GWP-TSan

Zero-Cost Detection of Data Races in Production

Matt Morehouse, Kostya Serebryan

October 2020
What is GWP-TSan?
What is **GWP-TSan**?

- **GWP-TSan Will Provide Thread Sanitization**

- Probabilistic data race detector (heap only).
  - Still under development.

- Built on top of [GWP-ASan](#).
Background: GWP-ASan

- Tiny fraction of allocations (e.g. 1/100,000) routed to GWP-ASan.
  - Sampling rate adjusted for negligible CPU overhead.

```
return GuardedAlloc();
```

```
// Normal malloc
...
return alloc;
```
## Background: GWP-ASan

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

![Diagram of memory allocation and protection](chart.png)
Background: DataCollider

```c
int racy_counter = 0;

int get_racy_counter() {
    return racy_counter;
}

void inc_racy_counter() {
    ++racy_counter;
}
```
Background: DataCollider

1. Set **breakpoint** on a random memory access.

```c
int racy_counter = 0;

int get_racy_counter() {
    return racy_counter;
}

void inc_racy_counter() {
    ++racy_counter;
}
```
Background: DataCollider

```c
int racy_counter = 0;

int get_racy_counter() {
  return racy_counter;
}

void inc_racy_counter() {
  ++racy_counter;
}
```

1. Set **breakpoint** on a random memory access.
2. When breakpoint fires, remove breakpoint and set a watchpoint on the accessed memory.
Background: DataCollider

```c
int racy_counter = 0;

int get_racy_counter() {
    return racy_counter;
}

void inc_racy_counter() {
    ++racy_counter;
}
```

1. Set breakpoint on a random memory access.
2. When breakpoint fires, remove breakpoint and set a watchpoint on the accessed memory.
Background: DataCollider

```c
int racy_counter = 0;

int get_racy_counter() {
    return racy_counter;
}

void inc_racy_counter() {
    ++racy_counter;
}
```

1. Set breakpoint on a random memory access.
2. When breakpoint fires, remove breakpoint and set a watchpoint on the accessed memory.
3. Wait.
Background: DataCollider

1. Set breakpoint on a random memory access.
2. When breakpoint fires, remove breakpoint and set a watchpoint on the accessed memory.
3. Wait.
4. If watchpoint fires while waiting, report a data race.
**Background: DataCollider**

```c
int racy_counter = 0;

int get_racy_counter() {
    return racy_counter;
}

void inc_racy_counter() {
    ++racy_counter;
}
```

1. Set breakpoint on a random memory access.
2. When breakpoint fires, remove breakpoint and set a watchpoint on the accessed memory.
3. Wait.
4. If watchpoint fires while waiting, report a data race.

**DATA RACE!**
Background: DataCollider

```c
int racy_counter = 0;

int get_racy_counter() {
    return racy_counter;
}

void inc_racy_counter() {
    ++racy_counter;
}
```

1. Set breakpoint on a random memory access.
2. When breakpoint fires, remove breakpoint and set a watchpoint on the accessed memory.
3. Wait.
4. If watchpoint fires while waiting, report a data race.
5. Otherwise, remove watchpoint and continue execution.
GWP-TSan = GWP-ASan + DataCollider

- Periodically set watchpoints on GWP-ASan allocations.

- Report a data race when concurrent accesses to the same address are detected, with:
  - At least one write.
  - At least one non-atomic access.
Watchpoints

- DataCollider uses debug registers.
  - + Trap on accesses to specific address only.
  - - Only 4 debug registers.

- GWP-TSan uses `mprotect(PROT_NONE)` and SEGV handler.
  - - Trap on any access within the same 4KB page.
  - + Unlimited watchpoints.
  - + Potential use of Intel pkeys for speed.
Challenges
Atomic Accesses

Problem:

- Concurrent atomic accesses should not be reported as races.
- How to tell if an access is atomic?

Solution:

- LLVM backend pass to create a PC table of atomic access instructions.
- Read the table into memory during GWP-TSan initialization.
- $O(1)$ `isAtomic()` check for any PC.
System Calls

Problem:

● Passing PROT_NONE memory to syscalls makes them fail with EFAULT.

Solution:

● Intercept glibc syscall wrappers.
● Remove watchpoints before syscalls.
Thank you!
(Feedback is welcome)