Talos Vulnerability Report

TALOS-2020-1123

Google Chrome DrawElementsInstanced information leak vulnerability

OCTOBER 22, 2020

CVE NUMBER

CVE-2020-6555

SUMMARY

An information disclosure vulnerability exists in the WebGL functionality of Google Chrome 83.0.4103.116 (Stable) (64-bit) and 86.0.4198.0 (Developer Build) (64-bit). Specially crafted JavaScript can cause an out-of-bounds read. The victim must visit a specially crafted, malicious web page to trigger this vulnerability.

CONFIRMED VULNERABLE VERSIONS

The versions below were either tested or verified to be vulnerable by Talos or confirmed to be vulnerable by the vendor.

Google Chrome 83.0.4103.116 (Stable) (64-bit)
Google Chrome 86.0.4198.0 (Developer Build) (64-bit)

PRODUCT URLS

Chrome - https://www.google.com/chrome/

CVSSV3 SCORE

6.8 - CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:L/A:L

CWE

CWE-125 - Out-of-bounds Read

DETAILS

Google Chrome is one of the most popular and feature-rich web browsers and supports the newest technologies and extensions, one of which is WebGL.

This vulnerability is in ANGLE, a compatibility layer between OpenGL and Direct3D used on Windows by Chrome browser and other project.

While executing the supplied PoC, Chromium browser crashes inside ConvertIndexArray from IndexDataManager. Snippet of this function is as follows:

```
 1:  void ConvertIndexArray(const void *input,
 2:                         gl::DrawElementsType sourceType,
 3:                         void *output,
 4:                         gl::DrawElementsType destinationType,
 5:                         GLsizei count,
 6:                         bool usePrimitiveRestartFixedIndex)
 7:  {
 8:      const InputT *in = static_cast<const InputT *>(input);
 9:      DestT *out       = static_cast<DestT *>(output);
10:
11:      if (usePrimitiveRestartFixedIndex)
12:      {
13:          InputT srcRestartIndex = static_cast<InputT>(gl::GetPrimitiveRestartIndex(sourceType));
14:          DestT destRestartIndex = static_cast<DestT>(gl::GetPrimitiveRestartIndex(destinationType));
15:          for (GLsizei i = 0; i < count; i++)
16:          {
17:              out[i] = (in[i] == srcRestartIndex ? destRestartIndex : static_cast<DestT>(in[i]));
18:          }
19:      }
20:      else
21:      {
22:          for (GLsizei i = 0; i < count; i++)
23:          {
24:              out[i] = static_cast<DestT>(in[i]);
25:          }
26:      }
27:  }
```

Function crashes at line 17 due to wrong memory address because value provied for variable `in` comes directly from function `void ext.drawElementsInstancedANGLE(mode, count, type, offset, primcount);`. At this point two values are directly and fully controlable from JavaScript. The value of `in` at line 8 which is `offset` in JavaScript and value `count` at line 15 which corresponds to variable `count` in JavaScript.

Values provided for `offset` variable in JS are in the range from `00000000` to `7fffffff`.

Unchecked boundaries for these values can lead to out of bounds read and if the value can be freely returned JavaScript, this vulnerability could be turned into info leak to be abused to bypass mitigations.

