Case Study

Purging Undefined Behavior and Intel Assumptions in a Legacy Codebase

ROTH MICHAELS

Cppcon
The C++ Conference

2022
September 12th-16th

Roth Michaels

Principal Software Engineer,
Soundwide Audio Research

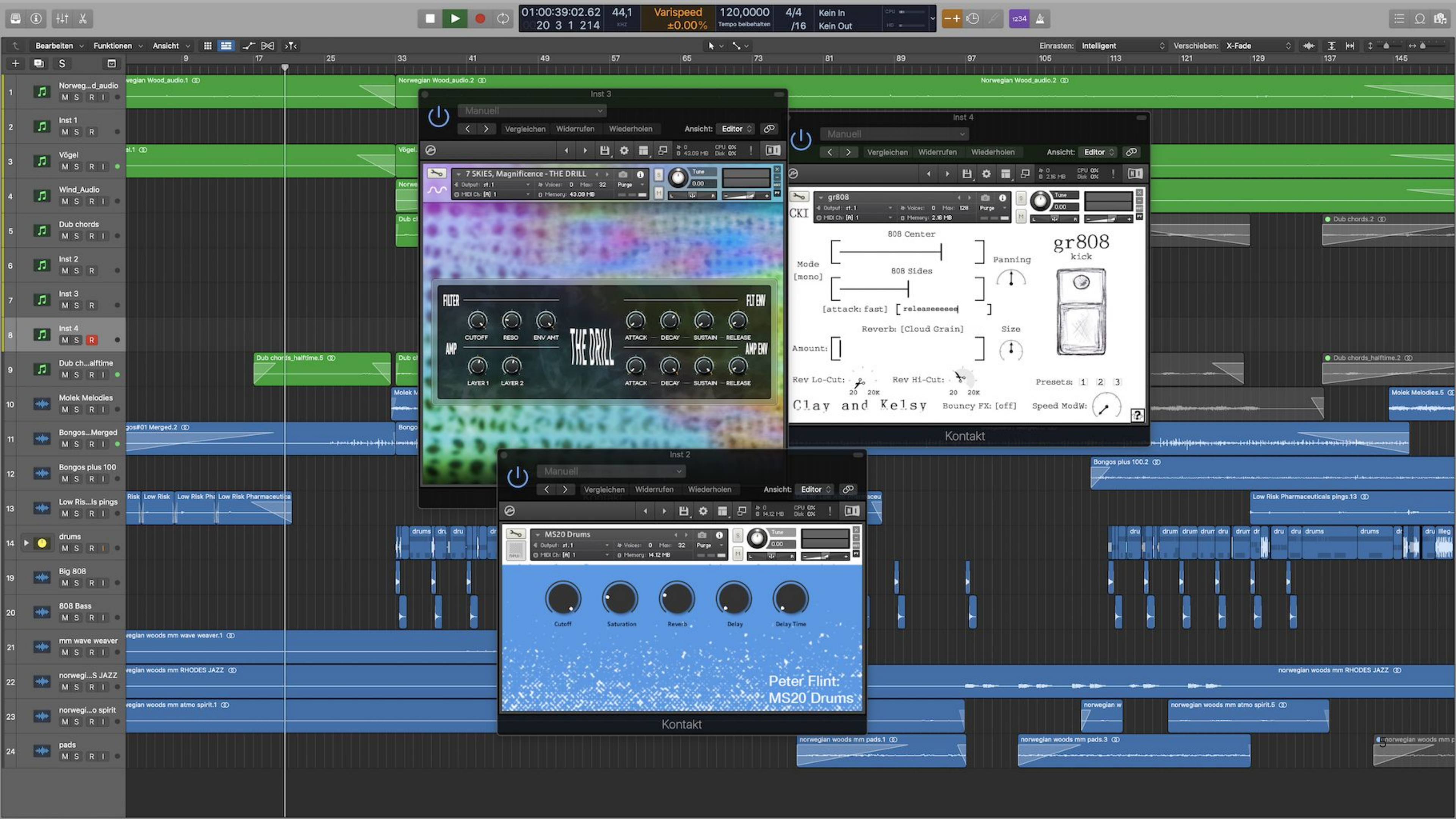
Thank you: Russell McCelllan

Principal Software Engineer,
Audio Production Architect

soundwide

soundwide

iZOTOPE NATIVE INSTRUMENTS Plugin Alliance BRAINWORX Sound Stacks

01:00:39:02.62
20 3 1 214
44,1
Varispeed ±0.00%
120,0000
Tempo beibehalten
4/4
/16
Kein In
Kein Out
Bearbeiten
Funktionen
Ansicht
Einrasten: Intelligent
Verschieben: X-Fade
Norweg...d_audio
Inst 1
Vögel
Wind_Audio
Dub chords
Inst 2
Inst 3
Inst 4
Dub ch...alftime
Molek Melodies
Bongos...Merged
Bongos plus 100
Low Ris...ls pings
drums
Big 808
808 Bass
mm wave weaver
norwegi...S JAZZ
norwegi...o spirit
pads
Norwegian Wood_audio.1
Norwegian Wood_audio.2
Dub chords_halftime.5
Low Risk Pharmaceuticals pings.13
norwegian woods mm RHODES JAZZ
norwegian woods mm atmo spirit.5
norwegian woods mm pads.1
norwegian woods mm pads.3
Inst 3
Manuell
Vergleichen
Widerrufen
Wiederholen
Ansicht: Editor
7 SKIES, Magnificence - THE DRILL
FILTER
CUTOFF
RESO
ENV AMT
AMP
LAYER 1
LAYER 2
THE DRILL
FLT ENV
AMP ENV
ATTACK
DECAY
SUSTAIN
RELEASE
Inst 4
gr808
808 Center
Mode [mono]
808 Sides
Panning
[attack: fast]
[releaseeeeee]
Reverb: [Cloud Grain]
Size
Amount:
Rev Lo-Cut:
Rev Hi-Cut:
20 20K
Presets: 1 2 3
gr808 kick
Clay and Kelsy
Bouncy FX: [off]
Speed ModW:
Kontakt
Inst 2
MS20 Drums
Cutoff
Saturation
Reverb
Delay
Delay Time
Peter Flint: MS20 Drums
Kontakt

```
unsigned random() {
  unsigned x
  return x;
}
```

Fast, High-Quality Pseudo-Random Numbers for Non-Cryptographers

ROTH MICHAELS

Cppcon
The C++ Conference

2022 | September 12th-16th

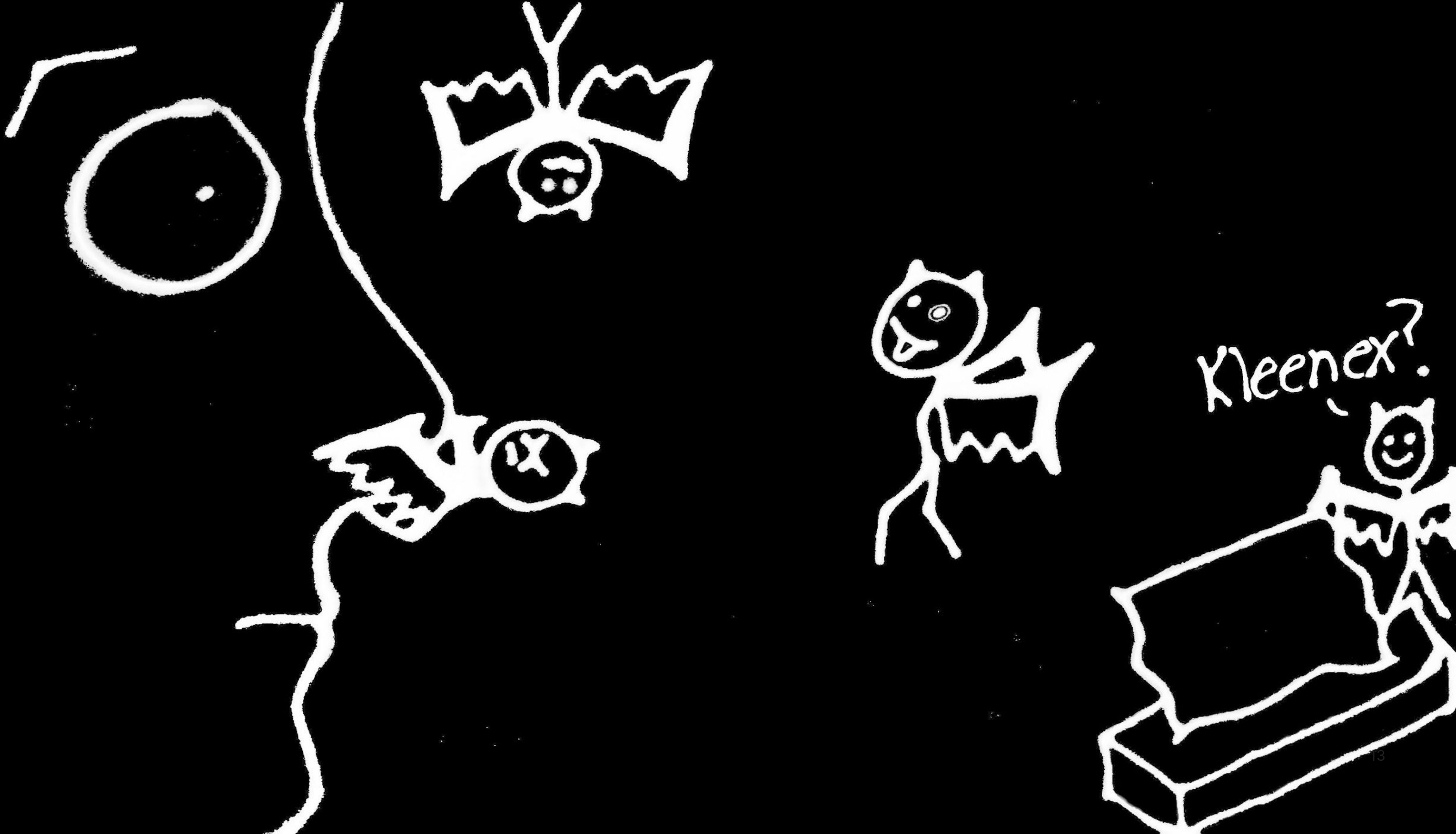
```
unsigned random() {
    unsigned x
    return x;
}
```

```
unsigned random() {
  unsigned x;
  return x;
}
```

