GWP-ASan: Sampling-Based Detection of Memory-Safety Bugs in Production

Kostya Serebryany (Google), Chris Kennelly (Google), Mitch Phillips (Google), Matt Denton (Google), Marco Elver (Google), Alexander Potapenko (Google), Matt Morehouse (Independent Researcher), Vlad Tsyrklevich (Independent Researcher), Christian Holler (Mozilla Corporation), Julian Lettner (Apple), David Kilzer (Apple), Lander Brandt (Meta)

ICSE SEIP, April 2024

Memory Safety in Programming Languages

- Memory-unsafe languages, specifically the C and C++ programming languages, define some well-typed programs to have undefined behavior
    - Memory-safe languages: no well-typed program has undefined behavior
- Heap buffer overflows and use-after-free accesses are two major bug classes introducing undefined behavior
    - Can result in anything from program crash (denial of service), data corruption, to an attackable exploit vector 🔥

Memory-safety bugs remain the single major source of security vulnerabilities:
70% of CVEs in Android, Chrome, and iOS are due to memory safety bugs.

Dynamic Memory-Safety Bug Detection

- Numerous **pre-production** dynamic analysis tools:
  - Valgrind
  - AddressSanitizer (and its variants)
  - ... and many more
- Hardware acceleration exists, but not (yet) widely deployed:
  - Arm Memory Tagging Extension (MTE)
  - SPARC ADI (legacy architecture)

- Electric Fence Malloc Debugger, introduced in 1987, one of the first *dynamic analysis* tools to detect memory safety bugs (more details later)

Electric Fence Malloc Debugger

Page-protection based addressability checks:

- Free object pages are protected, and unprotected after an allocation
  - Use-after-free results in page fault
- Object pages are surrounded by inaccessible "guard pages"
  - Buffer overflow results in page fault

Makes use of hardware feature available in all modern CPUs' Memory Management Units (MMUs): paged virtual memory and the ability to set memory pages inaccessible.

GWP-ASan: Memory-Safety Detection in Production

- *GWP-ASan adds an “if” statement to the* ***Electric Fence*** *algorithm*
- Finds heap-use-after-free and buffer-overflow errors
- Near zero performance overhead due to sampling:
  - very low probability of detecting a particular bug
  - needs to be deployed across a large fleet of machines
  - not a replacement for AddressSanitizer or other deterministic pre-production program analysis
- Better diagnostics compared to regular memory corruption
  - Accurate fault trace
  - Allocation and deallocation stack traces

The Name “GWP-ASan” is derived from Google-Wide Profiling (GWP), and AddressSanitizer (ASan). GWP-ASan is neither GWP nor ASan.

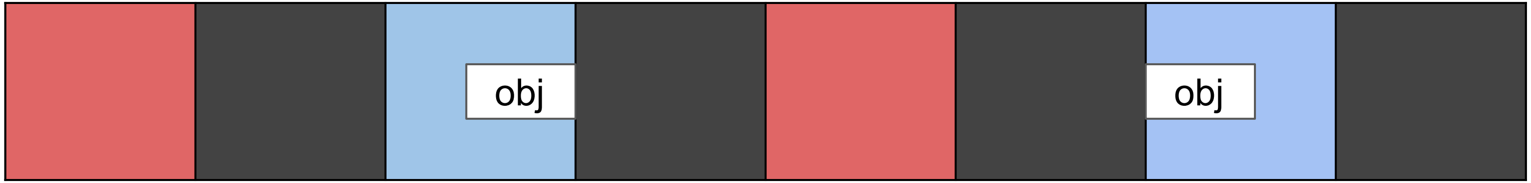
Implementation

Default Malloc + Sampling Electric Fence ⇒ GWP-ASan

```
void *malloc(size_t size) {


  return Allocate(size);
}

void free(void *ptr) {




  Deallocate(ptr);
}
```

```
  void *malloc(size_t size) {
+   if (WantToSample(size))
+     return GuardAlloc(size);
    return Allocate(size);
  }

  void free(void *ptr) {
+   if (IsGuarded(ptr)) {
+     GuardDealloc(ptr));
+     return;
+   }
    Deallocate(ptr);
  }
```