Crash Information

Windbg output from Chromium x64 head:

```
*******************************************************************************
*                                                                             *
*                        Exception Analysis                                   *
*                                                                             *
*******************************************************************************


KEY_VALUES_STRING: 1

    Key  : AV.Dereference
    Value: String

    Key  : AV.Fault
    Value: Read

    Key  : Analysis.CPU.mSec
    Value: 8421

    Key  : Analysis.DebugAnalysisProvider.CPP
    Value: Create: 8007007e on DESKTOP-JNM9A2J

    Key  : Analysis.DebugData
    Value: CreateObject

    Key  : Analysis.DebugModel
    Value: CreateObject

    Key  : Analysis.Elapsed.mSec
    Value: 48086

    Key  : Analysis.Memory.CommitPeak.Mb
    Value: 2262

    Key  : Analysis.System
    Value: CreateObject

    Key  : Timeline.OS.Boot.DeltaSec
    Value: 22813

    Key  : Timeline.Process.Start.DeltaSec
    Value: 39

    Key  : WER.OS.Branch
    Value: vb_release

    Key  : WER.OS.Timestamp
    Value: 2019-12-06T14:06:00Z

    Key  : WER.OS.Version
    Value: 10.0.19041.1


ADDITIONAL_XML: 1

OS_BUILD_LAYERS: 1

EXCEPTION_RECORD:  (.exr -1)
ExceptionAddress: 00007ffa4897d544 (libglesv2!rx::`anonymous namespace'::ConvertIndexArray<unsigned char,unsigned short>+0x00000000000000b4)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 0000000000000000
   Parameter[1]: 0000000041414141
Attempt to read from address 0000000041414141

FAULTING_THREAD:  000013b0

PROCESS_NAME:  content_shell.exe

READ_ADDRESS:  0000000041414141

ERROR_CODE: (NTSTATUS) 0xc0000005 - The instruction at 0x%p referenced memory at 0x%p. The memory could not be %s.

EXCEPTION_CODE_STR:  c0000005

EXCEPTION_PARAMETER1:  0000000000000000

EXCEPTION_PARAMETER2:  0000000041414141

STACK_TEXT:
000000a8`5c14b050 00007ffa`4897d1d9     : 000000a8`5c14b430 00000000`00000001 00000000`00000000 0000027a`9986b580 : libglesv2!rx::`anonymous
namespace'::ConvertIndexArray<unsigned char,unsigned short>+0xb4
000000a8`5c14b0c0 00007ffa`4897c6ab     : 0000027a`00000000 0000027a`00000001 00000000`00000039 00000000`00000000 : libglesv2!rx::`anonymous
namespace'::ConvertIndices+0x229
000000a8`5c14b4d0 00007ffa`4897c2fc     : 0000d87a`5d1ac6fd 00000000`0000001e 00000000`a0f6fa0f 0000027a`a0f6f950 : libglesv2!rx::`anonymous
namespace'::StreamInIndexBuffer+0x27b
000000a8`5c14b5a0 00007ffa`4897b43d     : 00000001`5c14b998 00000000`00000012 0000d87a`5d1ac49d 00000000`00040000 :
libglesv2!rx::IndexDataManager::streamIndexData+0x2ec
000000a8`5c14b7b0 00007ffa`48b4143d     : 0000d87a`5d1ac94d 00007ffa`485a6b83 000000a8`5c14bf0c 000000a8`5c14c0a8 :
libglesv2!rx::IndexDataManager::prepareIndexData+0x2ad
000000a8`5c14be30 00007ffa`48b40503     : 0000027a`9966cc60 00000001`95cd0000 0000d87a`5d1acf3d 00007ffa`48408a4c :
libglesv2!rx::StateManager11::applyIndexBuffer+0x16d