> “It’s getting better all the time…”
>
> Lennon-McCartney

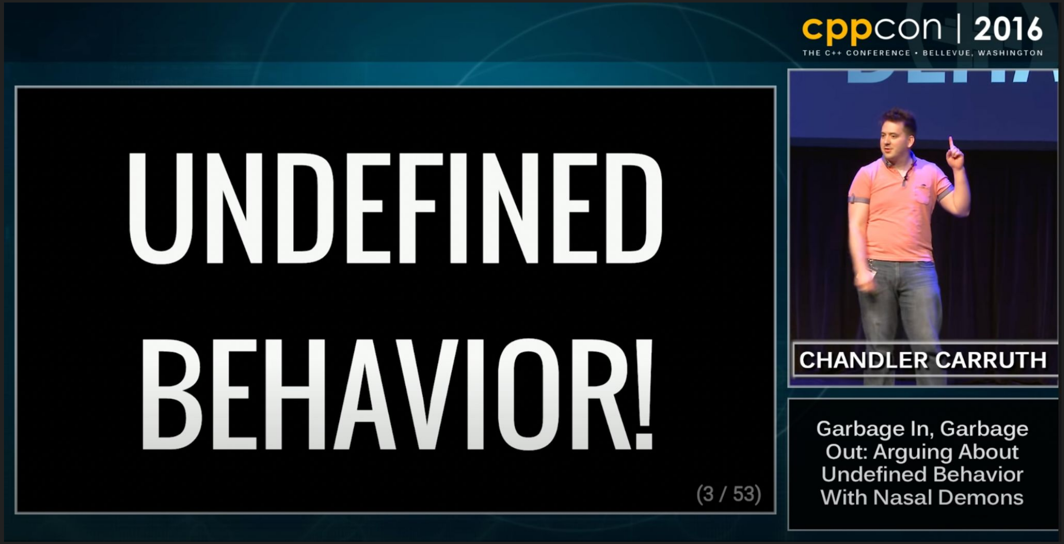
```
unsigned random() {
  unsigned x;
  return x;
}
```

0xABADBABE
0xFEE1DEAD



Kleenex?

Garbage In, Garbage Out: Arguing about UB...



UNDEFINED BEHAVIOR!

(3 / 53)

Garbage In, Garbage Out: Arguing About Undefined Behavior With Nasal Demons

https://www.youtube.com/watch?v=yG1OZ69H_-o

```
unsigned random() {
  unsigned x;
  return x;
}
```

Undefined Behavior (-O0)

```
random:
        push    rbp
        mov     rbp, rsp
        mov     eax, dword ptr [rbp - 4]
        pop     rbp
        ret
```

Undefined Behavior

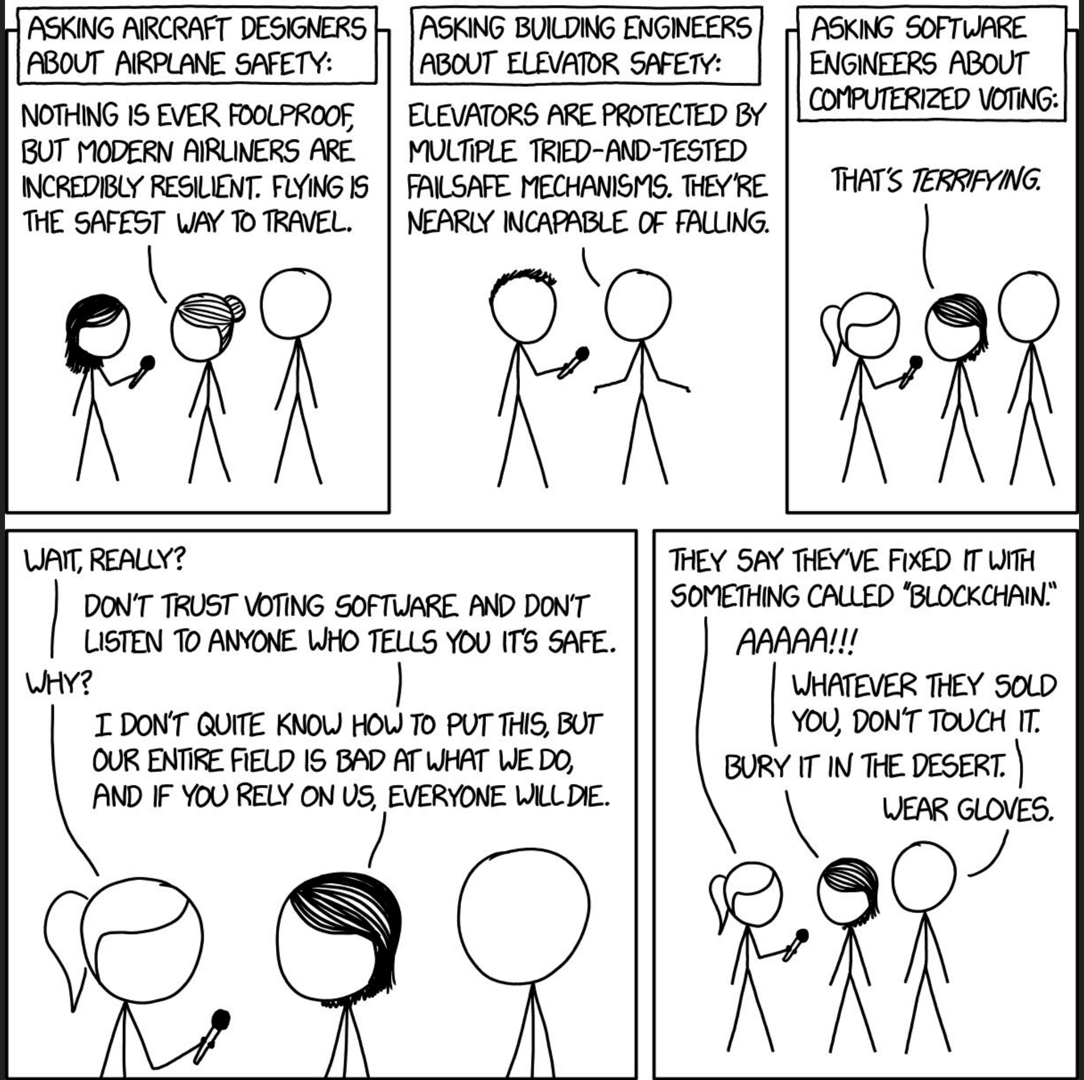
```
random:
        push    rbp
        mov     rbp, rsp
        mov     eax, dword ptr [rbp - 4]
        pop     rbp
        ret
```

Undefined Behavior (-O3)

```
random:
        ret
```

Who has written UB on purpose?

I’ve done it and checked my codegen…


ASKING AIRCRAFT DESIGNERS ABOUT AIRPLANE SAFETY:
NOTHING IS EVER FOOLPROOF, BUT MODERN AIRLINERS ARE INCREDIBLY RESILIENT. FLYING IS THE SAFEST WAY TO TRAVEL.
ASKING BUILDING ENGINEERS ABOUT ELEVATOR SAFETY:
ELEVATORS ARE PROTECTED BY MULTIPLE TRIED-AND-TESTED FAILSAFE MECHANISMS. THEY'RE NEARLY INCAPABLE OF FALLING.
ASKING SOFTWARE ENGINEERS ABOUT COMPUTERIZED VOTING:
THAT'S *TERRIFYING.*
WAIT, REALLY?
DON'T TRUST VOTING SOFTWARE AND DON'T LISTEN TO ANYONE WHO TELLS YOU IT'S SAFE.
WHY?
I DON'T QUITE KNOW HOW TO PUT THIS, BUT OUR ENTIRE FIELD IS BAD AT WHAT WE DO, AND IF YOU RELY ON US, EVERYONE WILL DIE.
THEY SAY THEY'VE FIXED IT WITH SOMETHING CALLED "BLOCKCHAIN."
AAAAA!!!
WHATEVER THEY SOLD YOU, DON'T TOUCH IT.
BURY IT IN THE DESERT.
WEAR GLOVES.

xkcd.com/2030/

> “...those guys are not serious programmers.”
>
> - drdriller

“C++: Enough rope to shoot yourself in the foot”

- unknown

> “Software engineering is programming over time.”

