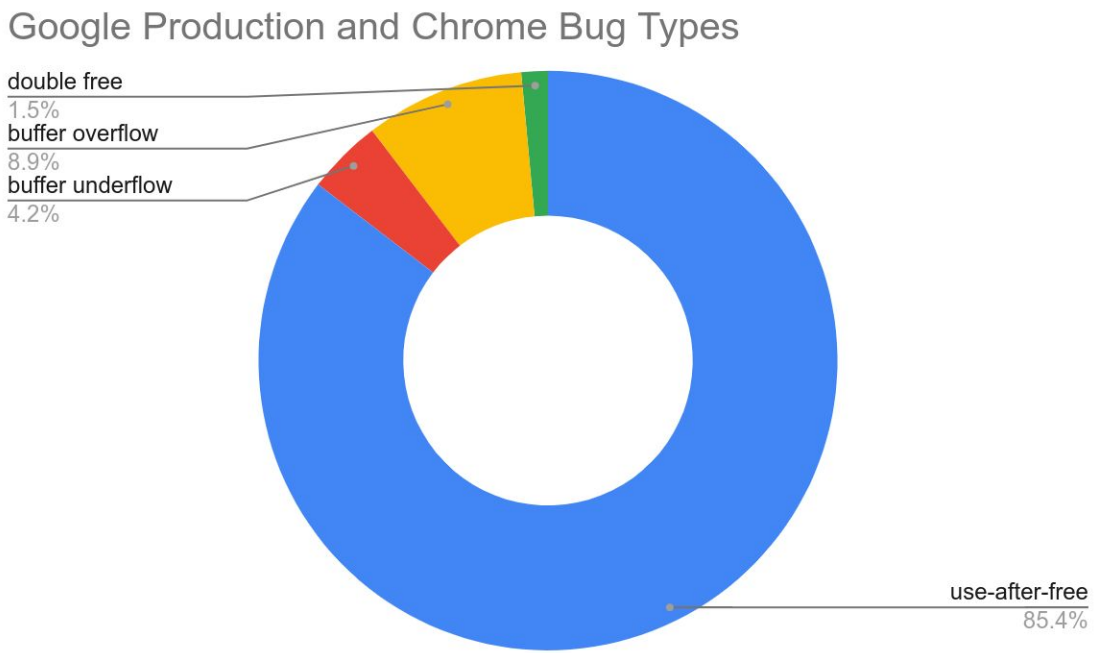
Results

- Chrome
  - 50+ bugs

    over past 10 months.

- Google Production
  - **400+ bugs**

    over past 10 months.



Using GWP-ASan

Available in LLVM

- GWP-ASan lives in compiler-rt.

- Comes with Scudo Hardened Allocator (*-fsanitize=scudo*)*
  - * x86/x86_64 only

- Simple integration with any other memory allocator.

Scudo Example

```
$ cat buggy_code.cc
#include <iostream>
#include <string>
#include <string_view>

int main() {
  std::string s = "Hellooooooooooooooo ";
  std::string_view sv = s + "World\n";
  std::cout << sv;
}
$ clang++ -g -std=c++17  -fsanitize=scudo buggy_code.cc && ./a.out
Hellooooooooooooooo World
$ for((i=0; i<1000; i++)); do GWP_ASAN_OPTIONS=SampleRate=500 ./a.out >/dev/null | symbolize.sh; done
*** GWP-ASan detected a memory error ***
Use after free at 0x7fb4b941e000 (0 bytes into a 41-byte allocation at 0x7fb4b941e000) by thread 140162 here:
  ...
  #9 /usr/lib/gcc/x86_64-linux-gnu/8.0.1/../../../../include/c++/8.0.1/string_view:547
  #10 /tmp/buggy_code.cpp:8

0x7f76bb8bafd0 was deallocated by thread 103932 here:
  ...
  #7 /tmp/buggy_code.cpp:8
```

Integrating with a Memory Allocator

```
static gwp_asan::GuardedPoolAllocator GuardedAllocator;

void initMalloc() {
  ...
  gwp_asan::options::Options Opts = ...  // Configure as desired.
  GuardedAllocator.init(Opts);
}

void *malloc(size_t Size) {
  ...
  if (PREDICT_FALSE(GuardedAllocator.shouldSample()))
    if (void *Ptr = GuardedAllocator.allocate(Size))
      return Ptr;
  ...
}

void free(void *Ptr) {
  ...
  if (PREDICT_FALSE(GuardedAllocator.pointerIsMine(Ptr)))
    return GuardedAllocator.deallocate(Ptr);
  ...
}
```

Thank You!