000000a8`5c14bf50 00007ffa`48ab1f6f     : 000000a8`5c14c5d8 00000000`00000000 0000027a`996b0038 00000000`00000000 :
libglesv2!rx::StateManager11::updateState+0x733
000000a8`5c14c4c0 00007ffa`48ab2258     : 00000000`a0eb21d0 3fffffff`7cfbfffa 0000027a`99375bd0 0000027a`a0eaf3c0 :
libglesv2!rx::Context11::drawElementsImpl+0x5af
000000a8`5c14c720 00007ffa`4855015f     : 00000004`0036baf0 00000000`00000000 0100027a`a0eaf3c0 00007ffa`489440ff :
libglesv2!rx::Context11::drawElementsInstanced+0xe8
000000a8`5c14c7f0 00007ffa`4840ea3b     : 0000027a`99375bd0 00007ffa`6f1ff72c 00007ffa`4957d4d7 00000000`00000008 :
libglesv2!gl::Context::drawElementsInstanced+0x12f
000000a8`5c14c880 00007ffa`484eb5cb     : 00007ffa`4957927a 00000000`00000008 00007ffa`49758900 00007ffa`49758978 :
libglesv2!gl::DrawElementsInstanced+0x27b
000000a8`5c14c960 00007ffa`5904547d     : 00007ffa`49758978 00007ffa`49758978 0100027a`a0ea98e8 0000027a`a0eaf3c0 :
libglesv2!glDrawElementsInstanced+0x4b
000000a8`5c14c9b0 00007ffa`6f365441     : 00000000`00000000 00007ffa`59043302 0000027a`a0ea9e70 0000027a`a0ea7b58 :
gl_wrapper!gl::GLApiBase::glDrawElementsInstancedANGLEFn+0x7d
000000a8`5c14ca20 00007ffa`6f38cfe5     : 0000027a`a0ea7a00 00000005`a0ee8500 00001401`0b2b9d09 0000027a`a0ea7a40 :
gles2!gpu::gles2::GLES2DecoderPassthroughImpl::DoDrawElementsInstancedANGLE+0x91
000000a8`5c14caa0 00007ffa`6f31cfa6     : 00008cc1`c560c678 00007ffa`6f3275c0 00000000`00000100 000000a8`5c14cc38 :
gles2!gpu::gles2::GLES2DecoderPassthroughImpl::HandleDrawElementsInstancedANGLE+0xf5
000000a8`5c14cb30 00007ffa`6f31c67d     : 00007ffa`6f8905b0 01000000`0000000a 0000027a`a0ea7a40 0000027a`a0ea7a78 :
gles2!gpu::gles2::GLES2DecoderPassthroughImpl::DoCommandsImpl<0>+0x286
000000a8`5c14cbe0 00007ffa`85421721     : 00000000`00000000 00000000`00000000 0000027a`99699a50 00007ffa`6f333786 :
gles2!gpu::gles2::GLES2DecoderPassthroughImpl::DoCommands+0x9d
000000a8`5c14cc50 00007ffa`56a251a6     : 0000000a`99699a90 00000000`00000001 00000000`00000000 01000000`00000018 :
gpu!gpu::CommandBufferService::Flush+0x751
000000a8`5c14cea0 00007ffa`56a31d1e     : 000000a8`5c14d3c8 00007ffa`85520fc6 000000a8`5c14d3d0 000000a8`5c14d3d0 :
```

```
gpu_ipc_service!gpu::CommandBufferStub::OnAsyncFlush+0x766
000000a8`5c14d1d0 00007ffa`56a31c4b     : 000000a8`5c14d3d0 00007ffa`56a31a2b 0000027a`998a1230 000000a8`5c14d2f8 :
gpu_ipc_service!base::DispatchToMethodImpl<gpu::CommandBufferStub *,void (gpu::CommandBufferStub::*)(int, unsigned int, const
std::__1::vector<gpu::SyncToken,std::__1::allocator<gpu::SyncToken> > &),std::__1::tuple<int,unsigned
int,std::__1::vector<gpu::SyncToken,std::__1::allocator<gpu::SyncToken> > >,0,1,2>+0x9e
000000a8`5c14d240 00007ffa`56a3193f     : 000000a8`5c14d3d0 000000a8`5c14d3cc 000000a8`5c14d3c8 000000a8`5c14d3c8 :
gpu_ipc_service!base::DispatchToMethod<gpu::CommandBufferStub *,void (gpu::CommandBufferStub::*)(int, unsigned int, const
std::__1::vector<gpu::SyncToken,std::__1::allocator<gpu::SyncToken> > &),std::__1::tuple<int,unsigned