- Titus Winters

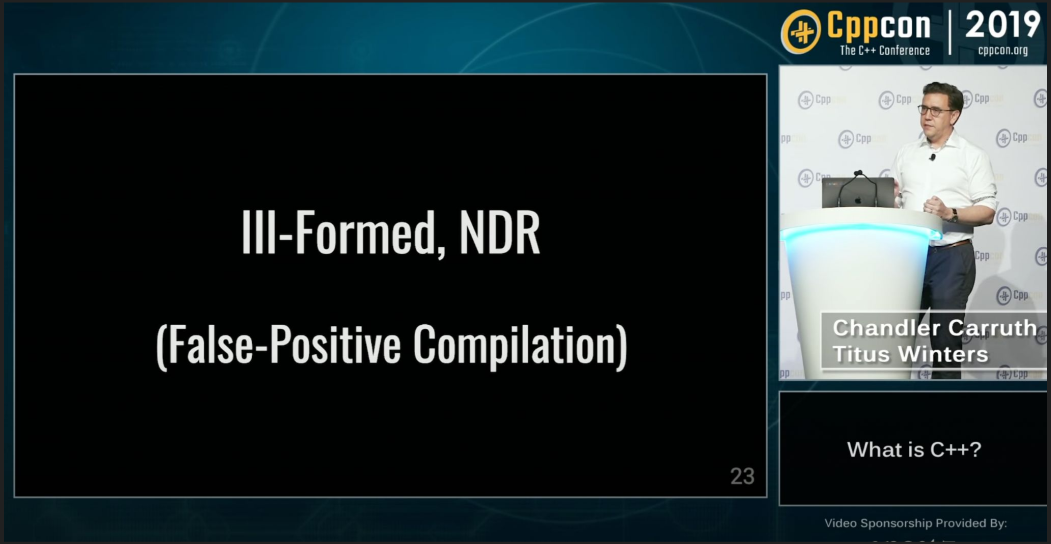
Agenda

- Undefined behavior (UB) overview
- Interesting bugs caused by invoking UB
- Culture changes to fight UB
- Tooling changes to fight UB

Standard C++ Programs

- Ill-formed
  - Compile error
- Ill-formed No Diagnostic Required
  - No error; not a C++ program
- Defined Behavior
- Implementation-Defined Behavior
- Unspecified Behavior
- Undefined Behavior

What is C++



Ill-Formed, NDR

(False-Positive Compilation)

https://www.youtube.com/watch?v=LJh5QCV4wDg

Behavior of the C++ Abstract Machine

- Defined Behavior
    - Deterministic behavior specified by the standard
- Implementation-Defined Behavior
    - Platforms can determine behavior, must document
- Unspecified Behavior
    - Non-deterministic behavior; suggested by standard, documentation not required
- Undefined Behavior
    - Standard makes no guarantees, compilers can assume this never happens, anything is allowed

Back to Basics: Undefined Behavior

https://www.youtube.com/watch?v=NpL9YnxnOqM

How my attitude on UB changed

- *What Every C Programmer Should Know About Undefined Behavior*
  - Chris Lattner
  - http://blog.llvm.org/2011/05/what-every-c-programmer-should-know.html

- *A Guide to Undefined Behavior in C and C++*
  - John Regehr
  - https://blog.regehr.org/archives/213

John Regehr: Behavior of Functions

- Type 1:
  - Behavior is defined for all inputs
- Type 2:
  - Defined for some inputs and undefined for others
- Type 3:
  - Behavior is undefined for all inputs

Type 1 Function

```
int32_t safe_div (int32_t a, int32_t b) {
  if ((b == 0) ||
      ((a == INT32_MIN) && (b == -1))) {
    report_integer_math_error();
    return 0;
  } else {
    return a / b;
  }
}
```

Type 1 Function

```
int32_t safe_div (int32_t a, int32_t b) {
  if ((b == 0) ||
      ((a == INT32_MIN) && (b == -1))) {
    report_integer_math_error();
    return 0;
  } else {
    return a / b;
  }
}
```

Type 1 Function

```
int32_t safe_div (int32_t a, int32_t b) {
  if ((b == 0) ||
      ((a == INT32_MIN) && (b == -1))) {
    report_integer_math_error();
    return 0;
  } else {
    return a / b;
  }
}
```

```
std::array<int, 4> table[4];
bool exists_in_table(int v) {
  for (int i = 0; i <= table.size(); ++i) {
    if (table[i] == v) {
      return true;
    }
  }
  return false;
}
```

Type 2 Function

```c
int32_t checked_div (int32_t a, int32_t b) {
  assert(b == 0);
  assert((a == INT32_MIN) && (b == -1));
  return a / b;
}
```

Type 3 Functions

```
unsigned random() {
   unsigned x;
   return x;
}
```

More undefined behavior optimization examples

```
std::array<int, 4> table[4];
bool exists_in_table(int v) {
  for (int i = 0; i <= table.size(); ++i) {
    if (table[i] == v) {
      return true;
    }
  }
  return false;
}
```

```
std::array<int, 4> table[4];
bool exists_in_table(int v) {
  return true;
}
```

Disappearing null-checks

```
int value_or_answer(int* p) {

  return p ? *p : 42;
}
```

Disappearing null-checks

```
int value_or_answer(int* p) {
  std::print(“p: {}”, *p);
  return p ? *p : 42;
}
```

Disappearing null-checks

```
int value_or_answer(int* p) {
  std::print(“p: {}”, *p);
  return p ? *p : 42;
}
```

Disappearing null-checks

```
int value_or_answer(int* p) {
  std::print(“p: {}”, *p);
  return *p
}
```

Time travel can result in time travel

- Ramond Chen
- https://devblogs.microsoft.com/oldnewthing/20140627-00/?p=633

Undefined behavior is awesome!

cppcon | 2017
THE C++ CONFERENCE • BELLEVUE, WASHINGTON
PIOTR PADLEWSKI
Undefined Behavior is awesome!

https://www.youtube.com/watch?v=ehyHyAIa5so

It works on Intel...

```
typedef volatile unsigned atomic_uint;

unsigned AtomicRead(const atomic_uint&);

void AtomicWrite(atomic_uint&, unsigned);
```

atomic Weapons: The C++ Memory Model and Modern Hardware

https://herbsutter.com/2013/02/11/atomic-weapons-the-c-memory-model-and-modern-hardware/

Windows Implementation

```
void AtomicWrite(atomic_uint& val,
                 unsigned valNew) {

(void)InterlockedExchange(
   reinterpret_cast<volatile LONG*>(&val),
   static_cast<LONG>(valNew));
}
```

Windows Implementation

```
unsigned AtomicRead(const atomic_uint& x) {
     // Note that reads of 32-bit values
     // are guaranteed to be atomic, even on
     // multiprocessor systems, according to
     // MSDN's "Interlocked Variable Access"
     // doc
     return x;
}
```

macOS Implementation

```
void AtomicWrite(atomic_uint& val,
                 unsigned valNew) {
   // [explaining stuff]...
   val = valNew;
}
unsigned AtomicRead(const atomic_uint& x) {
   // [explaining stuff]...
   return x;
}
```

How did this work?

Don’t worry about it

atomic Weapons: The C++ Memory Model and Modern Hardware

https://herbsutter.com/2013/02/11/atomic-weapons-the-c-memory-model-and-modern-hardware/

New Song
Soundcheck
Volume
volca drum
DIGITAL PERCUSSION SYNTHESIZER
Song idea 1
Edit

UB in legacy code can feel hopeless

How big is the universe?

~15 Millions Lines of C/C++/Objective-C(++)

~670,000 Lines

Product Code

- Ozone
- RX
- Neutron
- Nectar
- etc.

~1.33 Million Lines

Shared Code

- iZBase
- iZDSPBase
- Glass
- EqualizerIIR
- etc.

~13 Million Lines

Open Source

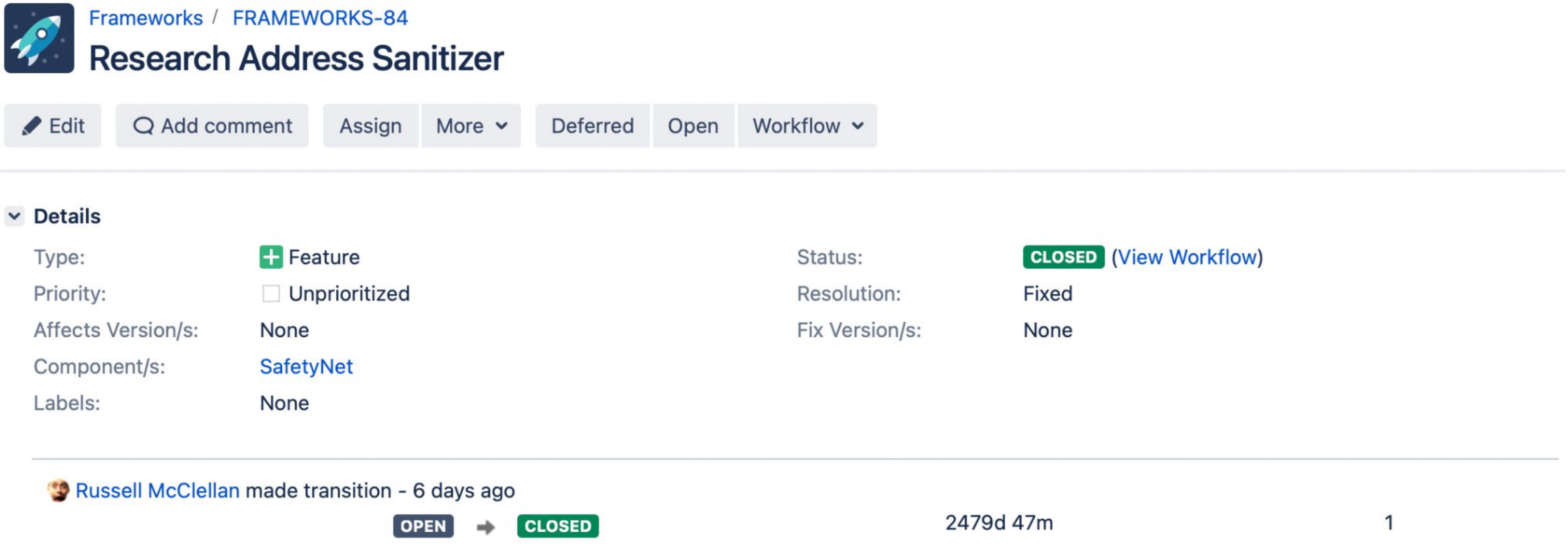
- Boost
- Skia
- libPNG
- libXML2
- etc.

