AddressSanitizer for Windows

Timur Iskhodzhanov
Google

AddressSanitizer (a.k.a. ASan)

- High performance
  - Uses compile-time instrumentation
  - Lightweight algorithm
  - Multi-threaded
- Focuses on severe bugs
  - buffer overflows
  - uses of freed / unavailable memory
  - and more
- Supports Linux, Mac OS; more in development

ASan overview follows

A more complete version:

Konstantin Serebryany, Derek Bruening, Alexander Potapenko, Dmitry Vyukov, ***AddressSanitizer: a fast address sanity checker****, Proceedings of the 2012 USENIX conference on Annual Technical Conference, 2012*

ASan code instrumentation

Original code:

```
*addr = 42;
```

Instrumented pseudocode:

```
if (!is_ok_to_use(addr))
  print_report_and_crash();
// memory is ok to use:
*addr = 42;
```

ASan shadow memory

A state of every aligned 8 bytes of memory is stored in a single *shadow* byte

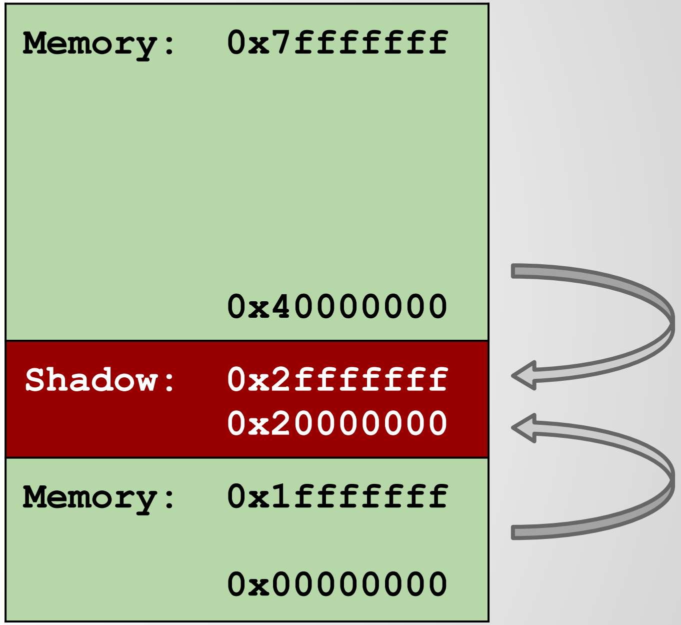
Simple shadow address calculation

```
shadow_addr = addr / 8 + offset
```

Allows very simple instrumentation,
performed at LLVM IR level

ASan shadow memory

- Easy to allocate memory for the shadow
- Fixed address range
- Have to do it early



Function interception

Have to intercept some functions:

- **`malloc`**, **`free`**, etc. – to track memory
- **`strlen`**, **`memcpy`**, etc. – to detect more errors
- **`pthread_create`**, etc. – to understand the app

Error reporting

- Grab the current stack trace
- Pinpoint the (mis)accessed memory allocation
- Get extra info from allocation metadata
- Print out everything
- Terminate the process

ASan for Windows – overview

- Goal: find nasty Chromium bugs on Windows
- Started in 2012 after ASan success on Linux
- “Beta” experience available mid-2014

Progress overview

- Instrumentation – no changes needed, thanks IR!
- Significant changes to the ASan run-time library (RTL)
- Massive effort on Clang C++ ABI support
- clang-cl bonus: can mix MSVC & Clang **`.obj`** files, supports automatic fallback (e.g. code with exceptions)

C run-time support

- Multiple C run-time (CRT) implementations:
  - `/MT` (static linkage)
  - `/MTd` (static linkage, debug)
  - `/MD` (DLL linkage)
  - `/MDd` (DLL linkage, debug)
- Each CRT requires different handling
- Currently supported: `/MT`, `/MD`
- Each DLL might have its own copy of `/MT` CRT, i.e. `malloc`, heap, CRT global state etc.

/MT CRT support

EXE

- Just define **`malloc`**, etc. to intercept them
- **`dllimport`**’ed functions like **`CreateThread`** need to be hot-patched at start-up
- Init ASan RTL as part of the first **`calloc`** early in CRT init

DLL

- Redirect calls to intercepted functions from DLL to the interceptor implementations in the EXE

/MD CRT support

- Also need to hot-patch **`MSVCR*.dll`** early
- RTL is a DLL without dependencies to CRT, gets initialized earlier

Report symbolization and debug info

ASan requires line tables to be useful.

Added COFF line table debug info support to LLVM

- Almost-free bonus: can step line by line in debuggers (VS, windbg)
- Can’t look up variable values though

Deployment

- Can build and run Chromium
- Deployed to ClusterFuzz,
  found 50+ security bugs in 3 months
- We’re working with Mozilla Firefox and other OSS developers

38
DEMO TIME ....
memegenerator.net

Thanks for listening!

Please try AddressSanitizer on your Windows app

p.s. tests and patches are welcome

Timur Iskhodzhanov

timurrrr@google.com