Sanitize your C++

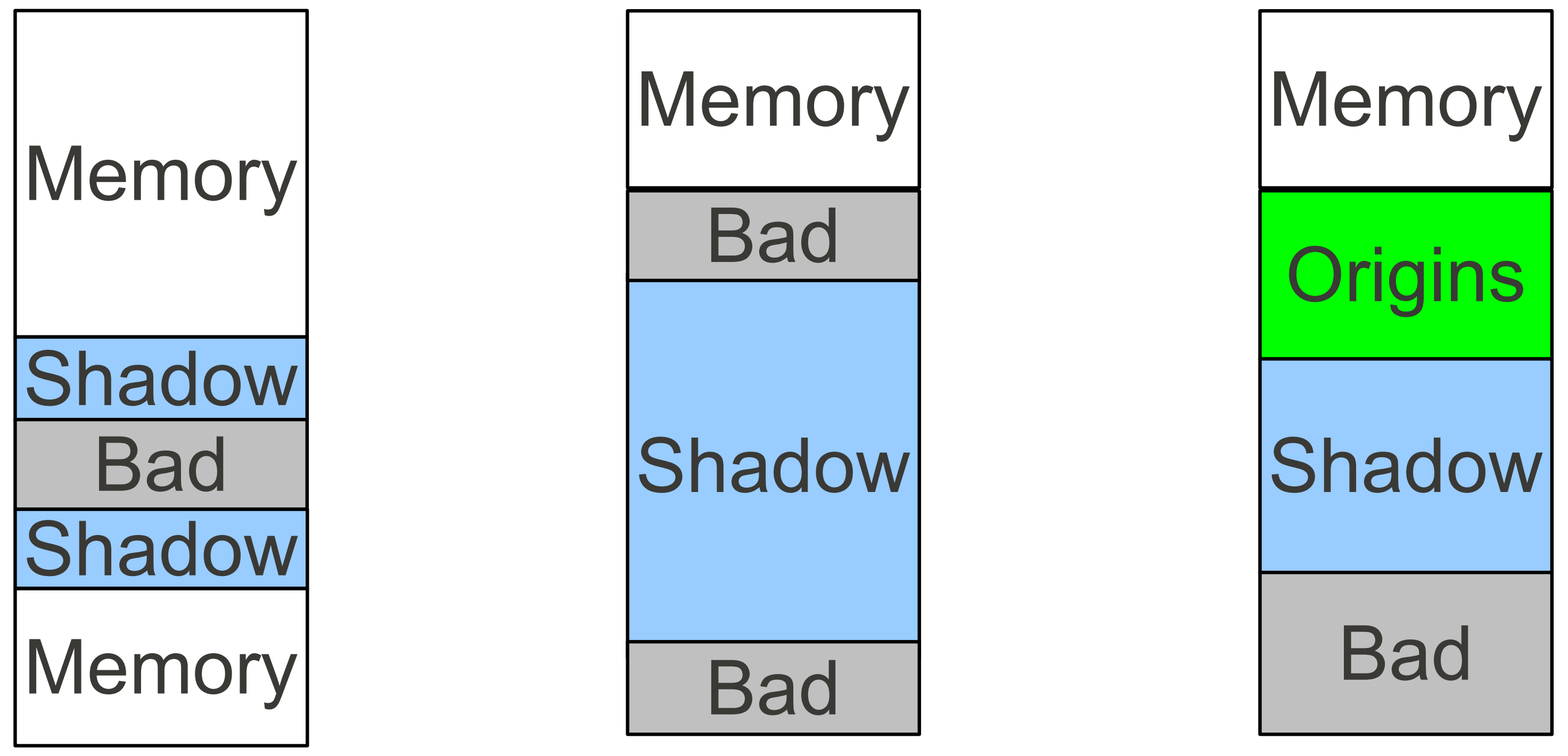
AddressSanitizer (ASan):
- Buffer overflow in heap, stack and globals
- Heap-use-after-free
- Stack-use-after-return (sometimes)

ThreadSanitizer (TSan):
- Data races

MemorySanitizer (MSan):
- Uses of uninitialized memory

Direct mapped Shadow Memory

ASan: 8:1   TSan: 1:4   MSan: 1:2



Low Overhead

```
ASan: CPU: 2x;     RAM: 2x-4x
TSan: CPU: 4x-8x;  RAM: 5x
MSan: CPU: 3x;     RAM: 3x
```

Valgrind, Dr.Memory
- 20x/10x slowdown
- Neither one handles stack/globals

Helgrind, Intel Inspector
- 30x+ slowdown
- No support for atomics

Found bugs everywhere:

WebKit FFmpeg Firefox Chrome SPEC 2006 FreeType Vim LLVM GCC WebRTC MySQL Perl

Compiler Instrumentation
- Insert extra instructions/calls before every memory access
- Shadow propagation for arithmetic instructions (MSan)
- Add redzones for stack/globals (ASan)

Run-time library
- Replace malloc, initialize shadow
- Insert redzones for heap (Asan)
- Race detection logic (TSan)

Status:
ASan: LLVM 3.1
TSan: LLVM trunk
MSan: under review

Challenge #1:
Port to Windows!

Challenge #2:
Instrument libraries!

Challenge #3:
Implement in hardware!

ASan Example

```
int main(int argc, char **argv) {
  int *array = new int[100];
  delete [] array;
  return array[argc];  // BOOM
}
% clang++ -g -O1 -faddress-sanitizer example.cc
% ./a.out
ERROR: AddressSanitizer heap-use-after-free
READ of size 4 at 0x7faa07fce084 thread T0
    #0 0x40433c in main example.cc:4
0x7faa07fce084 is located 4 bytes inside of
400-byte region [0x7faa07fce080,0x7faa07fce210)
freed by thread T0 here:
    #0 0x4058fd in operator delete[](void*)
    #1 0x404303 in main example.cc:3
previously allocated by thread T0 here:
    #0 0x405579 in operator new[](unsigned long)
    #1 0x4042f3 in main example.cc:2
```

TSan Example

```
void Thread1() { Global = 42; }
int main() {
  pthread_create(&t, 0, Thread1, 0);
  Global = 43;
  ...

% clang -fthread-sanitizer -g a.c -fPIE -pie
% ./a.out
WARNING: ThreadSanitizer: data race
  Write of size 4 at … by thread 1:
    #0 Thread1 a.c:1
  Previous write of size 4 at … by main thread:
    #0 main a.c:4
  Thread 1 (tid=20374, running) created at:
    #1 main a.c:3
```

MSan Example

```
int main(int argc, char **argv) {
  int x[10];
  x[0] = 1;
  if (x[argc]) return 1;
  ...
% clang -fmemory-sanitizer -fPIE -pie  a.c -g
% ./a.out
WARNING: MemorySanitizer: UMR
    #0 0x7ff6b05d9ca7 in main a.c:4
  ORIGIN: stack allocation: x@main
```