GWP-ASan Pool Layout



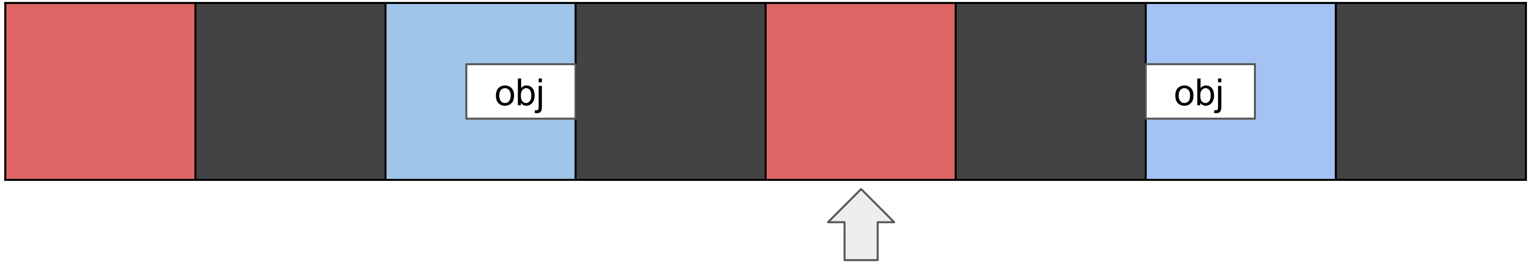
⬛ – Guard pages (`PROT_NONE`) ⇒ detect out-of-bounds accesses

🟦 – Active objects in unprotected pages (`PROT_READ|PROT_WRITE`)

🟥 – Free objects in protected page (`PROT_NONE`) ⇒ detect use-after-free

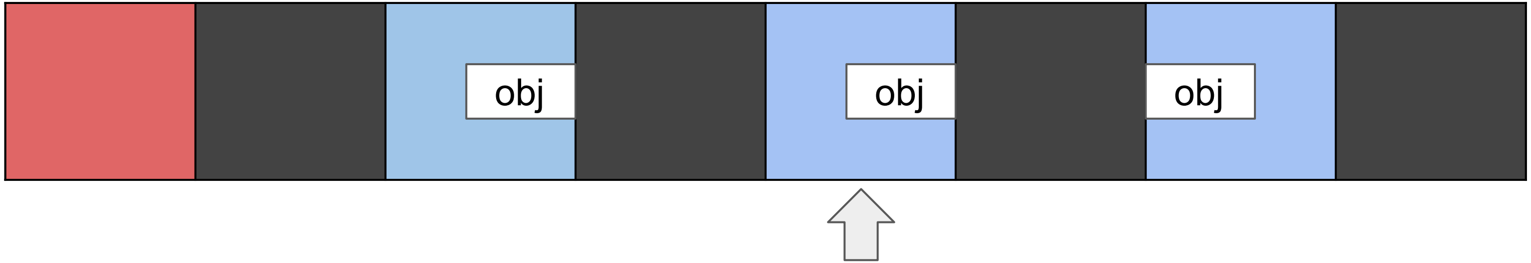
GWP-ASan: GuardAlloc()

Pick unused page…



GWP-ASan: GuardAlloc()

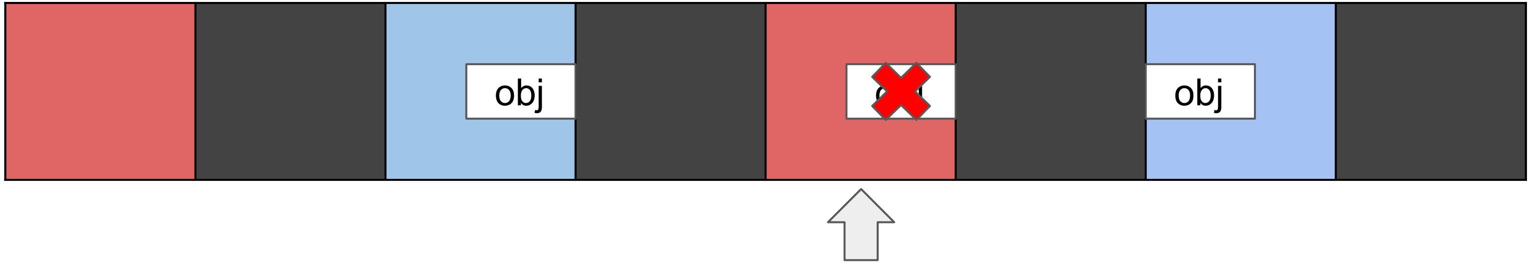
… unprotect it, and place the requested object at either end of the page:



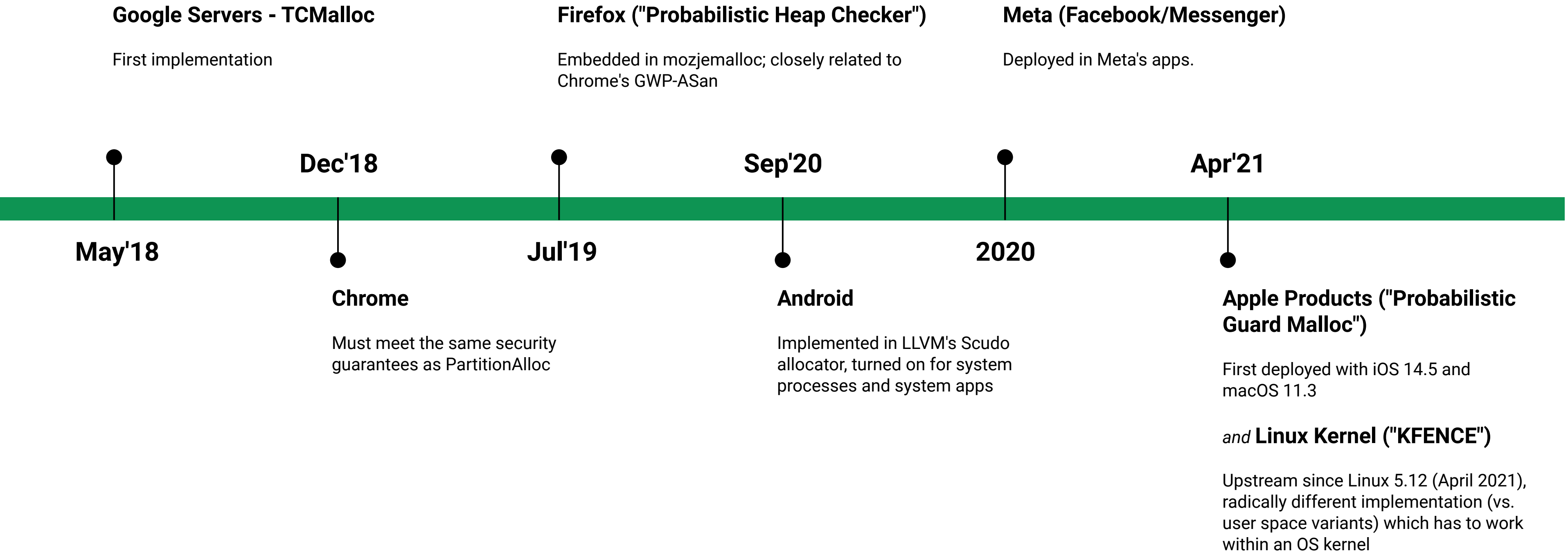
Depending on object placement (left or right), either overflow or underflow accesses will result in page faults. GuardAlloc() may randomly choose left or right placement.

GWP-ASan: GuardDealloc()

Protect the object page:



GWP-ASan Variants



May'18

Google Servers - TCMalloc

First implementation

Dec'18

Chrome

Must meet the same security guarantees as PartitionAlloc

Jul'19

Firefox ("Probabilistic Heap Checker")

Embedded in mozjemalloc; closely related to Chrome's GWP-ASan

Sep'20

Android

Implemented in LLVM's Scudo allocator, turned on for system processes and system apps

2020

Meta (Facebook/Messenger)

Deployed in Meta's apps.

Apr'21

Apple Products ("Probabilistic Guard Malloc")

First deployed with iOS 14.5 and macOS 11.3

and **Linux Kernel ("KFENCE")**

Upstream since Linux 5.12 (April 2021), radically different implementation (vs. user space variants) which has to work within an OS kernel

Results

Results: Google Server Software (TCMalloc)