int,std::__1::vector<gpu::SyncToken,std::__1::allocator<gpu::SyncToken> > > >+0x6b
000000a8`5c14d2b0 00007ffa`56a249ca     : 00000000`00000000 000000a8`5c14d5d0 00000000`00000000 000000a8`5c14d5d0 :
gpu_ipc_service!IPC::DispatchToMethod<gpu::CommandBufferStub,void (gpu::CommandBufferStub::*)(int, unsigned int, const
std::__1::vector<gpu::SyncToken,std::__1::allocator<gpu::SyncToken> > &),void,std::__1::tuple<int,unsigned
int,std::__1::vector<gpu::SyncToken,std::__1::allocator<gpu::SyncToken> > > >+0x5f
000000a8`5c14d320 00007ffa`56a22840     : 000000a8`5c14d478 000000a8`5c14d518 000000a8`5c14d518 000000a8`5c14d5e8 :
gpu_ipc_service!IPC::MessageT<GpuCommandBufferMsg_AsyncFlush_Meta,std::__1::tuple<int,unsigned
int,std::__1::vector<gpu::SyncToken,std::__1::allocator<gpu::SyncToken> >
>,void>::Dispatch<gpu::CommandBufferStub,gpu::CommandBufferStub,void,void (gpu::CommandBufferStub::*)(int, unsigned int, const
std::__1::vector<gpu::SyncToken,std::__1::allocator<gpu::SyncToken> > &)>+0x24a
000000a8`5c14d420 00007ffa`b6e6b22a     : 0000027a`a0ea11c0 00007ffa`56a32bd6 0000027a`994cb4d0 00000000`00000000 :
gpu_ipc_service!gpu::CommandBufferStub::OnMessageReceived+0x750
000000a8`5c14d6a0 00007ffa`56a4d669     : 0000027a`a0ea7820 00000000`00000000 0000939c`b71d360c 00007ffa`56a4ad99 :
ipc!IPC::MessageRouter::RouteMessage+0x7a
000000a8`5c14d700 00007ffa`56a4889e     : 010000a8`5c14d7c8 000000a8`5c14d7c8 000000a8`5c14d7c8 00007ffa`86e47f6d :
gpu_ipc_service!gpu::GpuChannel::HandleMessageHelper+0x79
000000a8`5c14d750 00007ffa`56a57af9     : 000000a8`5c14d9f8 00007ffa`853e1e97 0000027a`994cb4a0 0000027a`998a1220 :
gpu_ipc_service!gpu::GpuChannel::HandleMessage+0x21e
000000a8`5c14d990 00007ffa`56a57994     : 0000027a`998a1220 00007ffa`56a57c73 0000027a`99888d48 00007ffa`853e0bd9 :
gpu_ipc_service!base::internal::FunctorTraits<void (gpu::GpuChannel::*)(const IPC::Message &),void>::Invoke<void (gpu::GpuChannel::*)(const
IPC::Message &),base::WeakPtr<gpu::GpuChannel>,IPC::Message>+0x59
000000a8`5c14d9f0 00007ffa`56a578dd     : 000000a8`00000001 00007ffa`853f3dba 00000093`00000004 000000a8`5c14da98 :
gpu_ipc_service!base::internal::InvokeHelper<1,void>::MakeItSo<void (gpu::GpuChannel::*)(const IPC::Message
&),base::WeakPtr<gpu::GpuChannel>,IPC::Message>+0x84
000000a8`5c14da60 00007ffa`56a5785d     : 000000a8`00000000 00007ffa`86e85708 00007b52`df2800dd 00007ffa`853f3cf3 :
gpu_ipc_service!base::internal::Invoker<base::internal::BindState<void (gpu::GpuChannel::*)(const IPC::Message
&),base::WeakPtr<gpu::GpuChannel>,IPC::Message>,void ()>::RunImpl<void (gpu::GpuChannel::*)(const IPC::Message
&),std::__1::tuple<base::WeakPtr<gpu::GpuChannel>,IPC::Message>,0,1>+0x6d
000000a8`5c14dab0 00007ffa`8543922c     : 0000027a`99888d00 00000093`995414d8 0000027a`995414d8 00000000`00000000 :
gpu_ipc_service!base::internal::Invoker<base::internal::BindState<void (gpu::GpuChannel::*)(const IPC::Message
&),base::WeakPtr<gpu::GpuChannel>,IPC::Message>,void ()>::RunOnce+0x5d
000000a8`5c14db00 00007ffa`8543896b     : 000000a8`5c14dbc8 00007ffa`86cb4307 000000a8`00000000 00007ffa`86e85708 :