UB in legacy code can feel hopeless

- Too much to fix
- We’ve been doing it for years, why fix it
- It “works”, why fix it
- UB is an intellectual topic for language nerds

Where we started

- Enabled all warnings
- Warnings as errors
- Dreaming of static analysis in CI
- Aware of clang static analyzers
    - Non-trivial to get running



Research Address Sanitizer

Details

Type:	🟩 Feature	Status:	CLOSED (View Workflow)
Priority:	□ Unprioritized	Resolution:	Fixed
Affects Version/s:	None	Fix Version/s:	None
Component/s:	SafetyNet		
Labels:	None		

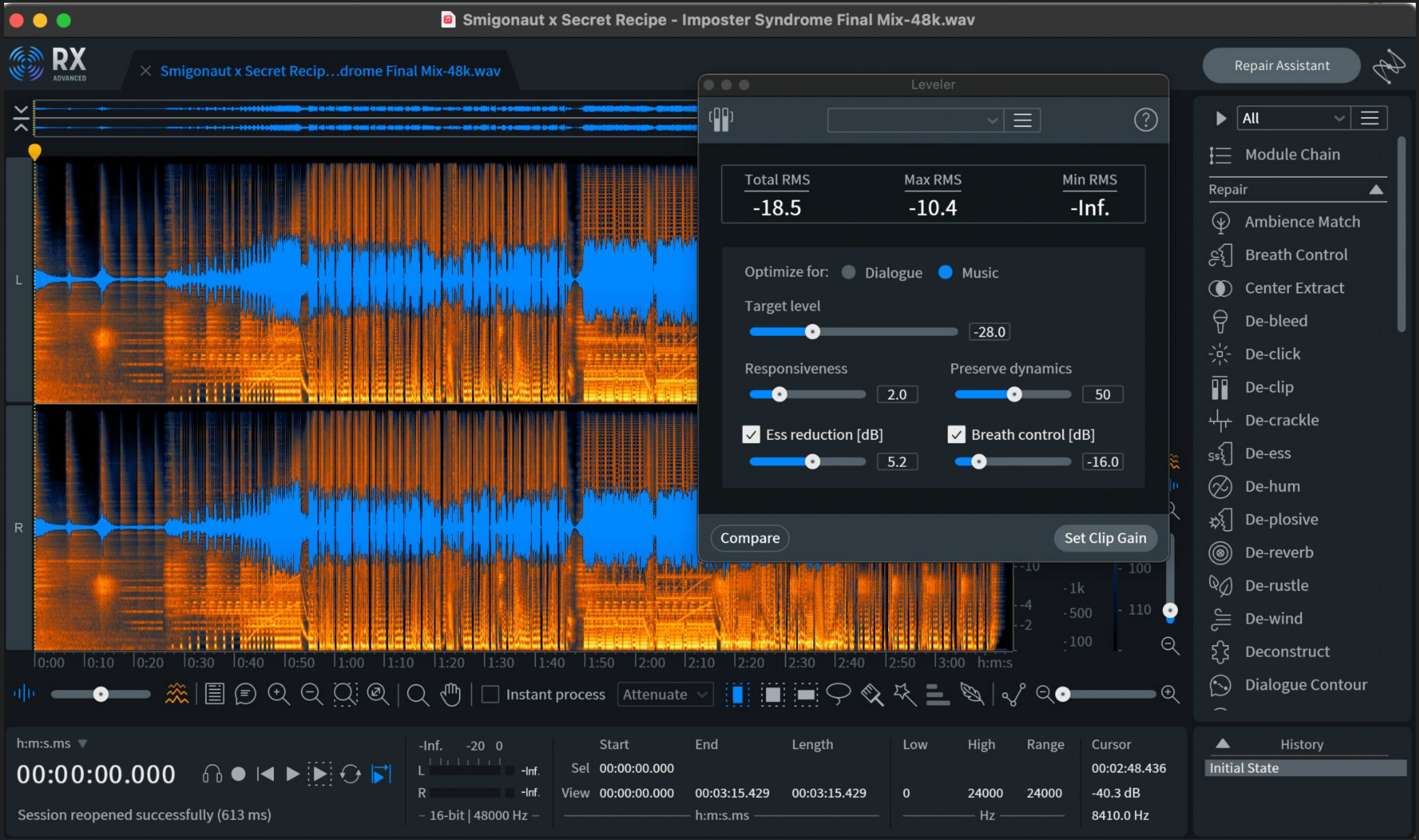
Russell McClellan made transition - 6 days ago

OPEN → CLOSED 2479d 47m 1

War stories fighting undefined behavior

Attack of the Xcode update

Strange stuff happening when a UX designer got an automatic Xcode update



Smigonaut x Secret Recipe - Imposter Syndrome Final Mix-48k.wav

Leveler

Total RMS	Max RMS	Min RMS
-18.5	-10.4	-Inf.

Optimize for: ○ Dialogue ● Music

Target level -28.0

Responsiveness 2.0

Preserve dynamics 50

☑ Ess reduction [dB] 5.2

☑ Breath control [dB] -16.0

Compare

Set Clip Gain

00:00:00.000

Session reopened successfully (613 ms)

16-bit | 48000 Hz

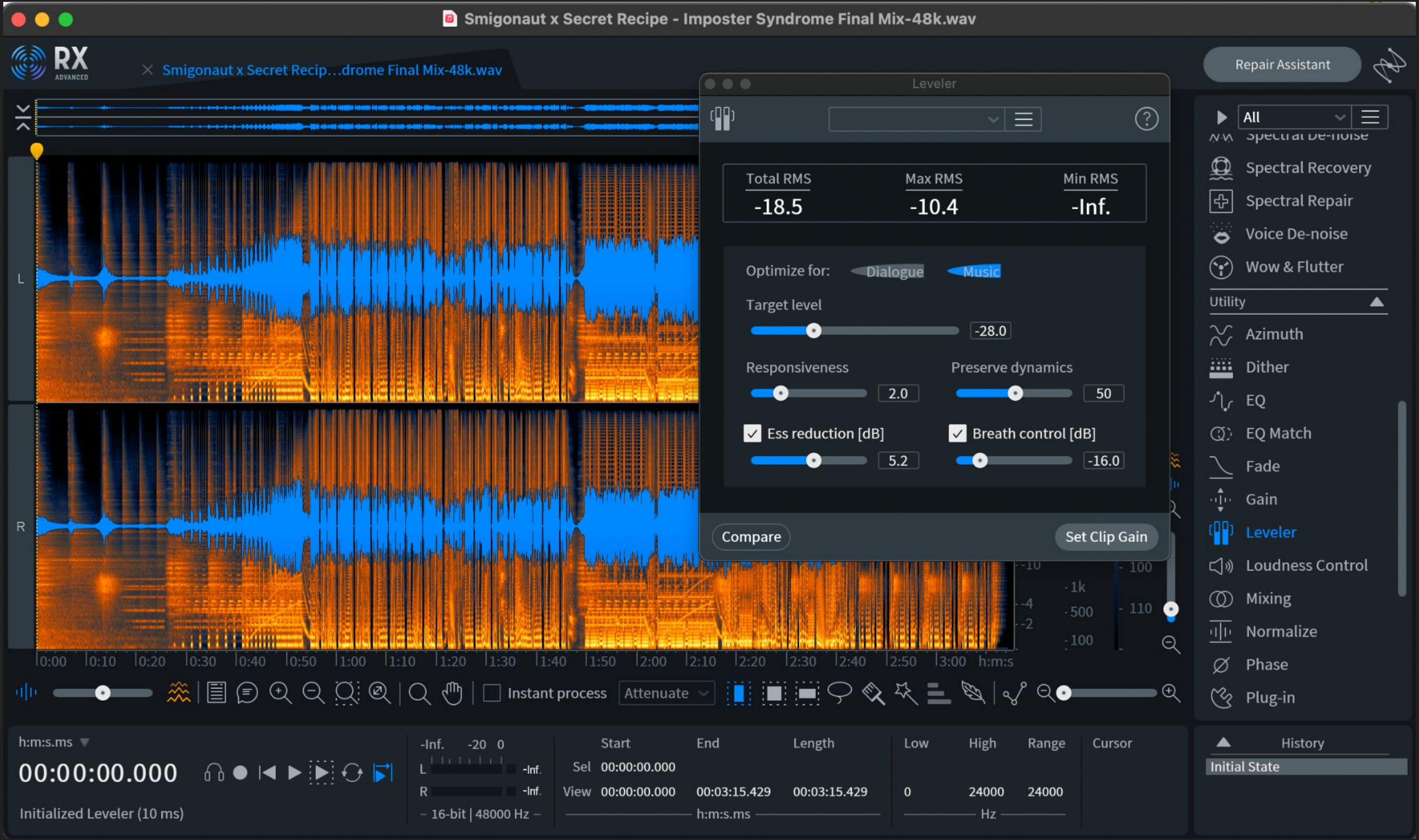
	Start	End	Length
Sel	00:00:00.000		
View	00:00:00.000	00:03:15.429	00:03:15.429

Low	High	Range
0	24000	24000

Cursor 00:02:48.436 -40.3 dB 8410.0 Hz

History

Initial State



Total RMS	Max RMS	Min RMS
-18.5	-10.4	-Inf.

Optimize for: ○ Dialogue ● Music

Target level -28.0

Responsiveness 2.0

Preserve dynamics 50

☑ Ess reduction [dB] 5.2

☑ Breath control [dB] -16.0

h:m:s.ms

00:00:00.000

Initialized Leveler (10 ms)

-Inf. -20 0

L -Inf.

R -Inf.