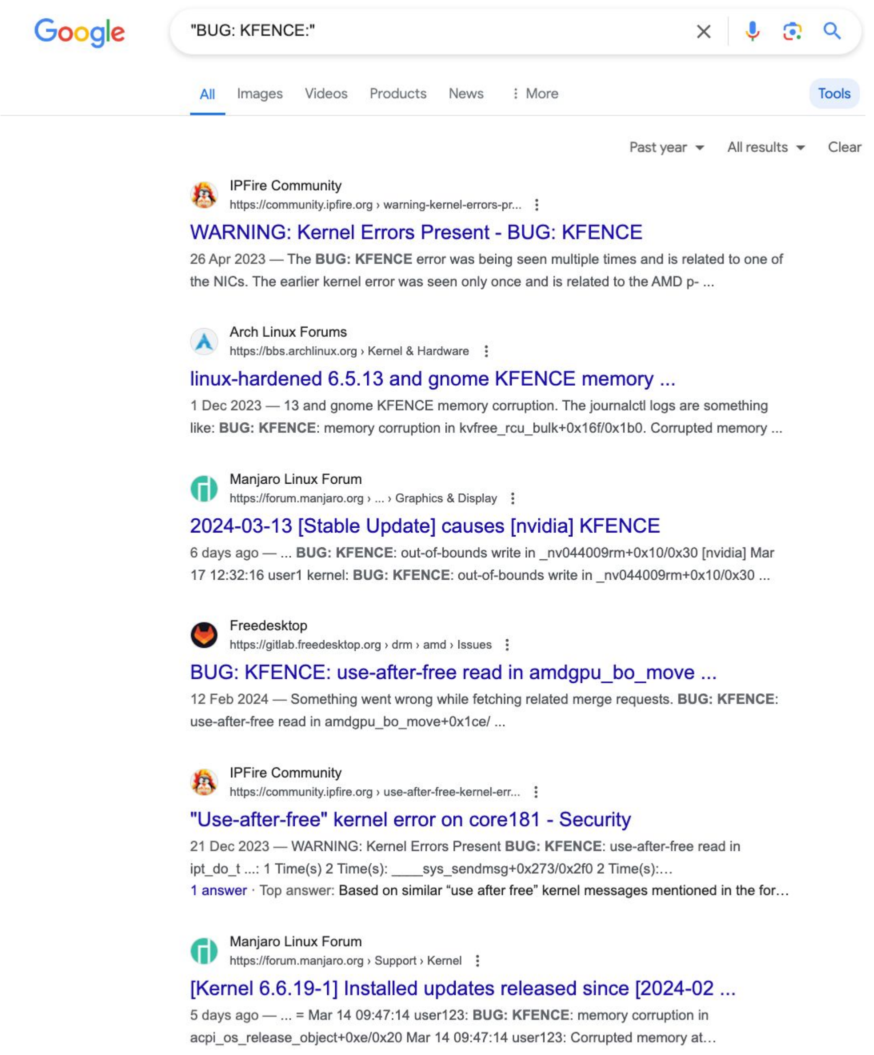
- Since 2019, more than 2300 bugs have been fixed due to GWP-ASan reports
- 80% heap use-after-free, 20% buffer overflow
- Several cases where GWP-ASan detected root cause of an ongoing issue
- Monitoring + benchmarks confirm no significant performance impact

Results: Google Chrome

- 271 bugs filed, with 65% to be possibly exploitable
- 243 bugs resolved, with 69% fix rate
- 80.4% of bugs filed were found and reported by GWP-ASan before any other crash bug was filed for the same crash

Results: KFENCE (Linux kernel)

- KFENCE has reported 60+ bugs in Google's downstream Linux kernels
- Upstream kernel up to version 6.3 has 12 fix commits mentioning KFENCE
- Enabled in various common Linux distributions and Linux CI systems



"BUG: KFENCE:"

IPFire Community
https://community.ipfire.org › warning-kernel-errors-pr... ⋮
WARNING: Kernel Errors Present - BUG: KFENCE
26 Apr 2023 — The **BUG: KFENCE** error was being seen multiple times and is related to one of the NICs. The earlier kernel error was seen only once and is related to the AMD p- ...

Arch Linux Forums
https://bbs.archlinux.org › Kernel & Hardware ⋮
linux-hardened 6.5.13 and gnome KFENCE memory ...
1 Dec 2023 — 13 and gnome KFENCE memory corruption. The journalctl logs are something like: **BUG: KFENCE**: memory corruption in kvfree_rcu_bulk+0x16f/0x1b0. Corrupted memory ...

Manjaro Linux Forum
https://forum.manjaro.org › ... › Graphics & Display ⋮
2024-03-13 [Stable Update] causes [nvidia] KFENCE
6 days ago — ... **BUG: KFENCE**: out-of-bounds write in _nv044009rm+0x10/0x30 [nvidia] Mar 17 12:32:16 user1 kernel: **BUG: KFENCE**: out-of-bounds write in _nv044009rm+0x10/0x30 ...

Freedesktop
https://gitlab.freedesktop.org › drm › amd › Issues ⋮
BUG: KFENCE: use-after-free read in amdgpu_bo_move ...
12 Feb 2024 — Something went wrong while fetching related merge requests. **BUG: KFENCE**: use-after-free read in amdgpu_bo_move+0x1ce/ ...