gpu!base::OnceCallback<void ()>::Run+0x7c
000000a8`5c14db60 00007ffa`8544d59f     : 0000027a`998ae4c8 00007ffa`8544af93 0000027a`998ae4c8 00007ffa`8544b293 :
gpu!gpu::Scheduler::RunNextTask+0x92b
000000a8`5c14de40 00007ffa`8544d53b     : 0000027a`998ae4c8 00007ffa`8544af4b 000000a8`5c14df78 000000a8`5c14df58 :
gpu!base::internal::FunctorTraits<void (gpu::Scheduler::*)(),void>::Invoke<void (gpu::Scheduler::*)(),base::WeakPtr<gpu::Scheduler>>+0x1f
000000a8`5c14de80 00007ffa`8544d4c9     : 000000a8`5c14df30 00007ffa`86cb64e7 000000a8`5c14df40 00007ffa`86f902a5 :
gpu!base::internal::InvokeHelper<1,void>::MakeItSo<void (gpu::Scheduler::*)(),base::WeakPtr<gpu::Scheduler>>+0x4b
000000a8`5c14dec0 00007ffa`8544d46d     : 00007ffa`872b3468 00007ffa`872b3468 0000939c`b71d3e4c 00007ffa`86cb6423 :
gpu!base::internal::Invoker<base::internal::BindState<void (gpu::Scheduler::*)(),base::WeakPtr<gpu::Scheduler> >,void ()>::RunImpl<void
(gpu::Scheduler::*)(),std::__1::tuple<base::WeakPtr<gpu::Scheduler> >,0>+0x49
000000a8`5c14df10 00007ffa`86cb208c     : 00000003`5c14e190 0000027a`95f01340 0000939c`b71d3e4c 00007ffa`86ea5810 :
gpu!base::internal::Invoker<base::internal::BindState<void (gpu::Scheduler::*)(),base::WeakPtr<gpu::Scheduler> >,void ()>::RunOnce+0x5d
000000a8`5c14df60 00007ffa`86ea530f     : 0000027a`95cf98d0 00007ffa`86cb42de 0000027a`95cf98c8 00007ffa`86ec1faa :
base!base::OnceCallback<void ()>::Run+0x7c
000000a8`5c14dfc0 00007ffa`86ef9162     : 00000000`00000000 00007ffa`86f902a5 00000000`00000000 000000a8`5c14e268 :
base!base::TaskAnnotator::RunTask+0x70f
000000a8`5c14e1d0 00007ffa`86ef86de     : 0000939c`b71d046c 00000000`00000000 00000000`00c62f84 00000000`00c62f84 :
base!base::sequence_manager::internal::ThreadControllerWithMessagePumpImpl::DoWorkImpl+0x802
000000a8`5c14e550 00007ffa`86d67e1e     : 0000027a`95d0cf40 000000a8`5c14e6c8 0000027a`95d0cf48 000000a8`5c14e6c8 :
base!base::sequence_manager::internal::ThreadControllerWithMessagePumpImpl::DoWork+0xfe
000000a8`5c14e640 00007ffa`86efa18f     : 0000027a`994f5df8 00007ffa`86d2ae1b 00000000`00000000 00000001`00000000 :
base!base::MessagePumpDefault::Run+0xae
000000a8`5c14e6e0 00007ffa`86e2e012     : 00000000`00000000 00000000`000000e6 0000939c`b71d086c 00007ffa`870613f8 :
base!base::sequence_manager::internal::ThreadControllerWithMessagePumpImpl::Run+0x36f
000000a8`5c14e930 00007ffa`7b81d414     : 00000000`000003d8 00000000`00000600 00000000`00000000 000000a8`5c14eb00 :
base!base::RunLoop::Run+0x342
000000a8`5c14ea70 00007ffa`7fa2e470     : 0000027a`95d0f920 0000027a`95d13080 000000a8`5c14f000 0000027a`95d13080 :
content!content::GpuMain+0xaf4
000000a8`5c14efb0 00007ffa`7fa2f7f7     : 000000a8`5c14f070 00000000`00000000 00000000`00000000 00000000`00000000 :
content!content::RunOtherNamedProcessTypeMain+0xe0
000000a8`5c14f020 00007ffa`7fa2a1d7     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 :
content!content::ContentMainRunnerImpl::Run+0x2f7