16-bit | 48000 Hz

	Start	End	Length
Sel	00:00:00.000		
View	00:00:00.000	00:03:15.429	00:03:15.429

h:m:s.ms

Low	High	Range	Cursor
0	24000	24000	

Hz

Maybe it is a compiler bug?

1. Ask for help on IRC
2. Try C-reduce
3. All code disappears
4. You be the compiler!

You didn’t find a compiler bug

Well, we did once:

https://developercommunity.visualstudio.com/t/Aligned-instructions-generated-incorrect/1640338

```
void DrawCircleOutline(float x, float y,
                       float radius) {
  auto e = ellipse{x, y, radius, radius};
  auto s = stroke<ellipse>{e};
  auto p = transform<decltype(s)>{
            s, getTransform()
  };
  m_rasterizer->addPath(p);
}
```

```
void DrawCircleOutline(float x, float y,
                       float radius) {
  auto e = ellipse{x, y, radius, radius};
  auto s = stroke<ellipse>{e};
  auto p = transform<decltype(s)>{
            s, getTransform()
  };
  m_rasterizer->addPath(p);
}
```

```
template <class V, class T>
class transform {
public:
  transform(V& source, const T& tr)
            : m_source(&source)
            , m_transform(&tr) {}
private:
  V* m_source;
  const T* m_transform;
    };
}
```

```
template <class V, class T>
class transform {
public:
  transform(V& source, const T& tr)
            : m_source(&source)
            , m_transform(&tr) {}
private:
  V* m_source;
  const T* m_transform;
    };
}
```

```
void DrawCircleOutline(float x, float y,
                       float radius) {
  auto e = ellipse{x, y, radius, radius};
  auto s = stroke<ellipse>{e};
  auto p = transform<decltype(s)>{
             s, getTransform()
  };
  m_rasterizer->addPath(p);
}
```

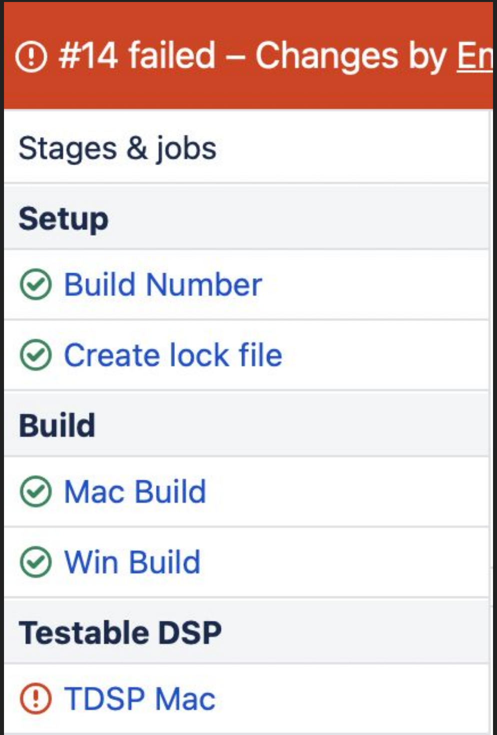
```
void DrawCircleOutline(float x, float y,
                       float radius) {
  auto e = ellipse{x, y, radius, radius};
  auto s = stroke<ellipse>{e};
  auto t = getTransform();
  auto p = transform<decltype(s)>{
            s, t
  };
  m_rasterizer->addPath(p);
}
```

Fast forward in time...

Crashing with the latest Windows update

A sleeping dragon awakes the week before a major version update release

Fear red builds



ⓘ #14 failed – Changes by En

Stages & jobs

Setup

✓ Build Number

✓ Create lock file

Build

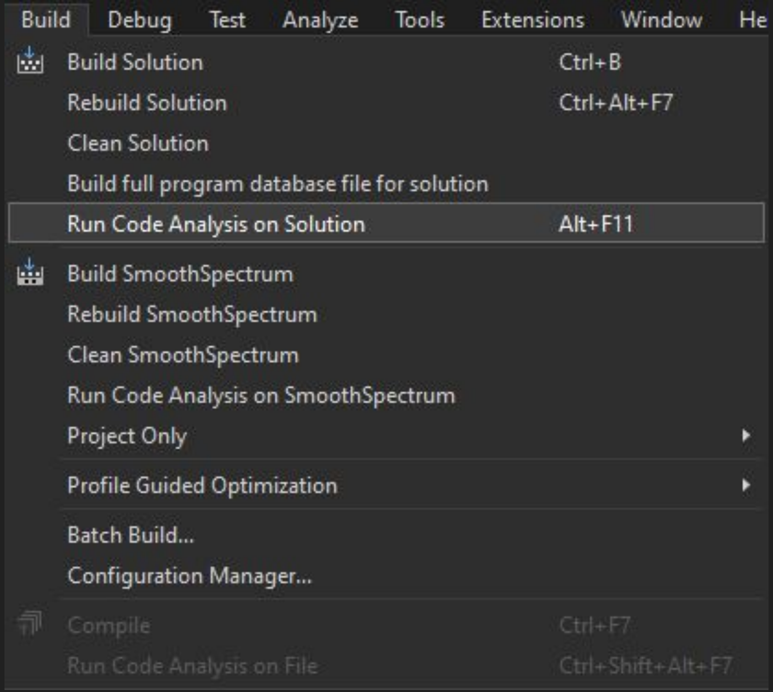
✓ Mac Build

✓ Win Build

Testable DSP

ⓘ TDSP Mac

Visual Studio Static Analysis


Build
Debug
Test
Analyze
Tools
Extensions
Window
He
Build Solution Ctrl+B
Rebuild Solution Ctrl+Alt+F7
Clean Solution
Build full program database file for solution
Run Code Analysis on Solution Alt+F11
Build SmoothSpectrum
Rebuild SmoothSpectrum
Clean SmoothSpectrum
Run Code Analysis on SmoothSpectrum
Project Only
Profile Guided Optimization
Batch Build...
Configuration Manager...
Compile Ctrl+F7
Run Code Analysis on File Ctrl+Shift+Alt+F7

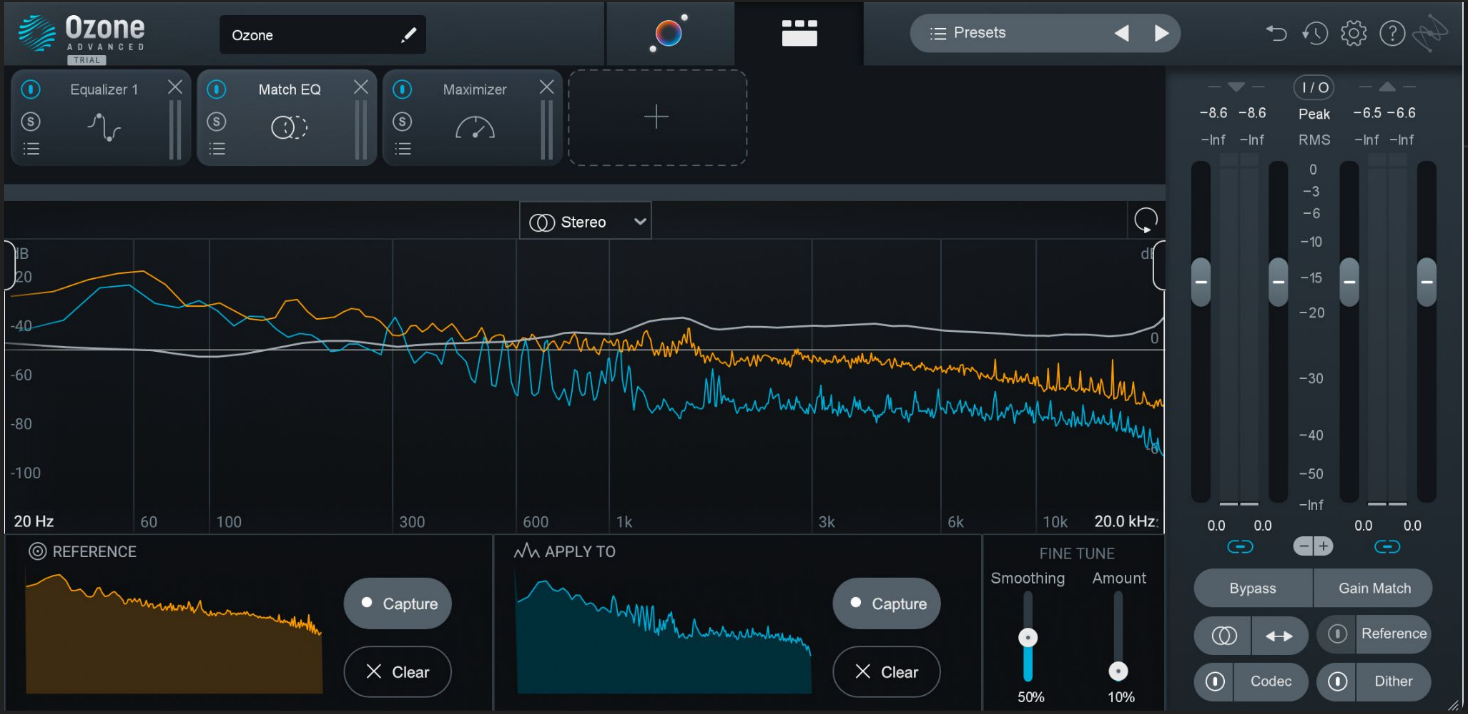
What it found:

```
struct SnapshotData {
    std::vector<float> freqs{20.f, 20000.f};
    std::vector<Float> dBAmps{2.f, -200.f};
    uint32_t snapshotColor;
    unsigned opacity{191};
    bool visible{false};
    bool enabled{false};
};
```