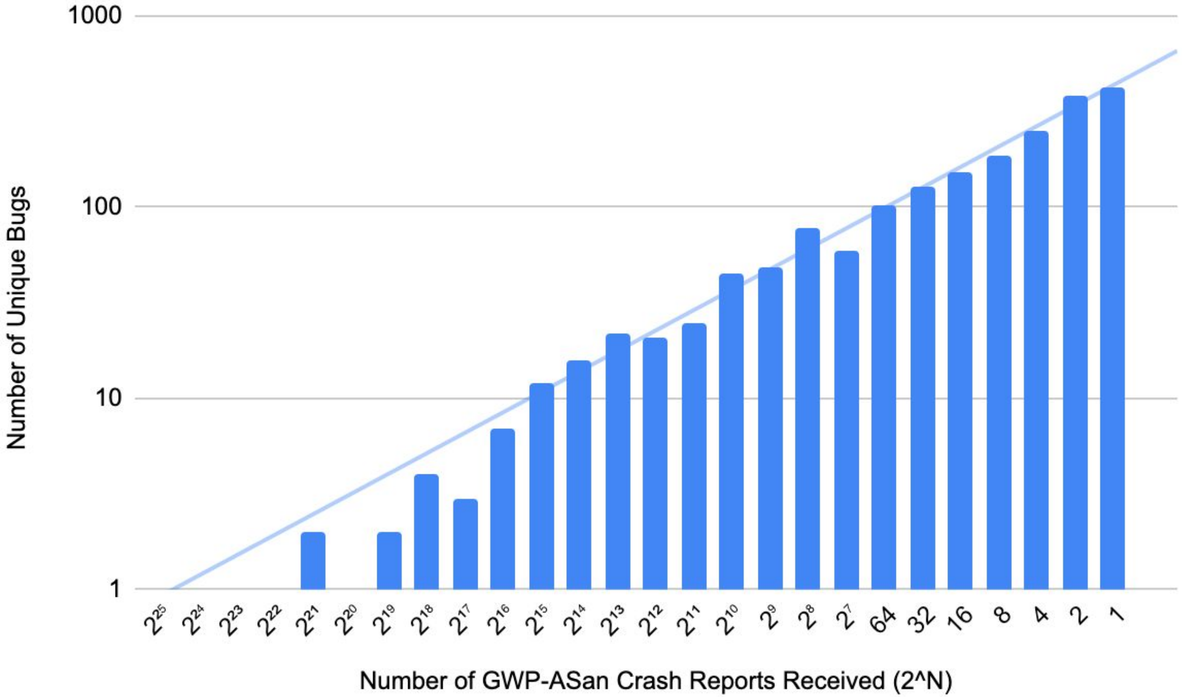
IPFire Community
https://community.ipfire.org › use-after-free-kernel-err... ⋮
"Use-after-free" kernel error on core181 - Security
21 Dec 2023 — WARNING: Kernel Errors Present **BUG: KFENCE**: use-after-free read in ipt_do_t ...: 1 Time(s) 2 Time(s): ____sys_sendmsg+0x273/0x2f0 2 Time(s):...
1 answer · Top answer: Based on similar "use after free" kernel messages mentioned in the for...

Manjaro Linux Forum
https://forum.manjaro.org › Support › Kernel ⋮
[Kernel 6.6.19-1] Installed updates released since [2024-02 ...
5 days ago — ... = Mar 14 09:47:14 user123: **BUG: KFENCE**: memory corruption in acpi_os_release_object+0xe/0x20 Mar 14 09:47:14 user123: Corrupted memory at...

Results: Android

- At time of writing the paper, ~2,000 bugs.
    - We rolled out GWP-ASan for apps in Android 14, non-crashing.
    - Now, a lot more!
- 2-3x more use-after-free than buffer-overflow
- Interesting learnings:
    - Lots of app crashes caused by memory corruption from non-app driver code (GPU, etc.)..
    - ... more bugs are yet to be found

Results: Android - more bugs are yet to be found!



Results: Apple – Probabilistic Guard Malloc (March 2024)

- Since 2021, more than 1,600 bugs have been fixed due to PGM reports
- 76% heap use-after-free, 24% buffer overflow
- About a third of fixed bugs diagnosed to be concurrency issues
- High **99% fix rate** compares very favorably with standard memory crashers
- Several cases where a single PGM report made the difference for diagnosing an ongoing, high-impact bug

On average, ***2.3 new bugs*** *have been found* ***every day***
since PGM was first deployed at scale in April 2021.

Summary

- Memory safety remains a major unresolved problem: eventually migrate away from memory-unsafe code, but this will take decades
- ***GWP-ASan offers a low overhead option for bug detection in production***
- Produces actionable reports
- Not a replacement for ASan or other testing tools
- Results from 6 major variants of GWP-ASan, which are deployed across real-world applications with billions of users

GWP-ASan is not a security mitigation mechanism; when used, however, it improves overall product security by allowing developers to detect and fix many vulnerabilities.