000000a8`5c14f120 00007ffa`51282498     : 0000027a`95cd0000 00007ffa`40000062 00000000`0000009c 00000000`000000b0 :
content!content::ContentServiceManagerMainDelegate::RunEmbedderProcess+0x37
000000a8`5c14f160 00007ffa`7fa2e228     : 000000a8`5c14f628 000000a8`5c14f628 000000a8`5c14f628 00000000`00000000 :
embedder!service_manager::Main+0x858
000000a8`5c14f4a0 00007ff6`e3391139     : 00007ff6`e6037eb8 00007ff6`e59b2447 00007ff6`e5410930 00007ff6`e602ecc0 :
content!content::ContentMain+0x88
000000a8`5c14f550 00007ff6`e59b49d2     : 00000000`00000000 00007ff6`e59b260d 00000000`00000000 00007ff6`e5ce4508 :
content_shell!wWinMain+0x139
000000a8`5c14f650 00007ff6`e59b4b0e     : 00007ff6`e5ce4400 00007ff6`e5ce44e0 00000000`00000000 00000000`00000000 :
content_shell!invoke_main+0x32
000000a8`5c14f690 00007ff6`e59b4b8e     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 :
content_shell!__scrt_common_main_seh+0x12e
000000a8`5c14f700 00007ff6`e59b4ba9     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 :
content_shell!__scrt_common_main+0xe
000000a8`5c14f730 00007ffa`ecb86fd4     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 :
content_shell!wWinMainCRTStartup+0x9
000000a8`5c14f760 00007ffa`ed01cec1     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 :
KERNEL32!BaseThreadInitThunk+0x14
000000a8`5c14f790 00000000`00000000     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 :
ntdll!RtlUserThreadStart+0x21


FAULTING_SOURCE_LINE:  C:\Work\Browsers\chromium_stable\src\third_party\angle\src\libANGLE\renderer\d3d\IndexDataManager.cpp

FAULTING_SOURCE_FILE:  C:\Work\Browsers\chromium_stable\src\third_party\angle\src\libANGLE\renderer\d3d\IndexDataManager.cpp

FAULTING_SOURCE_LINE_NUMBER:  44

FAULTING_SOURCE_CODE:  
    40:         InputT srcRestartIndex = static_cast<InputT>(gl::GetPrimitiveRestartIndex(sourceType));
    41:         DestT destRestartIndex = static_cast<DestT>(gl::GetPrimitiveRestartIndex(destinationType));
    42:         for (GLsizei i = 0; i < count; i++)
    43:         {
>   44:             out[i] = (in[i] == srcRestartIndex ? destRestartIndex : static_cast<DestT>(in[i]));
    45:         }
    46:     }
    47:     else
    48:     {
    49:         for (GLsizei i = 0; i < count; i++)


SYMBOL_NAME:  libglesv2!rx::`anonymous namespace'::ConvertIndexArray<unsigned char,unsigned short>+b4

MODULE_NAME: libglesv2

IMAGE_NAME:  libglesv2.dll
```

```
STACK_COMMAND:  ~20s ; .cxr ; kb

FAILURE_BUCKET_ID:  NULL_POINTER_READ_c0000005_libglesv2.dll!rx::_anonymous_namespace_::ConvertIndexArray_unsigned_char,unsigned_short_

OS_VERSION:  10.0.19041.1

BUILDLAB_STR:  vb_release

OSPLATFORM_TYPE:  x64

OSNAME:  Windows 10

IMAGE_VERSION:  2.1.0.0

FAILURE_ID_HASH:  {b019e0bb-749b-f43f-cf2e-5250983065ba}

Followup:     MachineOwner
---------
```

TIMELINE

2020-07-13 - Vendor Disclosure
2020-07-15 - Vendor patched
2020-10-13 - Public Release

CREDIT

Discovered by Marcin Towalski of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT TALOS-2020-1050

NEXT REPORT TALOS-2020-1148