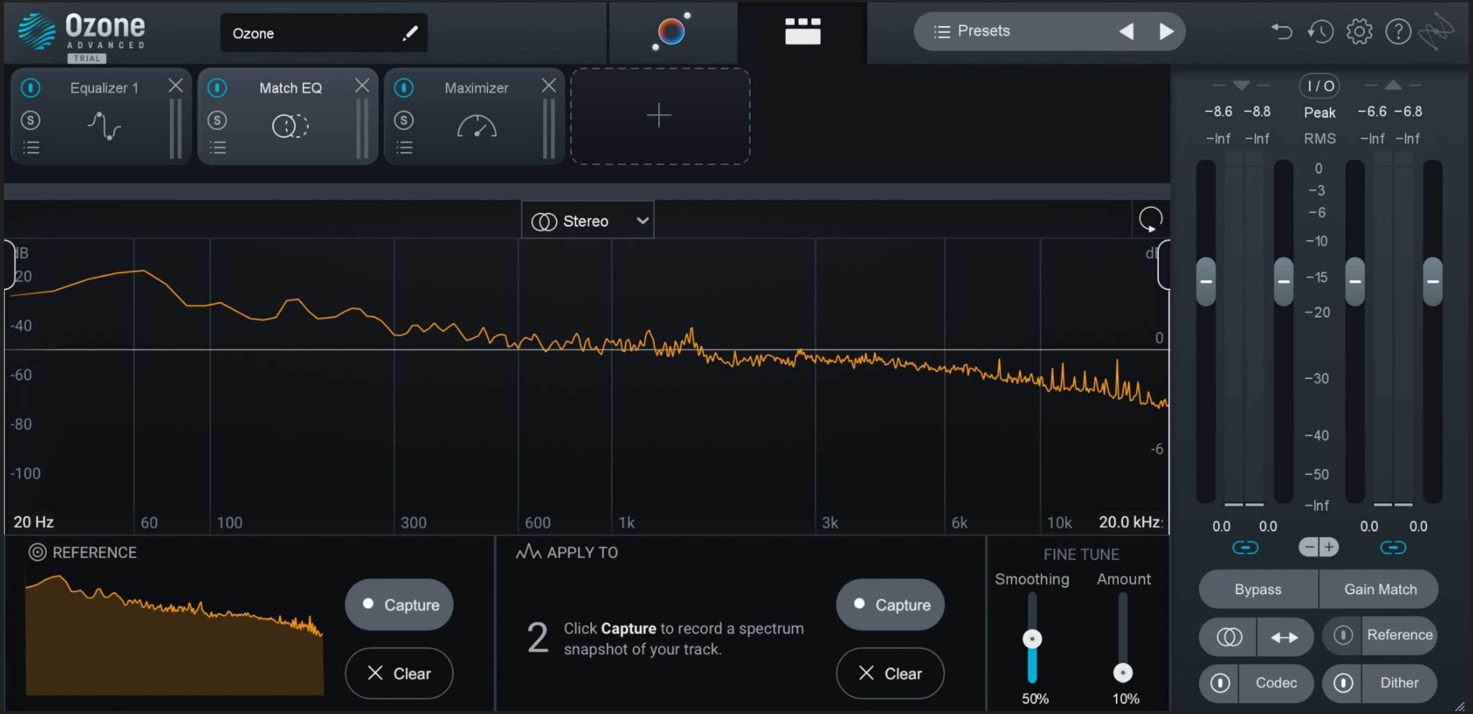
What it found:

```
struct SnapshotData {
    std::vector<float> freqs{20.f, 20000.f};
    std::vector<Float> dBAmps{2.f, -200.f};
    uint32_t snapshotColor;
    unsigned opacity{191};
    bool visible{false};
    bool enabled{false};
};
```

Ozone Match EQ



Ozone Match EQ



2 Click **Capture** to record a spectrum snapshot of your track.

What it found:

```
struct SnapshotData {
    std::vector<float> freqs{20.f, 20000.f};
    std::vector<Float> dBAmps{2.f, -200.f};
    uint32_t snapshotColor;
    unsigned opacity{191};
    bool visible{false};
    bool enabled{false};
};
```

```
std::vector<std::byte> compress(std::byte*, std::size_t size);


std::vector<std::byte> saveState() {
    const auto size = [] {
        auto s = SnapshotData{};
        compress(reinterpret_cast<std::byte*>(&s), sizeof(s));
    }();
    auto state = std::vector<std::byte>(size);
    auto s = SnapshotData{};
    auto data = compress(reinterpret_cast<std::byte*>(&s), sizeof(s));
    std::copy(data.begin(), data.end(), state.begin());
}
```

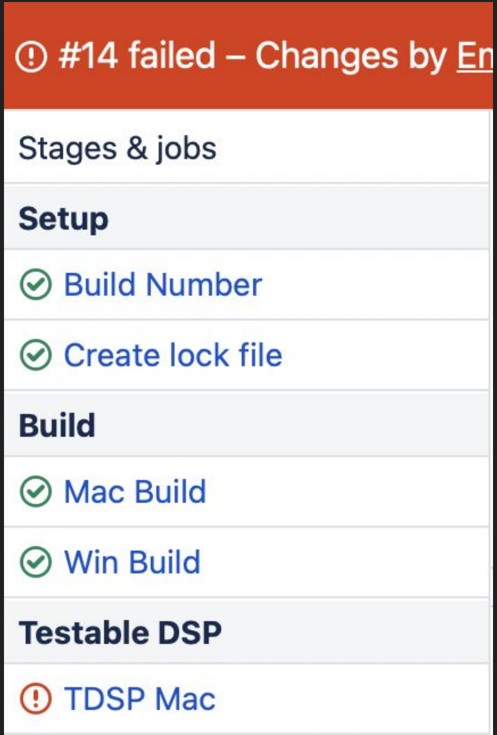
The fix:

```
struct SnapshotData {
    std::vector<float> freqs{20.f, 20000.f};
    std::vector<Float> dBAmps{2.f, -200.f};
    uint32_t snapshotColor{};
    unsigned opacity{191};
    bool visible{false};
    bool enabled{false};
};
```

Non-deterministic regression test failures

When your code breaks
when you didn’t change it

Fear red builds



#14 failed – Changes by En

Stages & jobs

Setup

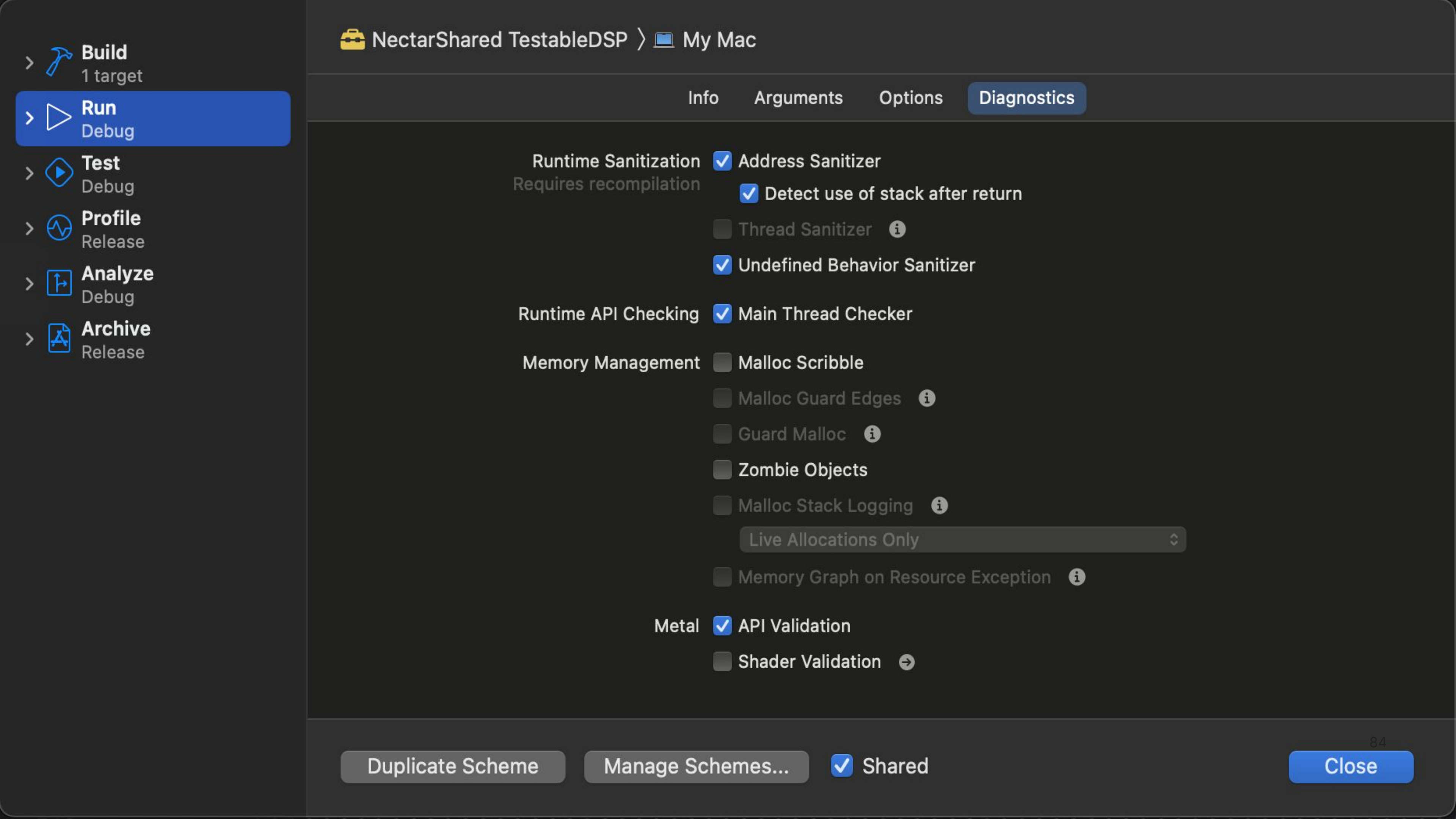
- ✓ Build Number
- ✓ Create lock file

Build

- ✓ Mac Build
- ✓ Win Build

Testable DSP

- ! TDSP Mac



Runtime Sanitization
Requires recompilation
☑ Address Sanitizer
☑ Detect use of stack after return
☐ Thread Sanitizer
☑ Undefined Behavior Sanitizer

Runtime API Checking
☑ Main Thread Checker

Memory Management
☐ Malloc Scribble
☐ Malloc Guard Edges
☐ Guard Malloc
☐ Zombie Objects
☐ Malloc Stack Logging
Live Allocations Only
☐ Memory Graph on Resource Exception

Metal
☑ API Validation
☐ Shader Validation

☑ Shared

```
MemoryPool* DSPProcessor::GetMemoryPool();


template <class T>
class MemoryPoolBuffer {
public:
  MemoryPoolBuffer(MemoryPool*, size_t);
  // api like std::vector<T>
}
```

```
DSPProcessor::Process(span<float> buf) {
  auto p = GetMemoryPool();
  const auto n = buf.size();
  MemPoolBuffer<float> analysisBuf(p, n);
  Analyze(buf, analysisBuf);
  MemPoolBuffer<double> calcBuf(p, n);
  HiResCalc(buf, analysisBuf, calcBuf);
  DoubleToFloat(calcBuf, buf);
}
```

```
DSPProcessor::Process(span<float> buf) {
  // n = 512 [0xCEBF40, 0xCEC740)
  MemPoolBuffer<float> analysisBuf(p, n);
  Analyze(buf, analysisBuf);
  // n = 512 [0xCEBF40, 0xCED740)
  MemPoolBuffer<double> calcBuf(p, n);
  HiResCalc(buf, analysisBuf, calcBuf);
  DoubleToFloat(calcBuf, buf);
}
```

```
DSPProcessor::Process(span<float> buf) {

  MemPoolBuffer<float> analysisBuf(p, n/2+1);
  Analyze(buf, analysisBuf);

  MemPoolBuffer<double> calcBuf(p, n);
  HiResCalc(buf, analysisBuf, calcBuf);
  DoubleToFloat(calcBuf, buf);
}
```

```
DSPProcessor::Process(span<float> buf) {
  // n = 512 [0xCEBF40, 0xCEC344)
  MemPoolBuffer<float> analysisBuf(p, n/2+1);
  Analyze(buf, analysisBuf);
  // n = 512 [0xCEC344, 0xCED344)
  MemPoolBuffer<double> calcBuf(p, n);
  HiResCalc(buf, analysisBuf, calcBuf);
  DoubleToFloat(calcBuf, buf);
}
```

```
DSPProcessor::Process(span<float> buf) {
  // n = 512 [0xCEBF40, 0xCEC344)
  MemPoolBuffer<float> analysisBuf(p, n/2+1);
  Analyze(buf, analysisBuf);
  // n = 512 [0xCEC344, 0xCED344)
  MemPoolBuffer<double> calcBuf(p, n);
  HiResCalc(buf, analysisBuf, calcBuf);
  DoubleToFloat(calcBuf, buf);
}
```

How did it ever work? What went wrong

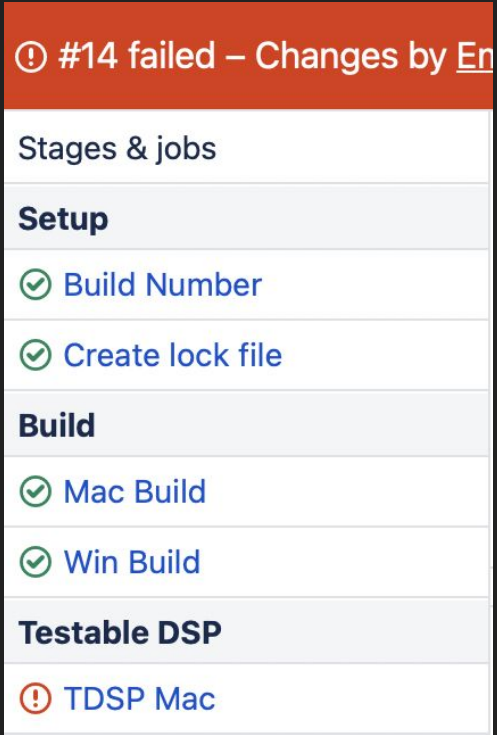
- Intel can do it?
- C++ says you can’t
- Was it Accelerate?
- Not worth understanding UB

How we made cultural and tooling changes

Cultural Changes

- Fearmongering
  - Fear non-deterministic build failures; don’t re-run
  - Re-prioritize bugs based on UB fear
- Education and affection
  - Teaching *everyone* about UB
- We made a rule
  - Don’t write new UB!

Fear red builds



! #14 failed – Changes by En

Stages & jobs

Setup

✓ Build Number

✓ Create lock file

Build

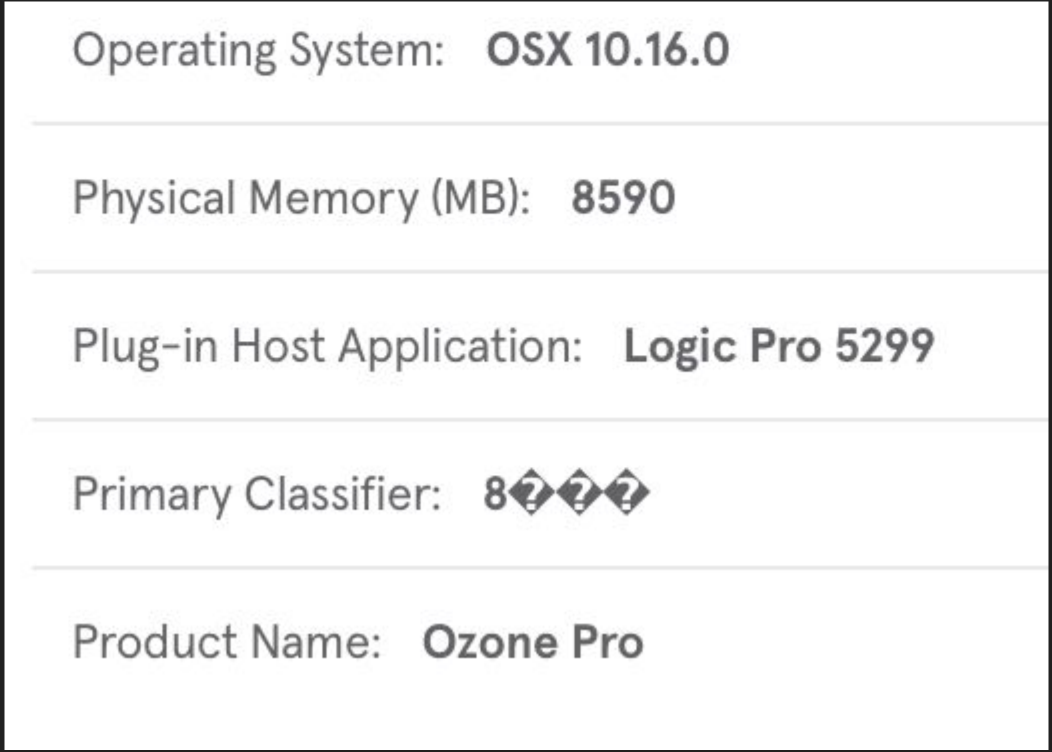
✓ Mac Build

✓ Win Build

Testable DSP

! TDSP Mac

Influencing bug triage



Operating System:	**OSX 10.16.0**
Physical Memory (MB):	**8590**
Plug-in Host Application:	**Logic Pro 5299**
Primary Classifier:	**8���**
Product Name:	**Ozone Pro**

What’s an “ub?”

Teaching developers, QA, and the rest of the company about undefined behavior

Battling Dragons in Ozone

A story of undefined behavior

P.iZ.1: Do not invoke undefined behavior

- New rule in iZotope CppCoreGuidelines fork
- Motivated by PR debate when flagging UB
- Even if we have UB, we don’t have to make it worse
- Reduced cross platform breaks
- More devs are watching out for UB in code review

Tooling Changes

- Deprecation warnings as errors on new code
- Clang-tidy
  - Checks on new code
  - Refactorings
- CI builds with Address Sanitizer and UB Sanitizer
- Manual use of Thread Sanitizer

Deprecations as warnings on new code

```
template <class T>
unsigned RegisterParam(T*, value_range<T>);
```

Deprecations as warnings on new code

```
template <class T>
[[deprecated(“Invokes UB”)]]
unsigned RegisterParam(T*, value_range<T>);

template <class T>
unsigned RegisterParam(std::atomic<T>&,
                       value_range<T>);
```

clang-tidy

- Static analysis (don’t need to run code)
- Many built-in checks (incl. clang static analyzer)
- Not hard to write custom checks
  - Cost: Maintaining clang/llvm fork
- Critical tool for preventing UB
  - Run on every pull request (only changed code)
  - Able to write custom rules
    - Automatic refactoring: e.g. implicit conversion in custom optional type

```
template <class T>
class checked {
  T m_value;
  bool m_valid;
public:
  checked()     : m_value{} , m_valid{false} {}
  checked(T x) : m_value{x}, m_valid{true} {}
  // ...
};
```

```
template <class T>
class checked {
  T m_value;
  bool m_valid;
public:
  checked<T>& operator=(T x) {
    m_value = x;
    m_valid = true;
  }
};
```

```
template <class T>
class checked {
  // ...
  checked<T>& operator=(T x) {
    m_value = x;
    m_valid = true;
  }
};
```

```
template <class T>
class checked {
  // ...
  operator T() { return m_value; }

  T cast() const {
    assert(m_valid);
    return m_value;
  }
};
```

clang-tidy to the rescue!

```
int running_count(checked<int> x) {
   static int count = 0;

   count += x;

   return count;

}
```

clang-tidy to the rescue!

```
int running_count(checked<int> x) {
    static int count = 0;

    count += x.cast();

    return count;

}
```

```cpp
void CheckedValueImplicit::registerMatchers(MatchFinder *Finder) {
  Finder->addMatcher(
      implicitCastExpr(
          allOf(
              unless(isInTemplateInstantiation()),
              has(cxxMemberCallExpr(allOf(
                  anyOf(has(memberExpr(has(ignoringImplicit(
                            declRefExpr().bind("was-simple"))))),
                        has(memberExpr(has(ignoringImplicit(
                            memberExpr().bind("was-member"))))),
                        has(memberExpr(has(
                            ignoringImplicit(callExpr().bind("was-call"))))),
                        anything()),
                  hasDeclaration(cxxConversionDecl(
                      hasParent(classTemplateSpecializationDecl(allOf(
                          hasName("checked_value"), templateArgumentCountIs(1),
                          hasTemplateArgument(
                              0, templateArgument().bind("checked-of")))))))))),
              hasType(qualType().bind("cast-to"))))
          .bind("potential-bad-conversion"),
      this);
}
```

soundwide

```
void CheckedValueImplicit::check(const MatchFinder::MatchResult &Result) {
  const auto *MatchedCast =
      Result.Nodes.getNodeAs<ImplicitCastExpr>("potential-bad-conversion");
  const auto *CheckedOf =
      Result.Nodes.getNodeAs<TemplateArgument>("checked-of");
  const auto *CastTo = Result.Nodes.getNodeAs<QualType>("cast-to");
  if (!MatchedCast || !CheckedOf || !CastTo)
    return;

  if (CastTo->getCanonicalType() != CheckedOf->getAsType().getCanonicalType())
    return;

  auto Diag =
      diag(MatchedCast->getExprLoc(),
           "Implicit conversion operator on `checked_value` is deprecated");
  SourceLocation EndLoc = Lexer::getLocForEndOfToken(
      MatchedCast->getEndLoc(), 0, *Result.SourceManager, getLangOpts());
  if (Result.Nodes.getNodeAs<Expr>("was-simple") ||
      Result.Nodes.getNodeAs<Expr>("was-member") ||
      Result.Nodes.getNodeAs<Expr>("was-call")) {
    Diag << FixItHint::CreateInsertion(EndLoc, ".cast()");
  } else {
    Diag << FixItHint::CreateInsertion(MatchedCast->getBeginLoc(), "(")
         << FixItHint::CreateInsertion(EndLoc, ").cast()");
  }
}
```

Address (ASan) and UB (UBSan) Sanitizers

- Runtime checks for various forms of UB
- Recommendation:
  - Build/run for all projects' standard unit tests
  - Make a plan for rolling it out
    - One step at a time
    - Move up testing pyramid over time
- May need environment variables to run (macOS)

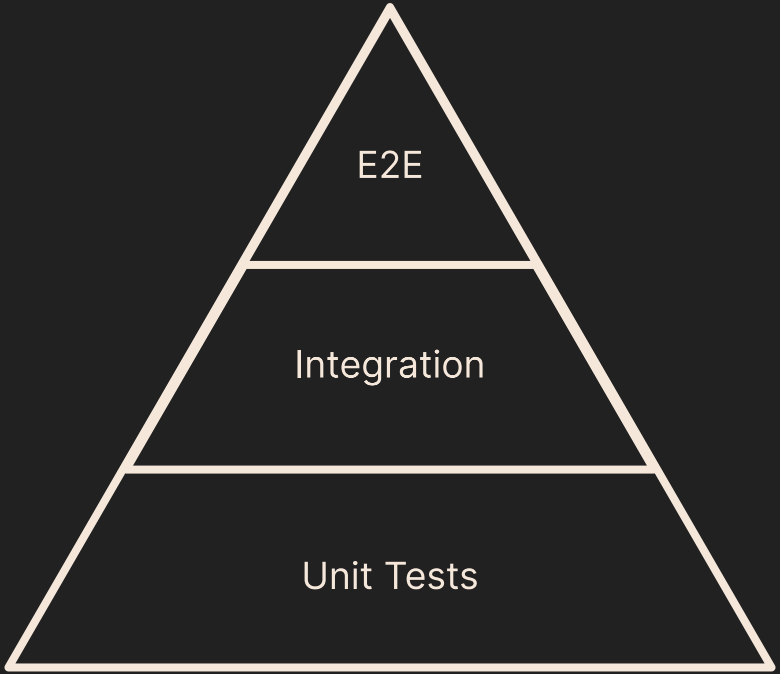
```
void RM::init(const std::byte* rec) {
    while( *rec ) {
        std::string strName(reinterpret_cast<const char*>(rec));
        rec+= strName.size() + 1;
        unsigned offset = ConvertFromLittleEndian(
                               *reinterpret_cast<const unsigned*>(rec));
        unsigned length = ConvertFromLittleEndian(
                               *reinterpret_cast<const unsigned*>(
                                   rec+sizeof(unsigned)));
        rec+= 2*sizeof(unsigned);
        // Add to our table of contents
        m_mapContents[strName]= pair<unsigned,unsigned>( offset, length );
    }
}
```

```
void RM::init(const std::byte* rec) {
    while( *rec ) {
        std::string strName(reinterpret_cast<const char*>(rec));
        rec+= strName.size() + 1;
        unsigned offset;
        memcpy(&offset, rec, sizeof(offset));
        offset = ConvertFromLittleEndian(offset);
        unsigned length;
        memcpy(&length, rec + sizeof(offset), sizeof(length));
        length = ConvertFromLittleEndian(length);
        rec+= 2*sizeof(unsigned);
        // Add to our table of contents
        m_mapContents[strName]= pair<unsigned,unsigned>( offset, length );
    }
}
```

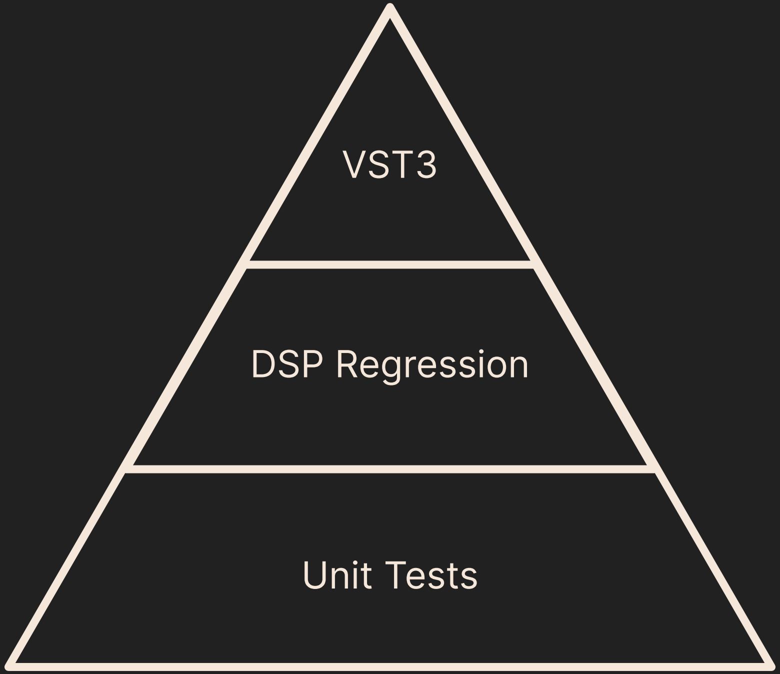
Loading sanitizer libraries

```
DYLD_INSERT_LIBRARIES=/Applications/Xcode.
app/Contents/Developer/Toolchains/XcodeDef
ault.xctoolchain/usr/lib/clang/10.0.0/lib/
darwin/libclang_rt.asan_osx_dynamic.dylib:
/Applications/Xcode.app/Contents/Developer
/Toolchains/XcodeDefault.xctoolchain/usr/l
ib/clang/10.0.0/lib/darwin/libclang_rt.ubs
an_osx_dynamic.dylib
```

Moving up the testing pyramid



Moving up the testing pyramid



U9G269Z5LL 2036
APL1102 339S00833 S
H9HCHNND8MMLSR NEH
VTD1KC 2031 1A
H9HCHNND8MMLSR NEH
VTD1KC 2031 1A

Manual Use of Thread Sanitizer (TSan)

- Another runtime sanitizer (can’t be run with others)
- Most unit test are single threaded
- Fix issues before Apple silicon manual QA
- Found many threading bugs (e.g. `RegisterParam`)
- Wasted dev cycles filing false positives in Jira:
  - `atomic_thread_fence` not supported
  - moodycamel readerwriterqueue
  - Use `AnnotateHappensBefore` with global
  - https://github.com/cameron314/readerwriterqueue/issues/116
  - https://github.com/cameron314/readerwriterqueue/commit/1f3c8e4213115484bcc6d49255b65526ed38cf5b

H9HCHNNND8MMLSR NEH
VTD1KC 2031 1A
H9HCHNNND8MMLSR NEH
VTD1KC 2031 1A
U9G269Z5LL 2036
APL1102 339S00833 S

Extra story: How `const_cast` changed the meaning of silence.

Thank you!

Roth Michaels — Principal Software Engineer
rmichaels@izotope.com
@thevibesman

soundwide